UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IMPLEMENTACIÓN DE SISTEMA DE VALIDACIÓN Y REPORTE PARA EL REGISTRO DE ENTREGAS DE RACIONES PAE DE JUNAEB

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

JORGE IGNACIO LOBOS PONCE

PROFESOR GUÍA:
JOCELYN SIMMONDS WAGEMANN

MIEMBROS DE LA COMISIÓN:
JOSÉ PINO URTUBIA
MARISA ERNST ELIZALDE

Este trabajo ha sido parcialmente financiado por: GeoVictoria S.A.

SANTIAGO DE CHILE
2020

RESUMEN

La Junta Nacional de Auxilio Escolar y Becas (JUNAEB) es un organismo de la administración del Estado, responsable de administrar los recursos estatales destinados a velar por los niños, niñas y jóvenes chilenos en condición de vulnerabilidad bio-psico-social, para que ingresen, permanezcan y tengan éxito en el Sistema Educativo.

Uno de los programas de JUNAEB es el Programa de Alimentación Escolar (PAE), cuya finalidad es entregar diariamente servicios de alimentación. Los adoptantes objetivo del PAE son los alumnos y alumnas en condición de vulnerabilidad de Establecimientos Educacionales Municipales y Particulares Subvencionados del país adscritos al programa. La entrega de raciones se realiza durante el año lectivo, con el objeto de mejorar la asistencia a clases de los alumnos y contribuir a evitar la deserción escolar.

Con el objetivo de mejorar la entrega oportuna del servicio, JUNAEB quiere cuantificar la entrega oportuna de alimentos a cada alumno, pues contrata a decenas de empresas en todo el país para la entrega de raciones asociadas al PAE. JUNAEB necesita que las empresas de reparto de alimentos incluyan en sus actividades un sistema de reconocimiento e identificación de estudiantes beneficiarios/as del programa. El registro y control de asistencia a los respectivos comedores de los colegios, bajo su asignación, debe ser con base en perfiles biométricos. Con estos perfiles, las empresas obtendrán la cantidad de raciones entregadas y su autorización, y JUNAEB obtiene la identificación de los alumnos. Dicho sistema es un requisito pedido por JUNAEB en sus bases de licitación para adjudicar la entrega de raciones de alimentación.

Debido a esto, se propone un sistema que consta de un sitio web, una API y una aplicación de integración que se comunique con los sistemas de JUNAEB. Con esto, se logra la identificación, verificación, chequeo, respaldo y reportes de gestión del proceso de entrega de raciones del PAE.

A pesar de que hubo problemas de comunicación con JUNAEB a mitad del proceso, el sistema era lo suficientemente independiente para afrontar esta contingencia, por lo que se terminó el desarrollo bruto de los objetivos propuestos. El trabajo pendiente tiende a la validación de estos sistemas con la contraparte de JUNAEB y a mejoras respecto al crecimiento de los datos en la plataforma web (como un sistema de paginación).

Sin perjuicio de ello, los conocimientos técnicos adquiridos a lo largo de este proceso permiten que este sistema sea mantenible y evolucionable en el tiempo.

Dedicado a todos los jóvenes de Chile,
con la esperanza que su alimentación sea mejor y más justa.

Gracias a quienes me apoyaron en la vida

Agradecimientos

A mi madre, que siempre ha hecho lo posible para criarme, a pesar de la distancia.

A mis compañeras y compañeros de trabajo, que cada día me enseñan y me exigen ser un mejor profesional. Ellas y ellos hacen del trabajo un espacio más llevadero, lugar donde las risas no faltan.

A mis compañeras y compañeros de universidad, quienes hicieron mi estadía en esta institución muy agradable, y me dieron preciosos momentos.

A los grupos organizados, agradecer a Seishin, CRI, Radio Integral, entre muchos otros, por las grandiosas actividades de las que fui participe. Deseo que sigan alegrando la vida universitaria a muchos estudiantes.

Al cuerpo docente y funcionarios del DCC, quienes se esfuerzan día a día por hacer del departamento un lugar mejor.

Y por último y más importante, a mi pareja, Javiera Soto, por su compañía y cariño incondicional, demostrándome día a día con su perseverancia que se puede ser mejor persona.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto

La Junta Nacional de Auxilio Escolar y Becas (JUNAEB) es un organismo de la administración del Estado, creado en 1964 por la Ley Nº 15.720, responsable de administrar los recursos estatales destinados a velar por los niños, niñas y jóvenes chilenos en condición de vulnerabilidad bio-psico-social, para que ingresen, permanezcan y tengan éxito en el Sistema Educativo. [18]

Uno de los programas de JUNAEB es el Programa de Alimentación Escolar (PAE), cuya finalidad es entregar diariamente servicios de alimentación (desayunos, almuerzos, once, colaciones y cenas según corresponda). Los adoptantes objetivo del PAE son los alumnos y alumnas en condición de vulnerabilidad de Establecimientos Educacionales Municipales y Particulares Subvencionados del país adscritos al programa. La entrega de raciones se realiza durante el año lectivo, en los niveles de Educación Parvulario (Pre-Kínder y Kínder), Básica, Media y Adultos, con el objeto de mejorar la asistencia a clases de los alumnos y contribuir a evitar la deserción escolar. [18]

Con el objetivo de mejorar la entrega oportuna del servicio, el directorio de JUNAEB, bajo el alero del MINEDUC, quiere cuantificar la entrega oportuna de alimentos, pues contrata a decenas de empresas en todo el país para la entrega de raciones asociadas al PAE. Las empresas de reparto de alimentos requieren para sus actividades un sistema de reconocimiento e identificación de estudiantes beneficiarios/as del programa. El registro y control de asistencia a los respectivos comedores de los colegios, bajo su asignación, debe ser con base en perfiles biométricos. Dicho sistema es un requisito pedido por JUNAEB en sus bases de licitación para adjudicar la entrega de raciones de alimentación. [14]

JUNAEB quiere que este sistema sea un medio por el cual se logre la identificación, verificación, chequeo, respaldo y reportes de gestión del proceso de entrega de raciones del PAE. Un ejemplo de proceso sería:

- Un alumno perteneciente al programa llega a las 13:00 hrs. a buscar su almuerzo al casino del colegio.

- El alumno registra su asistencia con el profesor PAE asignado al establecimiento por JUNAEB.
- Dicho alumno retira su ración de alimento.
- El alumno almuerza, y al terminar, entrega su bandeja en el área dispuesta para ello.

Nótese que la forma de este proceso de entrega es de acuerdo a la realidad de cada establecimiento, pues depende de la disposición de espacios, la cantidad de alumnos asociados al PAE, entre otros. A pesar de ello, las raciones están normadas en la minuta entregada por JUNAEB a las empresas adjudicadas en la licitación de entrega de raciones, y es revisada por un/a supervisor/a JUNAEB. [9] De esta forma, se obtiene la información necesaria para:

- Registrar la real asistencia del/a usuario/a al comedor para acceder a la alimentación, e identificar quienes lo hacen, pudiendo verificar si corresponden o no a la población objetivo del programa.
- Cuantificar de manera transparente y oportuna los servicios entregados por el PAE.
- Ajustar la preparación de alimentos en función de la asistencia diaria de la comunidad estudiantil al establecimiento.
- Mejorar los procesos de gestión del PAE, disminuyendo los tiempos involucrados en la certificación y pago de las raciones efectivas, ya que al ser este un proceso automático disminuyen la posibilidad de error.

1.2. Contexto del problema

Actualmente, el sistema de reparto de raciones PAE hacia los colegios se realiza mediante empresas distribuidoras de alimentos. Estas son elegidas mediante licitaciones que JUNAEB realiza para entregar colaciones PAE a los colegios.

El registro de raciones entregadas por el colegio a los alumnos se realiza a lápiz y papel por un profesor PAE. Este profesor es la persona encargada de ser el nexo entre el colegio, al que se le entregan las raciones, y la empresa de alimentación, enviando en un plazo entre 4 a 6 semanas las listas con los alimentos entregados.

El problema que surge con esto es que el registro de raciones usadas está sujeto a vicios. Por nombrar algunos:

- Entrega de alimentos a niños que no corresponda.
- Errores de conteo manual de las raciones entregadas en el periodo.
- Inexistencia de un tiempo fijo de entrega del conteo de raciones usadas a las empresas de reparto.

- Diferencias en el número de colaciones entregadas a alumnos entre el reporte que entregan los profesores PAE a las empresas de distribución de alimentos y el informe entregado por las distribuidoras a JUNAEB, cayendo en la posibilidad de manipulación de datos.

Por último, debido a la cantidad de alumnos beneficiarios, estimados en más de 3.000.000 a lo largo de Chile, la cantidad de recursos destinados corresponde a cerca del 75 % del presupuesto anual destinado a JUNAEB (entre 164 y 574 mil millones de pesos declarados cada año entre 2005 y 2010), por lo que el manejo del dinero es clave para la institución gubernamental. [25]

Debido a los problemas recién mencionados, JUNAEB solicitó en las bases de licitación del año 2018 para las empresas de repartición de alimentos que incluyeran un sistema de registro biométrico y de entrega de reportes. Con ello buscan centralizar los datos de los alumnos sujetos al PAE y las raciones que consumen, y facilitar análisis para futuras disposiciones de presupuestos. En caso de no entregar dichos reportes, las empresas arriesgan un término anticipado de contrato.

1.3. Solución

Actualmente, el alumno en proceso de titulación se encuentra trabajando en GeoVictoria S.A., empresa dedicada al control y gestión de asistencia. Dado que las empresas de reparto de alimento no cuentan con el personal ni con el conocimiento técnico asociado a desarrollar una solución que les permita entregar los reportes según las necesidades de JUNAEB, GeoVictoria vio la oportunidad de tomar a estas empresas como clientes. Se planea entregar a las empresas de reparto de alimentos licitadas un sistema que les permita coordinar los registros de entrega de raciones. Además, se pone a disposición una plataforma que permita generar reportes a pedido, centralizando la información.

Por lo general, un alumno para postular al PAE se inscribe con el o la asistente social de cada establecimiento entre noviembre y diciembre del año previo. Este alumno llena una ficha de ingreso al programa, entregando el Registro Social de Hogares, y algún antecedente adicional (enfermedades del grupo familiar, alergias, entre otros), recibiendo los resultados en marzo del año siguiente. Los alumnos pertenecientes al tramo del 60 % de hogares con mayor vulnerabilidad según el registro social quedan automáticamente seleccionados. Cabe destacar que JUNAEB asigna de antemano el número de beneficiarios por establecimiento.

El alumno adscrito al PAE llega al casino o lugar de entrega de colaciones de su colegio a buscar la ración que le corresponde. Para poder retirar esa ración, se marca en una nómina cuadriculada que incluye su nombre y día de entrega, mostrando alguna identificación al profesor PAE (cédula de identidad o Tarjeta Nacional Estudiantil, conocida también como Pase Escolar). Una vez confirmada su asistencia, el alumno pasa a un mesón o área de porcionamiento indicada a buscar sus alimentos. En caso de recibir algún implemento para consumirlos en el lugar (bandeja, tenedor, cuchara, entre otros), al terminar de comer debe dejar los utensilios en el área de entrega de bandejas, procediendo a retirarse.

En el caso de los alumnos, se reemplaza la firma en la nómina de papel por una máquina de registro de marcas mediante huella digital que entrega un vale de colación. Primero que todo, JUNAEB mediante el SINAE (Sistema Nacional de Asignación con Equidad) chequea las condiciones socioeconómicas de cada estudiante del país, con base en la información entregada por distintas plataformas, tales como sistema de salud, pertenencia a los programas "Ingreso Ético Familiar" o "Chile Solidario", información del Registro Civil, matrículas del MINEDUC, entre otras. Así, se le asigna un puntaje de vulnerabilidad para la asignación de beneficios y los alumnos pueden terminar su escolaridad básica y media[19].

Para el proceso de inscripción, una vez que se confirme que el alumno pertenece al programa con el o la asistente social, el alumno debe acercarse al profesor PAE para realizar el registro inicial en el reloj control, ingresando cuatro veces las huellas dactilares de cada dedo índice. Esta información es consumida por una API alojada en las instancias de servidor dispuestas por GeoVictoria, siendo guardadas en una base de datos SQL Server, extensibles en un futuro a sistemas de Azure Storage. De esta manera, se tienen registros inequívocos de quién, a qué fecha y hora y qué ración retirará un estudiante, lo que evita los problemas listados en el capítulo 2 Estado Actual. Por otra parte, dada la arquitectura de base de datos a utilizar, se solventa el tema de que la cantidad de registros sea escalable: Al usar el servicio Azure Serverless para alojar la base de datos SQL, se puede ajustar el tamaño dependiendo de la cantidad de datos almacenado y el cobro mediante la cantidad de transacciones realizadas.

GeoVictoria lleva más de 5 años usando los equipos de reloj control con huellero integrado, por lo que se tiene en cuenta la seguridad de los datos personales de los usuarios. Los relojes control cuentan con una entrada de energía para enchufar a la línea de corriente, un lector de huella dactilar, un panel con una pantalla TFT y botonera numérica y una impresora para entregar tickets, estando protegido y anclado por una caja de acero inoxidable. Estos equipos usan un algoritmo de hashing que transforma la imagen de la huella digital a un registro alfanumérico, lo que evita que se guarde información sensible. En el eventual caso que alguien extraiga una máquina de registro, la información que contiene no se puede descifrar sin tener la llave de seguridad del algoritmo de hashing. Dicha llave es entregada por la empresa proveedora de las máquinas a GeoVictoria.

En el caso de las empresas de alimentación y JUNAEB, los usuarios designados por estos tendrían acceso a una plataforma que permite generar reportes y visualizar un dashboard alimentado por las marcas registradas por los alumnos en los reloj control. De esta manera, tanto JUNAEB como las distribuidoras tienen a disposición, de manera sencilla, la cantidad de raciones realmente entregadas a alumnos. En caso de querer realizar reportes con una selección de usuarios acotada, se dispone de una sección de filtros en las que el usuario escribe el criterio o identificador a filtrar.

En términos económicos, los relojes control son la alternativa más barata en cuanto a marcaje biométrico, costando menos de 100USD cada reloj. En caso de que alguno de los componentes necesiten reemplazo, se pueden cambiar fácilmente (la capacitación a los profesores PAE incluye cambio de componentes modulares como el papel y tinta de impresión de tickets de colación), y envían los datos en tiempo real en caso de tener conexión a internet. En caso contrario, se guardan los datos cifrados y se envían en una cola que se borra localmente luego de ser enviada.

1.4. Objetivos

Objetivo General

El objetivo de esta memoria es diseñar, construir y validar un sistema que centralice y facilite el análisis de las raciones repartidas a los alumnos mediante el programa PAE de JUNAEB.

Dicha aplicación cuenta con una API que recibe las marcas biométricas de los alumnos beneficiarios usuarios del PAE, y una página web que muestre a los clientes un dashboard con reportes a medida.

Objetivos Específicos

1. Diseñar las interfaces, modelo y API del sistema, eligiendo la arquitectura tecnológica adecuada para la solución del problema.
2. Probar la API de reloj control: cómo enviar y recibir información a través de los equipos, cómo configurarlos para que se comuniquen con los parámetros adecuados.
3. Probar mediante test de stress que el sistema funcione bajo condiciones críticas. Este debe ser capaz de recibir por lo menos 1.000 transacciones de escritura por minuto en horarios de alta demanda. Este número se determinó debido a que la cantidad de alumnos que se contempla en el piloto de este proyecto son 10.000 alumnos, y se espera que en un plazo de 10 minutos todos los alumnos tengan su comida en un caso de alta demanda.
4. Implementar una aplicación que reciba como entrada los datos enviados por los reloj control al motor de base de datos, y que genere automáticamente, o a pedido, consultas o reportes con la información de las raciones repartidas.
5. Validar el trabajo realizado una vez cada 6 semanas con personal del área de pruebas (QA) de GeoVictoria, revisando los tiempos de ejecución, los datos de entrada y la estabilidad del sistema.

La metodología de trabajo fue de desarrollo iterativo incremental. Así, se pudo validar y marcar metas claras en el plazo entre validaciones.

Se eligió SQL Server como almacén de datos provisorio, escalable a Azure Storage para registro de raciones y finalmente a Hadoop para el manejo de datos masivo. Todo esto, para poder guardar toda la información de los tickets emitidos en particiones por rango temporal. Se usó `Microsoft Power BI` para realizar una interfaz que permite a los usuarios obtener una visualización sencilla de la información relevante.

1.5. Alcances y solución propuesta

Los alcances de esta memoria se definen en conjunto con GeoVictoria, pues hay elementos que escapan del campo del alumno, como manejo organizacional, realización de talleres, implantación de las máquinas, entre otros.

Así, se le muestra el contexto del problema al alumno y se le detallan las tareas que tendría que realizar y en qué puede apoyarlo GeoVictoria:

- La petición, importación e instalación de los reloj control corre por parte de cada empresa de alimentación, mientras que la configuración de la información a registrar, cómo se entregará y dónde se alojará será tarea del alumno.
- GeoVictoria posee la mayoría de sus soluciones de software en Azure, el servicio en la nube de Microsoft. La empresa proveerá al alumno de la infraestructura de hardware y software necesarias para realizar la implementación de la solución y el servicio de reportes, siendo el alumno el encargado de investigar, dado el contexto del problema, la arquitectura tecnológica adecuada para realizar la aplicación.
- Dado que las empresas de reparto de raciones pidieron el desarrollo de la solución a GeoVictoria, se dejará a disposición del alumno el contacto con las entidades correspondientes para realizar la toma de requerimientos, poder recibir retroalimentación de la aplicación en su fase de testeo y realizar entrega a las empresas cliente.
- La capacitación a profesores PAE en el uso de las máquinas de registro de huella será hecha por personal capacitado de las empresas de alimentación, quienes son instruidos a su vez por personal técnico de GeoVictoria, por lo que el alumno no participará en dicho proceso. Lo que sí debe hacer es entregar la documentación adecuada a los jefes de proyecto y personal de gerencia que corresponda, para así entregar la información de mejor manera al personal técnico.
- La validación del sistema fue realizada por GeoVictoria, pues a pesar que no ocurrió la situación mencionada inicialmente en esta memoria (que JUNAEB modifique las bases para futuras licitaciones), sí ocurrió que se perdió el contacto con la entidad estatal. Se propone que la empresa tome a futuro el repositorio y continúe el desarrollo para las peticiones de los clientes, luego que termine la memoria del alumno.

1.6. Estructura del informe

El resto del documento consta de cinco capítulos. En el Capítulo 2 se entrega un glosario con las explicaciones de términos recurrentes a usar, un resumen de las características de los sistemas biométricos, ejemplos de flujo de alimentación de los alumnos, y una revisión bibliográfica.

En el Capítulo 3 se trata el análisis de los factores explicados en el capítulo 2, lo que implica que se definen las historias de usuario de estudiantes, profesores PAE y administradores para

el proyecto desarrollado, y se revisa el diseño del nuevo sistema en cuanto a su arquitectura lógica y la revisión del modelo entidad-relación de la base de datos.

Luego, en el Capítulo 4 se explica la implementación del sistema diseñado con base en los tres repositorios creados, las herramientas utilizadas para ello y los desafíos que se encontraron.

En el Capítulo 5 se muestran los resultados de la validación realizada con el componente de API propio. En caso del proyecto de integración, se explica uno de los posibles criterios de validación en caso de retomar contacto con JUNAEB, y en el componente de plataforma se muestran los resultados y mejoras futuras propuestas por los voluntarios de testeo.

Finalmente, el capítulo 6 muestra un resumen del trabajo realizado, las conclusiones de éste y el trabajo por hacer a futuro.

Capítulo 2

Estado Actual

Considerando la introducción realizada en el Capítulo 1, en este capítulo se describirán los términos a usar a lo largo del proyecto, para luego dar información relevante respecto a los sistemas biométricos. Luego, se da un ejemplo de flujo de alimentación de estudiantes, bajo el cual se arman las historias de usuario a explicar en el siguiente capítulo, finalizando con una revisión bibliográfica.

Tal y como se detalló en la sección 1.1, JUNAEB contrata mediante licitación pública el servicio de suministro de raciones alimenticias, correspondiente a desayunos, onces, almuerzos, cenas, colaciones y tercer servicio, entre otras, en las unidades territoriales que esta organización determine. El volumen de raciones alimenticias diarias a licitar, alcanzó una cantidad estimada de 1.527.804 el año 2019, de las cuales: corresponden a la Junta Nacional de Auxilio Escolar y Becas JUNAEB 1.235.768, a la Junta Nacional de Jardines Infantiles JUNJI 205.635 y a Integra 86.401. Cabe señalar que JUNAEB, JUNJI o Integra entregarán raciones de emergencia en coordinación con los organismos estatales que organicen e implementen acciones frente a situaciones de emergencia y catástrofe, tales como la pandemia por COVID-19 que afecta actualmente a Chile y el mundo.

Al 2020, hay más de 10 mil profesores que voluntariamente ayudan a JUNAEB a registrar las raciones que son efectivamente servidas a los alumnos. Estos "Profesores PAE" son docentes que entregan parte de su tiempo en controlar que las empresas concesionarias sirvan las raciones a los niños que les corresponde, porque solo así JUNAEB puede pagar por plato servido. El registro de raciones entregadas por el colegio a los alumnos se realiza a lápiz y papel por un profesor PAE, que envía en un plazo entre 4 a 6 semanas las listas con los alimentos entregados.

El problema que surge con esto es que el registro de raciones usadas está sujeto a vicios. Por nombrar algunos:

- Entrega de alimentos a niños que no corresponda.
- Errores de conteo manual de las raciones entregadas en el periodo.
- Inexistencia de un tiempo fijo de entrega del conteo de raciones usadas a las empresas de reparto.

- Diferencias en el número de colaciones entregadas a alumnos entre el reporte que entregan los profesores PAE a las empresas de distribución de alimentos y el informe entregado por las distribuidoras a JUNAEB, cayendo en la posibilidad de manipulación de datos.

Por último, debido a la cantidad de alumnos beneficiarios, la cantidad de recursos destinados corresponde a cerca del 75 % del presupuesto anual destinado a JUNAEB (entre 164 y 574 mil millones de pesos declarados cada año entre 2005 y 2010), por lo que el manejo del dinero es clave para la institución gubernamental. [25]

Debido a los problemas mencionados anteriormente y reiterando lo expuesto al final de la sección 1.2, JUNAEB agregó en las bases de licitación del año 2018 a las empresas de reparto de alimentos que incluyeran un sistema de registro biométrico y de entrega de reportes. Con ello buscan centralizar los datos de los alumnos sujetos al PAE y las raciones que consumen, y así facilitar análisis para futuras disposiciones de presupuestos. Cabe destacar que las empresas que incumplan esta condición son sancionadas mediante un término anticipado de contrato, en caso de no entregar dichos reportes.

2.1. Glosario

Cuando hablemos de reloj o reloj control, se refieren a lo mismo: el aparato que realiza los procesos de registros y marcaje de los alumnos. La Figura 2.1a muestra un reloj control a punto de instalar, por anclar a la pared y mostrando el cofre metálico de seguridad (caja blanca que dice "GeoVictoria"), la botonera, el lector de huella dactilar (panel verde en la esquina superior derecha) y una ranura en la esquina inferior derecha del cofre donde salen los tickets impresos. La Figura 2.1b muestra la impresora, en forma independiente del cofre. Está conectada al reloj control y oculta dentro del cofre, dejando solo el espacio para poder recoger los tickets mediante la ranura en el cofre.

Cuando hablemos de marcas registradas, registros, marcas o registro de marcas, se refiere al proceso de colocar la huella dactilar para reconocer al alumno en base al hash guardado en la base de datos interna del reloj control.

Al referirse más adelante a ADMS o servidor de relojes, se refiere al componente que recoge las marcas de los relojes control y verifica que la comunicación esté disponible (se tratará con más detalle en la Sección 3.2.1).

Cuando se mencione tickets o vouchers, se refiere al registro de marcas procesado para mostrar en el sistema web o para enviar a JUNAEB. Una marca se almacena como un voucher/ticket en la base de datos y se imprime para que el usuario tenga una copia.

En caso de usar los términos integración o aplicación de integración, se refiere al componente IntegracionJunaeb, que permite la comunicación entre los sistemas de datos pertenecientes a JUNAEB y los usados en esta memoria.

Cuando se habla de plataforma, sistema web o aplicación web, se refiere al componente PlataformaJunaeb, que muestra los reportes detallados de alumnos, profesores y raciones,

(a) Reloj control en cofre de seguridad.

(b) Reloj control en cofre de seguridad y su impresora.

además del dashboards resumen a los usuarios administradores.

2.2. Información respecto a sistemas biométricos

Existen en el mercado muchas soluciones de software, que permiten la identificación de estudiantes en el contexto de petición de raciones, pero estas son usadas como sistema de canje de productos dentro de un casino o en el comercio. Esto se hace cargando un saldo a un identificador, que podrá ser usado por el alumno.

Un ejemplo de esto es la Beca de Alimentación de Educación Superior (BAES), en la que durante el año lectivo, se cargan $32.000 mensuales a una tarjeta con banda magnética con el nombre del alumno, para que este pueda solventar sus gastos de alimentación [23].

En el caso del PAE, se requiere que actúe como un sistema de control de acceso. Ejemplos de sistemas de control de acceso son ofrecidos por IdentiSys [17], BadgePass [11] y Capture Technologies [13], permitiendo identificar al alumno para poder emitir un ticket personalizado.

Al día de hoy, las soluciones de identificación se separan en tres grandes categorías [22]:

- Basadas en conocimiento, por ejemplo contraseñas o llaves públicas y privadas.

- Basadas en objetos, como tarjetas NFC o con banda magnética [12], tokens con objetos que se lleven en la ropa (como brazaletes, relojes, anillos) o dispositivos generadores de claves de uso único (por ejemplo, Digipass usado por Banco de Chile).
- Basadas en identidad, más conocidas como biométricas, tales como uso de huellas dactilares, escaneo de iris ocular, escaneo de manos [16], autenticación por voz, e incluso reconocimiento de patrones de caminata y de tecleo.

JUNAEB limita en su licitación a usar solamente alternativas basadas en identidad (biométricas), por lo que, a pesar de ser más económicas, las basadas en conocimiento y en objetos quedan descartadas como posible solución.

Entre las soluciones biométricas, la alternativa más difundida debido a su relación costo eficiencia es el reconocimiento de huellas dactilares. Cada reloj control completo, con su cofre de seguridad, cuesta 100USD (este precio es un valor aproximado de cada equipo al comprar al por mayor al distribuidor), mientras que un solo escáner de iris cuesta 480USD [15], un sistema de reconocimiento facial alrededor de 320USD [10].

Otro factor a tomar en cuenta es la facilidad de uso del dispositivo. Estos equipos deben ser anclados en alguna pared, pero la varianza de estatura entre alumnos de pre-kinder y pertenecientes al programa de nivelación de estudios para adultos, pasando por educación básica y media, es un factor a tomar en cuenta.

Por esto, usar un sistema que reconozca un elemento perteneciente a la cabeza, sin mucha movilidad (e.g., rostro, iris, voz) no resulta viable, a diferencia de los lectores de huella digital, en los que en brazo posee más facilidad para alcanzar el reloj control.

2.3. Ejemplos de flujo de alimentación

En esta sección se explicarán algunos casos básicos de flujo de alimentación de los estudiantes. Estos flujos supondrán casos en que los estudiantes reciben o no sus raciones, así como irregularidades o vicios presentes al momento de entregar las raciones.

En un colegio de la capital, Pedro, Pablo, Juan y Diego son cuatro amigos que van a almorzar al casino del colegio. Martín es un estudiante que pertenece al PAE, también amigo de ellos, pero que no utiliza su beneficio regularmente. Diego es un estudiante que usualmente llega atrasado a sus clases y al almuerzo. Por último, Carlos es un estudiante que no posee el beneficio, pero que lo necesita.

- Primero llega Pedro, que pertenece al PAE, y por llegar a la hora designada de almuerzo, el profesor PAE correspondiente escribe su nombre en la lista del día. Pedro pasa a la fila para recibir su ración del día y espera a sus amigos.
- Junto a Pedro llega Pablo, quien NO pertenece al PAE. Al ser un colegio grande, Pablo toma el nombre de un estudiante que si pertenece al PAE, a lo que el profesor PAE lo deja pasar sin verificar correctamente su identidad. Pablo pasa con el nombre de Martín Vallejo, y pasa detrás de su amigo a esperar la ración designada.

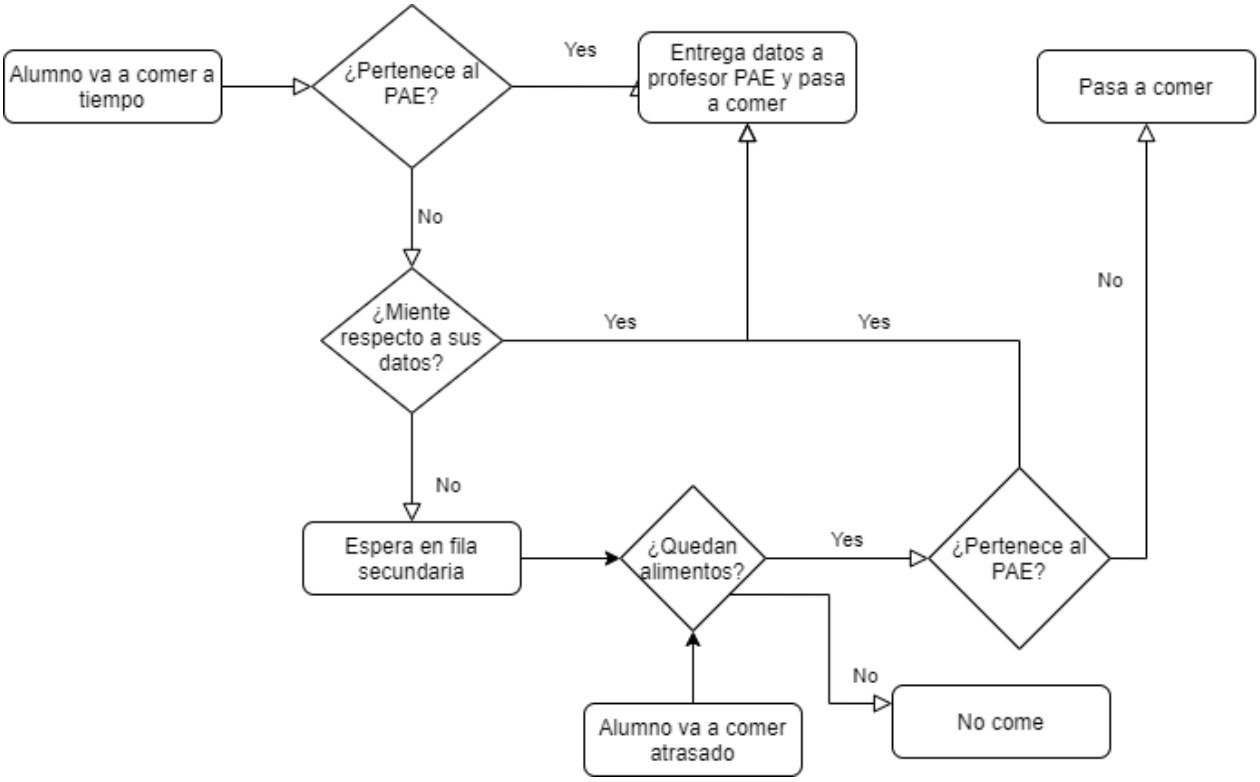


Figura 2.1: Diagrama de flujo del proceso actual de alimentación, de parte de un alumno.

- Juan tampoco pertenece al PAE, pero en lugar de usar un nombre falso, va a esperar en una segunda fila disponible. Dicha segunda fila se hace para los estudiantes que no poseen el beneficio, y se les deja pasar a pedir una ración 10 minutos después de finalizado el horario de almuerzo oficial para los beneficiarios del PAE.
- Martín, a pesar de poseer el beneficio, no va usualmente a almorzar, pues lo hace en su casa. Esta vez, quería almorzar en el colegio, por lo que fue al casino. El profesor PAE no lo dejó pasar, pues ya había pasado alguien con su mismo nombre, por lo que se quedó esperando junto a Juan en la segunda fila.
- Diego sí pertenece al PAE, pero llega media hora tarde, por lo que ya no hay raciones disponibles y se queda sin almuerzo.
- Carlos es un estudiante que va todos los días a comer no solo almuerzo, también todas las raciones disponibles, dado que en su casa no hay suficientes recursos. A pesar de ello, no es parte del PAE, por lo que el profesor PAE, que sabe de su situación, lo hace pasar inmediatamente, sin registrar su asistencia.

Como podemos ver en este simple ejercicio, existen fuentes de irregularidades de parte de los alumnos, de los profesores PAE, del personal a cargo de revisar las raciones, entre otros. Que se podrían resumir en:

- Alumno no perteneciente al PAE toma nombre de compañero, profesor PAE no verifica correctamente, por lo que toma una ración que no le corresponde.

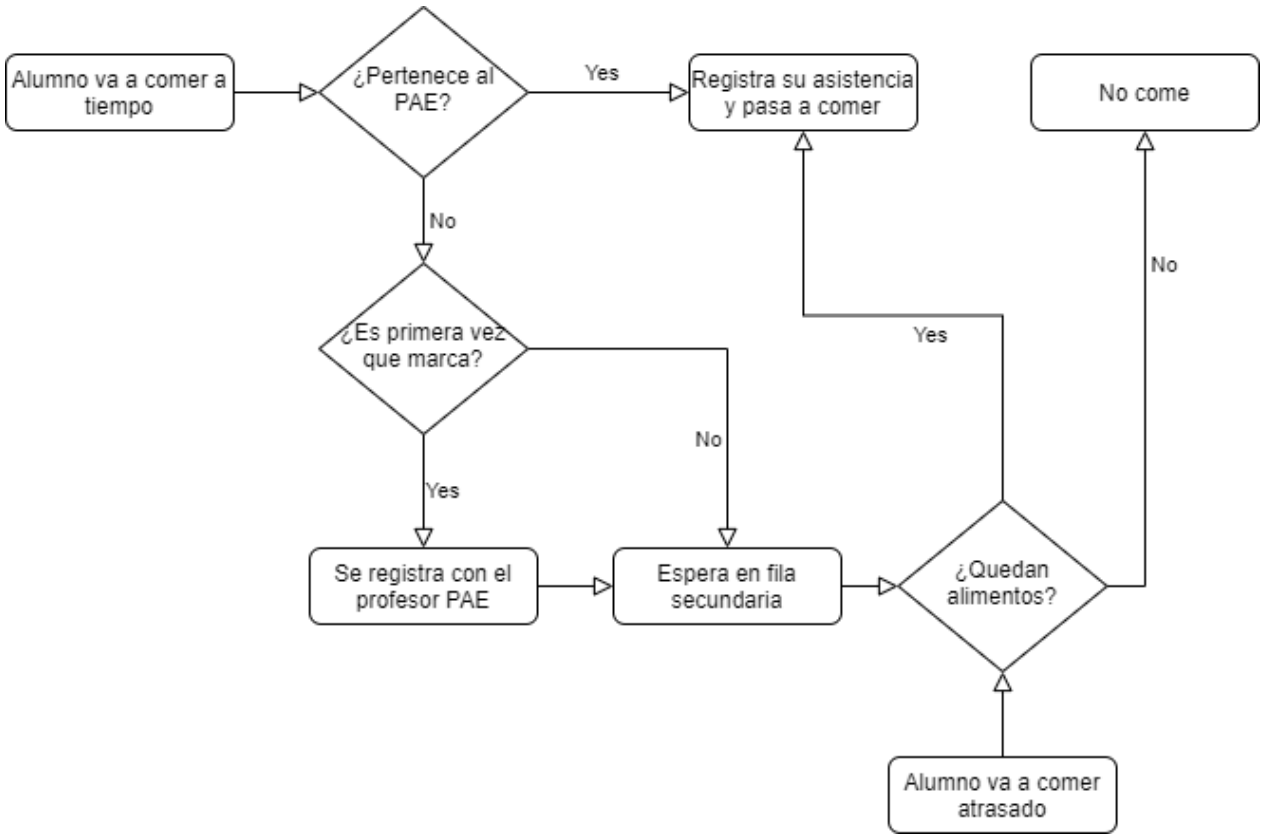


Figura 2.2: Diagrama de flujo esperado del proceso de reparto de raciones, de parte de un alumno.

- Alumno no pertenece al PAE, espera terminado el horario de almuerzo de alumnos beneficiarios del programa, profesor PAE deja pasar al alumno y no queda registro de dicha ración consumida.
- Alumno pertenece al PAE, pero su ración fue tomada por alguien que no pertenecía al programa, dado que profesor PAE se equivocó.
- Alumno pertenece al PAE, llega fuera de horario estipulado y no recibe su ración.
- Alumno no pertenece al PAE, pero necesita del programa. Profesor PAE lo hace pasar y no registra su ración.

Con el proyecto desarrollado, las irregularidades mencionadas son minimizadas o eliminadas de las siguientes maneras:

- En caso que alumno no pertenezca al PAE, de todas maneras queda individualizado con su huella dactilar. El profesor PAE consigna sus datos mediante el mecanismo de registro que tanto JUNAEB, como la empresa de reparto designen para estos casos, y queda registro de que ese alumno obtuvo una ración.
- Otra irregularidad que se elimina es que un alumno obtiene la ración de otro: Dado que las huellas dactilares de cada persona son perennes (duraderas), inmutables (no cambian

de forma), diversiformes (no existen dos huellas dactilares iguales) y originales (al ser impresas, se puede determinar si son verídicas o artificiales), la identificación de cada alumno es inequívoca, por lo que incrementa notoriamente la dificultad de suplantar la identidad de otro alumno.

- Dado que cada profesor PAE no interviene en el proceso de marcaje, no hay posibilidad que este cometa un error respecto a su interacción con cada alumno. Nótese que el profesor PAE sigue estando presente en otros procesos, tales como el registro de alumnos que no hayan marcado nunca, mencionado en el primer punto de esta lista, y la verificación de las óptimas condiciones de funcionamiento del espacio, tales como comportamiento de los alumnos en el espacio designado, u organización de la entrada y salida de los alumnos del espacio designado para recibir y comer las raciones entregadas.
- El conteo de las raciones consumidas es automático. Este conteo está dado por el número de marcas emitidas por los alumnos, por lo que se evita el error de conteo manual.
- La entrega de la información de las raciones está dada por la frecuencia de ejecución de la integración. Así, se fijan plazos de entrega de la información de las raciones.
- Tal como se mencionó en un punto anterior, la cantidad de raciones entregadas a alumnos son las consumidas, por lo que no hay posibilidad de que las empresas de distribución entreguen sus propios números. Es decir, se evita la posibilidad de manipulación de datos de parte de las empresas de distribución de alimentos.

Con ello, el flujo esperado del proceso de entrega de raciones queda descrito en la figura 2.2.

2.4. Revisión bibliográfica

Dentro de la bibliografía que se pudo consultar para construir el marco de referencia que delimita esta aplicación, se tomarán como punto de referencia principal los siguientes textos:

- Acevedo Ferrer, Cristóbal. Secretario General JUNAEB 2017. Aplicación del acta de la supervisión de la gestión operativa y propuesta técnica del PAE para la licitación id 85-10-lp14, 3 2017. [9]

  Para poder conocer las condiciones previas de alimentación de los alumnos pertenecientes al PAE, se requería un documento que sintetizara los aspectos a regular por un supervisor externo. Se usa de referencia este documento para ver los requerimientos técnicos de las labores de un supervisor PAE, encargado de velar por el correcto funcionamiento de los distintos aspectos del programa, tales como espacios, minutas de alimentos, instrumentales, entre otros. Así, se pudo obtener el contexto de los aspectos a chequear en la licitación que se usó como base para la propuesta inicial de esta memoria, y su posterior implementación.
- Villena, Marcelo. DIPRES: Evaluación de Impacto de los Programas de Alimentación de la JUNAEB, del Ministerio de Educación, Diciembre 2013. [25]

Luego de tener la información a revisar acerca del PAE, se requería saber la cantidad de dinero destinada al programa. Así, se puede obtener una medida de utilidad de la aplicación desarrollada respecto a la capacidad de generar con los datos mostrados, propuestas que permitan destinar de mejor manera los recursos. Este informe es revisado para ver las cifras declaradas en cuanto a cantidad de beneficiarios, montos destinados, caracterización de los estudiantes y muchos datos más.

Citando una de las principales conclusiones del documento, al final de la página 336 se menciona que *"No existe claridad sobre los criterios utilizados ni tampoco sobre la cantidad de raciones que se envían a los establecimientos una vez realizada la priorización individual de los estudiantes"*, por lo que es clave el poder mejorar la administración de los recursos.

Cabe destacar que al momento de realizar esta memoria, la única Evaluación de Impacto (EI) similar corresponde al año 1997 [20]. Dicha información se obtuvo desde la sección *Evaluación y Control de Gestión / Evaluación y Revisión del Gasto / Evaluación de Programas e Instituciones* de la plataforma de la DIPRES.

- JUNAEB, Unidad de licitaciones: Suministro de raciones alimenticias, Junio 2018. [14]

  Con toda la información mencionada anteriormente, se estudió la licitación realizada a las empresas de alimentación. De esta ficha de licitación se obtienen los requerimientos generales hacia las empresas de alimentación, incluida la necesidad de implementar un sistema de registro biométrico para los alumnos. Con ello, se vio la oportunidad de desarrollar un conjunto de componentes que permitan implementar este sistema de registro, el cual culmina en este proyecto de memoria.

- Tohá, Jaime. Oficio Fiscalización Respuesta Licitación Junaeb 85-27-LR18, Marzo 2019. [24]

  A medida que se avanzaba en el desarrollo de las aplicaciones, hubo aspectos que hicieron sospechar al memorista que podría resultar en dificultades para continuar el proceso. Debido a ello, se buscó más documentación respecto al proceso de licitación, y se encontró el oficio mencionado. En este oficio se entregan detalles como contrato directo, e incumplimiento de declarar entidades relacionadas. Esto permite declarar la licitación como desierta y realizar otro proceso administrativo contemplado en la legislación vigente.

- Tohá, Jaime. Denegación solicitud acceso a información, Junio 2020.

  Al intentar preparar las pruebas de integración con los identificadores de proveedores provistos por JUNAEB, en mayo del presente año los webservices provistos por JUNAEB (listados en la sección de Anexos) dejaron de funcionar. Ante ello, se realizaron intentos de contacto con la entidad estatal, a lo que se responde que la solicitud de apertura de servicios debe hacerse vía Consejo de Transparencia. Se realizó la solicitud respectiva, la cual fue rechazada dos semanas después.

  En este documento se detalla la denegación de la solicitud de acceso a información que alumno poseía previamente, y con la que se había trabajado en el proceso de Introducción a la Memoria de Título. Los motivos esgrimidos fue que dicha información no existía, y que se debería destinar personal para recopilar y habilitar la información. Ver Anexo B

Capítulo 3

Análisis y Diseño

3.1. Historias de usuario

A continuación se describen las historias de usuario que explican el sistema a realizar en esta memoria.

En primer lugar se define la existencia de 3 tipos de usuario: Estudiante, Profesor PAE y Administrador. Los estudiantes y el profesor PAE son usuarios indirectos del sistema, puesto que ellos alimentan los datos que usará el sistema a mostrar. Por otra parte, los administradores son usuarios directos del sistema. Esta definición se hace dado el alcance de la memoria, puesto que no se diseñan las máquinas que obtienen los registros de huella dactilar, ni la aplicación que usarán los profesores PAE.

Un administrador es un usuario perteneciente a JUNAEB, que puede revisar en la plataforma del sistema propuesto, datos generales y detallados acerca del consumo de raciones de los alumnos. En específico, tiene acceso a un dashboard con los datos de raciones entregadas vs cantidad de servicios esperados en el día, una matriz de raciones entregadas por región y tipo de servicio, y la cantidad de servicios entregados por nivel educacional (a esto se le denomina "estrato"). Además, el administrador puede ver tablas con datos de los profesores PAE y alumnos registrados, así como el detalle de las raciones entregadas.

3.1.1. Estudiantes

Los estudiantes poseen las siguientes historias de usuario relevantes para el registro de datos:

- HE1: Como estudiante perteneciente al PAE, quiero marcar mi servicio a consumir.
- HE2: Como estudiante no perteneciente al PAE, quiero marcar mi servicio a consumir.

3.1.2. Profesor PAE

Los profesores PAE poseen las siguientes historias de usuario relevantes para el registro de datos:

- HP1: Como Profesor PAE, quiero registrar alumnos pertenecientes al PAE en el sistema.
- HP2: Como Profesor PAE, quiero registrar alumnos que no pertenecen al PAE en el sistema.

3.1.3. Administrador

Los administradores poseen las siguientes historias de usuario relevantes para la visualización de datos:

- HA1: Como administrador del sistema, quiero revisar un dashboard con datos de interés.
- HA2: Como administrador del sistema, quiero visualizar de manera detallada los datos de los profesores PAE registrados.
- HA3: Como administrador del sistema, quiero visualizar de manera detallada los datos de los alumnos registrados.
- HA4: Como administrador del sistema, quiero visualizar de manera detallada los datos de las raciones entregadas a los alumnos.

3.2. Análisis del sistema a diseñar

Considerando los objetivos de esta memoria y las historias de usuario, se diseñó una aplicación web, en conjunto a una integración que obtiene datos de los webservice disponibilizados en su momento por JUNAEB (Ver Anexo A), y una API de comunicación con la base de datos.

Para analizar el código, se utilizan las herramientas de métricas de código de Visual Studio 2019, las que entregan información de índice de mantenibilidad, complejidad ciclomática, profundidad de herencia, acoplamiento de clases y líneas de código. [26]

En esta sección se analizará la arquitectura lógica del sistema, el modelo de datos y las aplicaciones web, integración y API que forman parte del sistema.

3.2.1. Arquitectura Lógica

Se creó una aplicación web basada en servicios web y separada en capas, que recibe los datos a mostrar a partir de una base de datos. Dicha BD se alimenta de información de

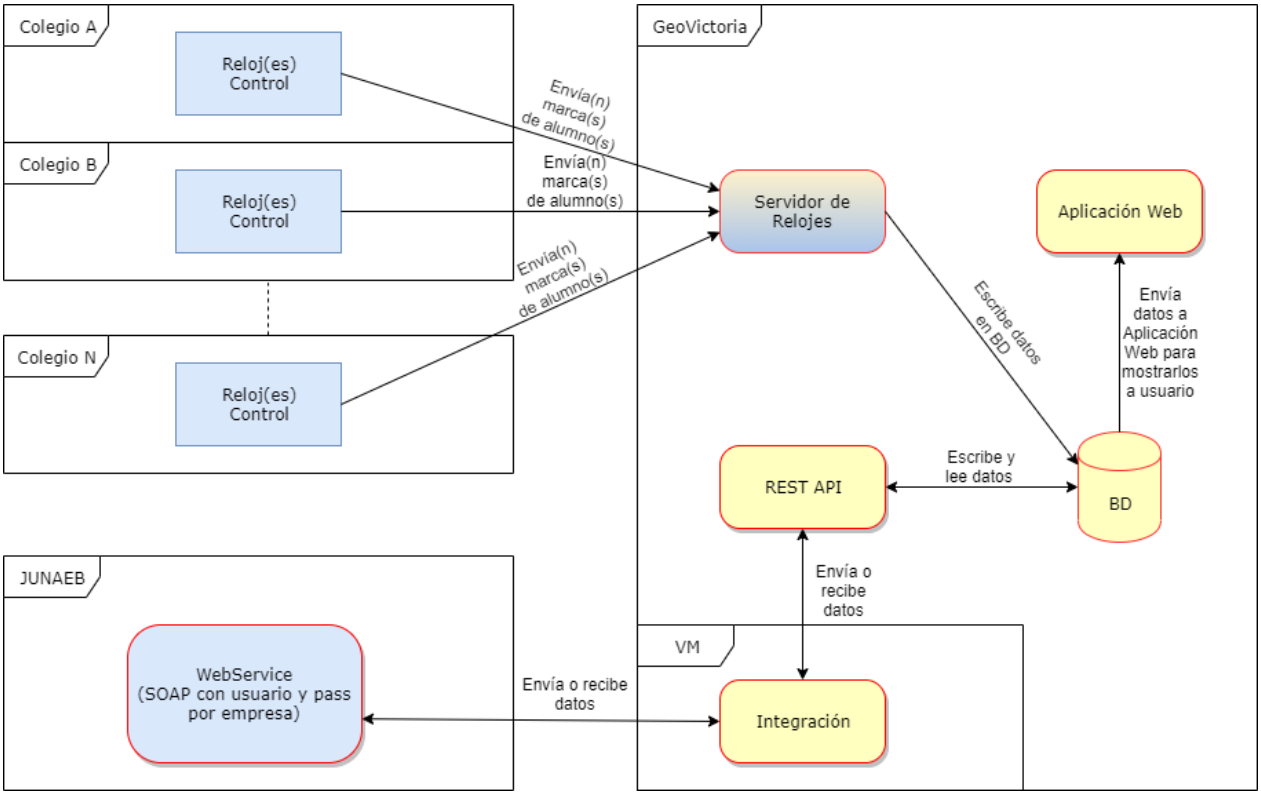


Figura 3.1: Diagrama inicial de arquitectura del sistema

dos fuentes: Primero, obtiene la información de marcaje de los alumnos a través de los reloj control, los que envían la información a una aplicación que traduce la información para guardarla. La segunda fuente de datos de la BD son los webservice de JUNAEB, los cuales usan una integración como medio para comunicarse con los sistemas de GeoVictoria.

En la Figura 3.1 se presenta un diagrama simplificado de arquitectura. Los elementos de color amarillo son los a implementar por el alumno, mientras que los azules son construidos o implementados por un ente externo. El Servidor de relojes es un caso especial, pues parte del código proviene del fabricante de los reloj control (el descifrado de los mensajes de los relojes), y otra es de creación de GeoVictoria. Los elementos de bordes azules rectos son hardware, mientras que los de bordes redondos rojo son software. La base de datos es una instancia de SQL Server provista por Azure, por lo que no se toma en cuenta como arquitectura física.

Se procede a explicar en detalle cada uno de los elementos del diagrama:

Para la aplicación web, se desarrolló como un proyecto de aplicación de consola en C#, con el framework .NET Core 3.1. Esta aplicación tiene un diseño de un único componente que tiene todo lo necesario para funcionar: una aplicación con arquitectura tradicional MVC que abarca elementos base para mostrar al usuario en cuanto esté operativo: Database Access Object (DAO) que mapean los objetos de la base de datos, procesamiento en una Business-Layer de dichos objetos mapeados y la capa de Vista (View) para mostrar al usuario. Se propone más adelante presentar la aplicación web como una aplicación basada en microservicios, comunicados entre sí vía API interna, para delegar responsabilidades a cada parte. De esta manera, con un proyecto web se muestra a los usuarios del sistema los datos que

necesitan visualizar.

Para el sistema de almacenamiento de datos, dado que se acordó con JUNAEB que el proceso en que se trabajaría con las empresas distribuidoras de alimentos sería gradual, se conversó con Juan Francisco Duhart y Jorge Bossa, respectivos subgerente y gerente de tecnología de GeoVictoria, la mejor manera de abordar la arquitectura que recibirá los datos de marcas. Se llegó en conjunto a que, en lugar de empezar inmediatamente con un servicio HDInsight de Azure que contenga Apache Hadoop, se empezara con una base de datos relacional SQL Server para objetivo de esta memoria. Esto, debido a que por ahora, se trabajará con un orden de 10.000 alumnos, mientras que la propuesta con el servicio de Apache Hadoop fue pensada para un orden de más de 3.000.000 de alumnos alimentándose diariamente [25](tabla 9, página 32). Además, el costo de tener una máquina "D13v2" con 8 CPUs y 56 GB de RAM está por sobre los $545USD, mientras que una base de datos serverless cuesta $0.115USD/GB al mes más USD$0.0001450/virtualCore-segundo (costo estimado por transacción), por lo que resulta más económico en el plazo que abarca esta memoria. Más adelante, como se seguirá trabajando en el proyecto con JUNAEB y a medida que el sistema se masifique, se migrarán los datos a un Azure Storage Table cuando la cantidad de alumnos marcando incremente, para finalmente llegar al modelo de almacenamiento de marcas propuesto inicialmente.

Respecto al sistema que obtendrá los datos de cuando marcan los alumnos para retirar su ración, se mantendrá el sistema elegido de los reloj control. Cada reloj control es un elemento de hardware construido por una empresa externa, conectado a internet mediante 3G para evitar la necesidad que los colegios tengan conexiones físicas de internet. Como se menciona en una sección anterior, el reloj cifra las marcas con un algoritmo de hashing y luego se envía. De esta manera, se cumple el requerimiento en las bases de licitación de JUNAEB que sea un sistema biométrico seguro.

El o los reloj control de cada colegio necesitan una aplicación que se encargue de traducir las marcas que dejan los usuarios en datos que puedan alimentar la base de datos y eventualmente llegar a JUNAEB. Para ello, dichos relojes se comunican con GeoVictoria mediante un ADMS (Automatic Data Master Server). El ADMS o servidor de relojes (mismo concepto) funciona igual que una API, interpretando los datos que envían los relojes y transformándolos a DTOs (Data Transfer Objects). Una vez construido el objeto DTO, se escribe en la base de datos para ser usados por la aplicación web y la integración.

De parte de JUNAEB, existe un Web Service SOAP, que cuenta con información separada por empresa y cuyos servicios son descritos en el Anexo A. Este Web Service se debe comunicar mediante una Integración: un archivo ejecutable como tarea programada que verifica el estado de los datos de parte del Web Service de JUNAEB, observa los cambios y de existir, los comunica a GeoVictoria. Para dicha integración, los documentos fueron recibidos el día 19 de diciembre del año 2019 por Francisco Queupil, Jefe de Proyectos en GeoVictoria y encargado de ser el nexo entre la empresa y JUNAEB. Este componente Integración se comunica con la base de datos mediante una REST API dedicada a ello.

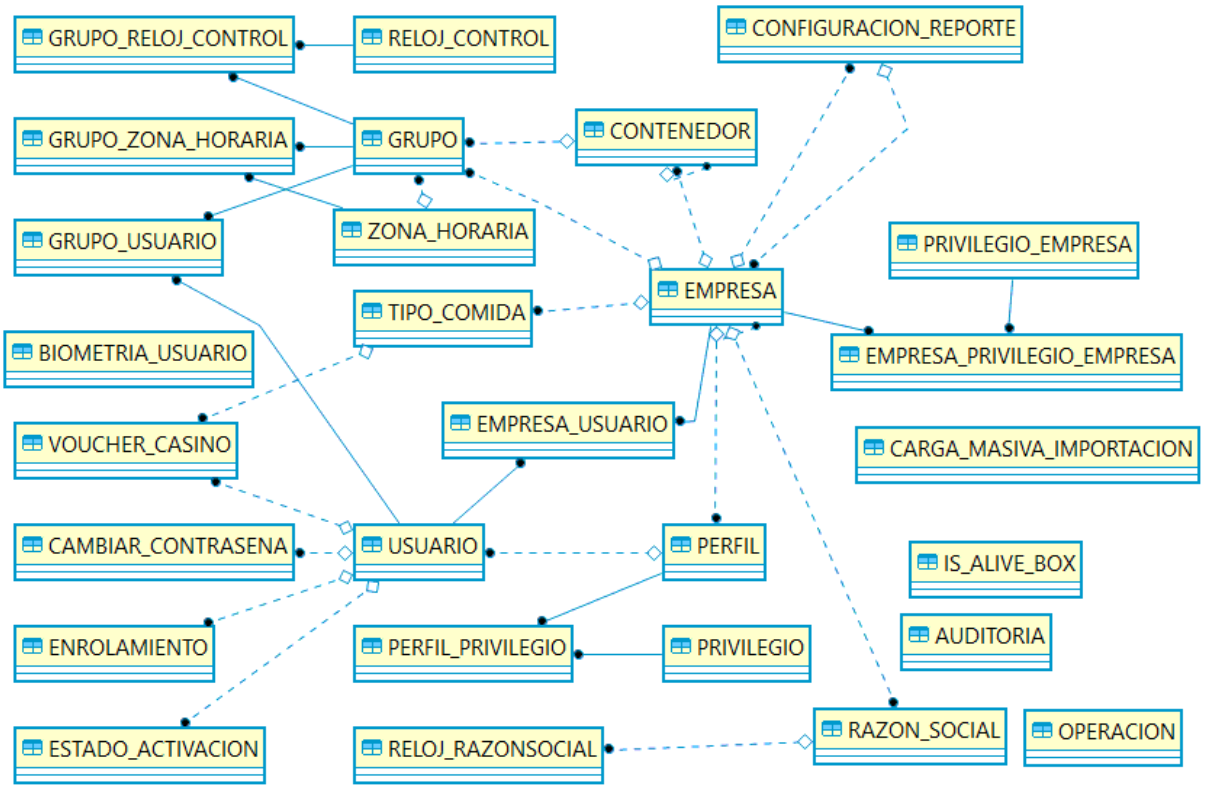


Figura 3.2: Diagrama entidad-relación que muestra el diseño a futuro de la base de datos del sistema a desarrollar.

3.3. Modelo Entidad-Relación de la Base de Datos

En la Figura 3.2 se presenta un diagrama entidad-relación simplificado de la base de datos implementada.

La notación del diagrama está bajo el estándar IDEF1X, por lo que:

- Las líneas sólidas indican relaciones identificadoras (las llaves del padre forman parte de la llave de la entidad hija), mientras que las líneas punteadas que identifican entidades no-identificadoras (las llaves del padre forman llaves foráneas de la entidad hija).
- Las líneas sólidas unidas hacia un punto oscuro indican una relación de uno a cero, uno o varios (Zero, One or Many).
- Las líneas punteadas unidas desde un rombo hacia un punto oscuro indican una relación opcional a cero, uno o varios.

Las tablas más importantes son USUARIO, EMPRESA y GRUPO, debido a que poseen datos de los usuarios del sistema, las empresas de alimentación inscritas, y como se subdividen los colegios. Cada USUARIO pertenece a un GRUPO que la EMPRESA de alimentación organiza. El detalle de estas entidades y relaciones en la base de datos se encuentra en la Figura A.1 del Anexo de este documento.

Al enrolar un usuario, se guarda en ENROLAMIENTO datos importantes del proceso, como una relación entre el identificador que provee JUNAEB al estudiante PAE y un hash del dedo enrolado, de esa manera se puede consultar cuando un usuario se unió al sistema. Cada marca o ticket del usuario se guarda en VOUCHER_CASINO, cuyo TIPO_COMIDA decide la EMPRESA (normalmente se divide en desayuno, almuerzo y once, pero pueden haber más categorías, tales como colación o cena). Los estudiantes PAE además tienen un ESTADO_ACTIVACION, que indica si puede continuar marcando o no. Por otro lado, los usuarios de la plataforma (administradores) deben ser capaces de cambiar sus contraseñas cuando sea implementado el sistema de sesiones, por lo que se guarda el historial de cambios. El detalle de esta relación en la base de datos se encuentra en la Figura A.2 (ver Anexo).

Cada EMPRESA presenta una RAZON_SOCIAL, bajo la cual se identifican legalmente. Dicha razón social tiene relojes control asociados, así como una lista de PRIVILEGIO_EMPRESA para acceder a reportes personalizados, cuyos parámetros se ven en la tabla CONFIGURACION_REPORTE. Un ejemplo de estos reportes personalizados puede ser un Excel que contenga información geográfica por comuna y cuantos beneficiarios hay. Otro ejemplo puede ser un resumen de raciones estimadas a entregar y raciones realmente entregadas, organizadas por GRUPO. El detalle de esta relación en la base de datos se encuentra en la Figura A.3 (ver Anexo).

El sistema que se espera realizar más allá de los alcances de esta memoria permitirá a las empresas de reparto de alimentos organizar a los alumnos en base a GRUPOs y CONTENEDORes que estime convenientes; si se piensa como un árbol, los contenedores son nodos intermedios, mientras que los grupos son las hojas. Por ejemplo, definiendo cada colegio como un grupo, se puede armar contenedores por comuna, en las que cada uno contenga todos los colegios perteneciente a dicha unidad territorial, luego tener contenedores que correspondan a las provincias, que contengan sus comunas correspondientes. Otra empresa puede agrupar los colegios en contenedores en base a si son municipales, particulares subvencionados o particulares. En otros casos, el nivel mínimo que definen las empresas de alimentación son por nivel educacional, por lo que colegio no sería el nivel mínimo. Cada grupo tiene asociada una ZONA_HORARIA (pensando en los casos borde como Isla de Pascua o Magallanes), así como RELOJ_CONTROL. Periódicamente se chequea mediante un handshake que los relojes control sigan funcionando. El detalle de esta relación en la base de datos se encuentra en la Figura A.4 (ver Anexo).

Dado que se necesita que el sistema sea configurable, cada empresa de reparto de alimentos posee un PERFIL, este es distinto para usuarios que tenga bajo su cargo como estudiantes, personal de la empresa, profesores PAE o administradores. Dichos perfiles otorgan uno o varios PRIVILEGIOs a los USUARIOs, que definen que pueden ver o no en la página. Por ejemplo, una empresa pequeña no contratará todos los servicios disponibles en la plataforma, pues solo le interesa poder visualizar los datos de manera simple y que GeoVictoria se encargue de administrar su configuración y qué puede visualizar, como razón social, dirección de facturación, tipos de reportes, entre otros. Por ello, puede tener un usuario que revise datos con un PERFIL que corresponde a dicho cargo, es decir, un PERFIL “Revisor de datos”. A ese PERFIL se le otorgan los PRIVILEGIOs “Visualizar alumnos bajo empresa”, “Visualizar tickets”, “Visualizar profesores PAE”. Por otro lado, un administrador de una empresa de alimentación que tenga una gran cantidad de colegios licitados puede tener unos cuantos usuarios con PERFIL “Administrador”, al que se le otorgan PRIVILEGIOs de “Editar de

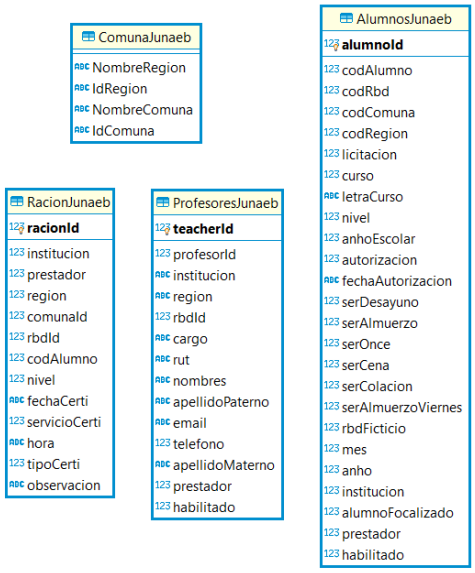


Figura 3.3: Diagrama entidad-relación que muestra las tablas actualmente llenadas en el proyecto. En ellas, se guardan los datos de alumnos, profesores PAE y raciones, tal cual las necesita JUNAEB. Además, se encuentra la tabla que contiene la información de las comunas, con su identificador para conversión.

dirección de facturación", "Visualizar grupos", "Visualizar estadísticas detalladas". De esta manera, se le da un grado de personalización a cada empresa para que pueda decidir qué pueden hacer o no sus usuarios y les otorga la posibilidad de configurar dicha personalización en base a perfiles de usuario. El detalle de esta relación se encuentra en la Figura A.5 del Anexo.

Para guardar los registros biométricos de los estudiantes PAE, se guarda su hash biométrico en la tabla BIOMETRIA_USUARIO, en la que se compara el campo HASH_BIOMETRIA con la marca del usuario correspondiente, y verifica con ello si el usuario es efectivamente quien marca. JUNAEB requiere realizar AUDITORIAs respecto a la utilización de las marcas y los usuarios registrados, por lo que se guarda quien y cuando se realizó para tener un histórico. La estrategia de como poblar esta tabla todavía está por definir y en reuniones futuras se conversará con JUNAEB. Por último, si un usuario quiere cambiar su contraseña, puede modificarla, quedando guardado el código de verificación del cambio de contraseña. El detalle de esta relación en la base de datos se encuentra en la Figura A.6 del Anexo.

Cabe destacar que todo este proceso fue creado para que en el futuro, más allá de los alcances de esta memoria, se puedan extender las funcionalidades a usar por la aplicación PlataformaJunaeb, a detallar en el siguiente capítulo.

Capítulo 4

Implementación

Considerando el diseño propuesto en el Capítulo 3, en este capítulo se describe la implementación del sistema diseñado. Este capítulo se separa por repositorios:

- **IntegracionJunaeb** se encarga de obtener y enviar los datos a los webservice de JUNAEB. Esto, para que la responsabilidad de obtener los datos no recaiga en la ejecución de la aplicación web. Los datos de raciones deben enviarse en cuanto estén disponibles, y los datos de alumnos y profesores recibirse independiente para poder estar en el mismo contexto que JUNAEB.
- **JunaebRawData** es una API que se encarga de ser el puente entre la integración y la base de datos. Se decidió así pues la integración no puede tener la responsabilidad de asegurar una conexión con la base de datos, para mantener los tipos de datos lo más limpio al momento de transmitir.
- **PlataformaJunaeb** es la aplicación web de cara al usuario. Esta plataforma recoge los datos obtenidos por la integración para que los usuarios puedan ver las raciones entregadas, los alumnos y profesores PAE.

Cabe destacar que se eligió este proceso de diseño para desacoplar responsabilidades, y pensando en que podrían ocurrir eventualidades. `IntegracionJunaeb` y `JunaebRawData` están preparados para soportar contigencias en cuanto al acceso de los datos ante la eventual denegación de accesos a los webservices de JUNAEB. Esto, dado que al distribuir las responsabilidades de transacciones de datos a la aplicación de integración y el acceso a la base de datos a la API, se encapsulan los cambios futuros en dos componentes en sectores específicos (los Data Transfer Objects, o DTOs).

Esto incrementa la mantenibilidad del sistema al poder realizar correcciones en puntos específicos, y le permite evolucionar en caso de que, al restablecerse los webservices, JUNAEB incluya algún nuevo tipo de estructura de datos y con ello, se deban agregar nuevas funcionalidades. Todo esto se logra gracias a que se pensó en un patrón mediador [1] al diseñar todos los componentes.

El patrón mediador es un patrón de diseño de tipo comportamiento, que permite reducir la dependencia entre objetos. El patrón restringe la comunicación entre los objetos, y los

fuerza a interactuar mediante un objeto "mediador". De esta manera, en lugar de que cada objeto llame a otro directamente, el mediador es el encargado de llamar a otro objeto. Esto se puede ver en que para pedir o recibir objetos de la base de datos, en los proyectos se crearon objetos `Filtro`, que actúan de canal para realizar las peticiones desde la aplicación de integración hacia la API `JunaebRawData`.

De esta manera, cuando a futuro se pueda retomar los diálogos con JUNAEB, los componentes pueden volver a integrarse con los webservices de manera simple, limpiando la base de datos y usando los datos provistos por ellos para cargar los datos reales, implicando un bajo esfuerzo de actualización.

Para cada repositorio, se describirán su funcionalidad, la implementación de cada módulo, problemas en el proceso y cómo se resolvieron.

4.1. Herramientas ocupadas

En esta sección se describirán las herramientas ocupadas para implementar los distintas aplicaciones.

4.1.1. .NET Framework

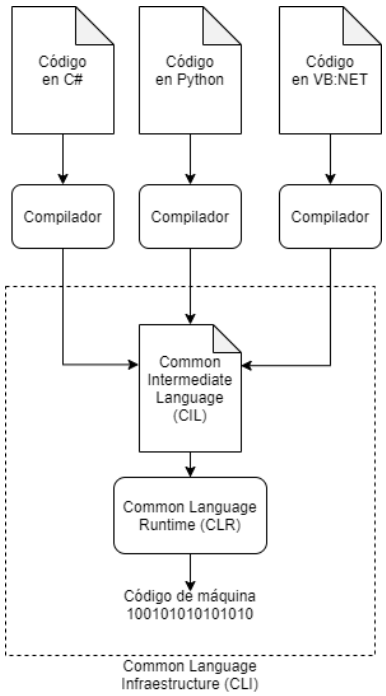


Figura 4.1: Diagrama que muestra a grandes rasgos cómo funciona .NET Framework.

`.NET Framework` [4] es un framework creado por Microsoft que facilita el desarrollo de aplicaciones en Windows. Este framework consta de dos partes importantes: la Common

Language Interface, o CLI, que permite que códigos en distintos lenguajes se comuniquen entre sí, y la Common Language Runtime, o CLR, que es una máquina virtual que se encarga de revisar el manejo de memoria del programa, el manejo de excepciones, garbage collection, ver errores de tipo, y hacer compilación JIT (Just-In-Time) al lenguaje intermedio de éste, llamado Common Intermediate Language o CIL.

4.1.2. Kestrel

`Kestrel` [7] es un webserver de código abierto, multiplataforma y usado por defecto para las aplicaciones .NET Core. Es decir, que cualquier plataforma que soporte .NET Core también soportará `Kestrel`. El problema de `Kestrel` es que no es un webserver con muchas funcionalidades de front-end, por lo que funciona en este caso como reverse proxy server, siendo una puerta de entrada a otros webservers, como `Apache`, `NGINX` o `IIS`, evitando tener que configurarlos.

4.1.3. Razor

Explicando `Razor` [5] en simple, es una sintaxis para templates, basada en C#. Esta sintaxis permite escribir código HTML usando un enfoque orientado más hacia código con mínimos cambios. Así, se simplifica el crear bloques de código dentro del template sin tener que explicitar la delimitación de este bloque.

4.1.4. Microsoft Power BI

`Microsoft Power BI` [8] es un conjunto de software y servicios que permiten entregar datos de una forma visualmente atractiva e interactiva. Consta de tres servicios que trabajan en conjunto:

- `Power BI Desktop`, que permite crear los reportes visuales de manera local.
- `Power BI Service`, que almacena y permite compartir los reportes generados en `Power BI Desktop` para integrarlos en una aplicación web.
- `Power BI apps`, que son las aplicaciones donde se ejecutará el reporte compartido.

Para obtener los datos, Power BI puede descargar los datos de antemano, o usar el método `DirectQuery`. `DirectQuery` es un tipo de conexión en que los datos no son cargados directamente en el modelo del reporte, sino que son pedidos, como su nombre indica, como una consulta directa a la base de datos cada vez que se carga el reporte. Ello implica que no usa memoria local adicional, dado que no hay copia de los datos.

4.2. IntegracionJunaeb

4.2.1. Aspectos Generales

La integración es una aplicación de consola desarrollada en C# con el framework `.NET Framework 4.6.2`. La aplicación cuenta con un controlador principal, encargado de realizar las tres acciones principales: la obtención de datos de alumnos, la obtención de datos de profesores, y el envío de datos de raciones. Esto, se hace llamando a las respectivas capas de negocios, declaradas en Business las que se encargan de procesar los datos y verificar qué datos agregar, editar o eliminar. Las clases que abordan la lógica de negocios, llamadas "business", a su vez, están hechas con base en sus respectivas interfaces en IBusiness. Por último, tanto para la interacción con la API `JunaebRawData` como los webservice de JUNAEB, se cuenta con una capa DAO con su respectiva interfaz IDAO. Por último, la carpeta Comunes cuenta con los modelos de datos para hacer la transformación de datos respectiva dentro de la carpeta ApiModels, así como archivos Helpers con métodos que ayudan a realizar un log de la integración en el día correspondiente, y un comparador de objetos.

Cuando se ejecuta el proyecto de integración, primero parte con un mensaje de log "*Inicio de la ejecución del programa*", y revisa si fue llamada sin argumentos específicos para ejecutar todos los módulos, o con alguno de los siguientes argumentos para ejecutar solamente dicho módulo:

- `getalumnosjunaeb`, que indica que solo debe ejecutar el módulo de obtención de alumnos para alimentar la base de datos.
- `getprofesoresjunaeb`, que indica que solo debe ejecutar el módulo de obtención de profesores PAE para alimentar la base de datos.
- `sendraciones`, que indica que solo debe ejecutar el módulo de envío de raciones al Web-Service de JUNAEB dispuesto a aquello.

Luego, se ejecutan los módulos respectivos, contenidos en su clase Business, los que se detallarán en las siguientes subsecciones.

4.2.2. Obtención de datos de alumnos

El controlador principal llama al método `GetStudents()` de StudentBusiness. Este inicia el log con las frases "*INICIO SINCRONIZACION ALUMNOS*" e "`INICIO OBTENCION ALUMNOS`", para luego ejecutar los métodos `GetStudents()` de `JunaebStudentDAO` (que obtiene los datos de alumnos de JUNAEB) y `StudentDAO` (que obtiene los datos de alumnos de la base de datos propios).

Los datos a traer de los alumnos corresponden a los descritos en el Anexo A, en el documento *Implementación Servicio de Obtención de Matrículas RBD*, en el apartado *Descripción y Tipo de Campos* (página 69).

```
0 references
public class Program
{
    0 references
    static void Main(string[] args)
    {
        try
        {
            GeneralHelper.Log("-------------------------------------");
            GeneralHelper.Log("Inicio de la ejecución del programa.");
            List<string> operaciones = new List<string>();
            if (args.Length > 0)
            {
                foreach (string s in args)
                {
                    if (s.ToLower() == OperacionConst.GetAlumnosJunaeb)
                    {
                        if (!operaciones.Contains(OperacionConst.GetAlumnosJunaeb))
                            operaciones.Add(OperacionConst.GetAlumnosJunaeb);
                    }
                    else if (s.ToLower() == OperacionConst.GetProfesoresJunaeb)
                    {
                        if (!operaciones.Contains(OperacionConst.GetProfesoresJunaeb))
                            operaciones.Add(OperacionConst.GetProfesoresJunaeb);
                    }
                    else if (s.ToLower() == OperacionConst.SendRaciones)
                    {
                        if (!operaciones.Contains(OperacionConst.SendRaciones))
                            operaciones.Add(OperacionConst.SendRaciones);
                    }
                }
            }
            else
            {
                operaciones.Add(OperacionConst.GetAlumnosJunaeb);
                operaciones.Add(OperacionConst.GetProfesoresJunaeb);
                //operaciones.Add(OperacionConst.SendRaciones);
            }
            foreach (string operacion in operaciones)
            {
                EjecutarOperacion(operacion);
            }
            GeneralHelper.Log("Fin de la ejecución del programa.");
        }
        catch (Exception ex)
        {
            GeneralHelper.Log(ex.Message);
            GeneralHelper.Log(ex.StackTrace);
            GeneralHelper.Log("ERROR EN LA INTEGRACION");
        }
    }
```

Figura 4.2: Método principal de la integración.

En el caso de JunaebStudentDAO, se realiza una request GET a la URL https://servicio bio-test.junaeb.cl/, al endpoint "`RESTfulBiometrico/json/buscarFocalizado/obtenerAutorizados/[Prestador]`", siendo [Prestador] un identificador correspondiente a cada empresa de entrega de alimentos, definido en la raíz del proyecto, en el archivo `App.Config`, donde también se define la URL como `UrlApiJunaeb`. Esta request además posee un usuario y contraseñas definidos en el mismo archivo, como `UsuarioApiJunaeb` y `PassApiJunaeb`, respectivamente. Este endpoint devuelve un JSON que debe ser deserializado en una lista de objetos `AlumnoJunaebAPI`. `AlumnoJunaebAPI` está definido en la carpeta `ApiModels` señalada en la sección 4.4.1.

En el caso de `StudentDAO.GetAllStudents`, se realiza una request POST a la API `JunaebRawData`, cuya URL está por definir en `UrlApiVictoria`, dado que el servicio todavía no está en producción, al endpoint `/Student/GetAllStudents/`. La respuesta es chequeada

```
public static class OperacionConst
{
    public const string GetAlumnosJunaeb = "getalumnosjunaeb";
    public const string GetProfesoresJunaeb = "getprofesoresjunaeb";
    public const string SendRaciones = "sendraciones";
}
```

Figura 4.3: Constantes de ejecución a comparar con los argumentos de inicio para ejecutar módulos de manera individual.

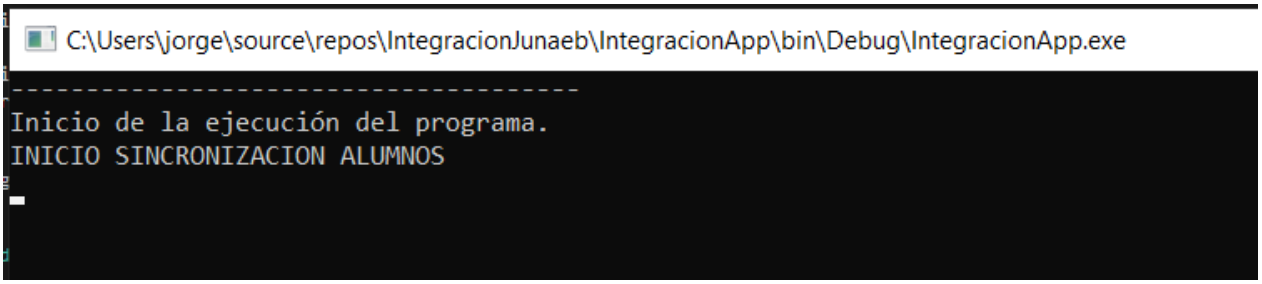


Figura 4.4: Mensajes de log en consola.

si vuelve con un código HTTP 200 (OK). De ser así, este endpoint devuelve un JSON que debe ser deserializado en una lista de objetos `TeacherAPI`. `TeacherAPI` está definido en la carpeta `ApiModels` señalada en la sección 4.4.1. En caso contrario, dispara una Exception con la descripción "Error studentGetAllStudents:" y el contenido de la Response.

Luego de la obtención de datos, se chequea que alumnos se deben agregar, editar o eliminar, comparando ambas listas de alumnos.

Para el chequeo de alumnos a agregar, se llama al método:

`GetStudentsToAdd(List<AlumnoJunaebAPI> alumnosJunaeb, List<StudentAPI> alumnosVictoria)`

que recibe la lista de alumnos de JUNAEB y la lista de alumno de la base de datos propia. En este método, se chequea cada alumno que llega de Junaeb, y si en la lista de alumnos propia no hay un campo de alumno que coincida en el campo `codAlumno`, se realiza una copia de este alumno a un objeto `StudentAPI`, y se le agregan los campos de `habilitado = 1` y `prestador = [Prestador]`, definido anteriormente. Al final, este método devuelve la lista de los alumnos a agregar como una `List<StudentAPI>`.

Para el chequeo de alumnos a eliminar, se llama al método:

`GetStudentsToDelete(List<AlumnoJunaebAPI> alumnosJunaeb, List<StudentAPI> alumnosVictoria)` que recibe la lista de alumnos de JUNAEB y la lista de alumno de la base de datos propia. En este método, se chequea cada alumno que llega de la base de datos propia, y si en la lista de alumnos de JUNAEB **NO** hay un campo de alumno que coincida en el campo `codAlumno`, se le modifica el campo de `habilitado = 0` .Al final, este método devuelve la lista de los alumnos a eliminar como una `List<StudentAPI>`. Nótese que esto es una modificación de un campo solamente, y no una eliminación completa de los datos de la

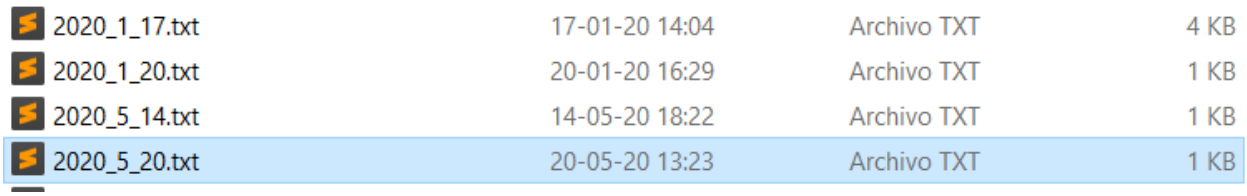


Figura 4.5: Archivos de log directorio.

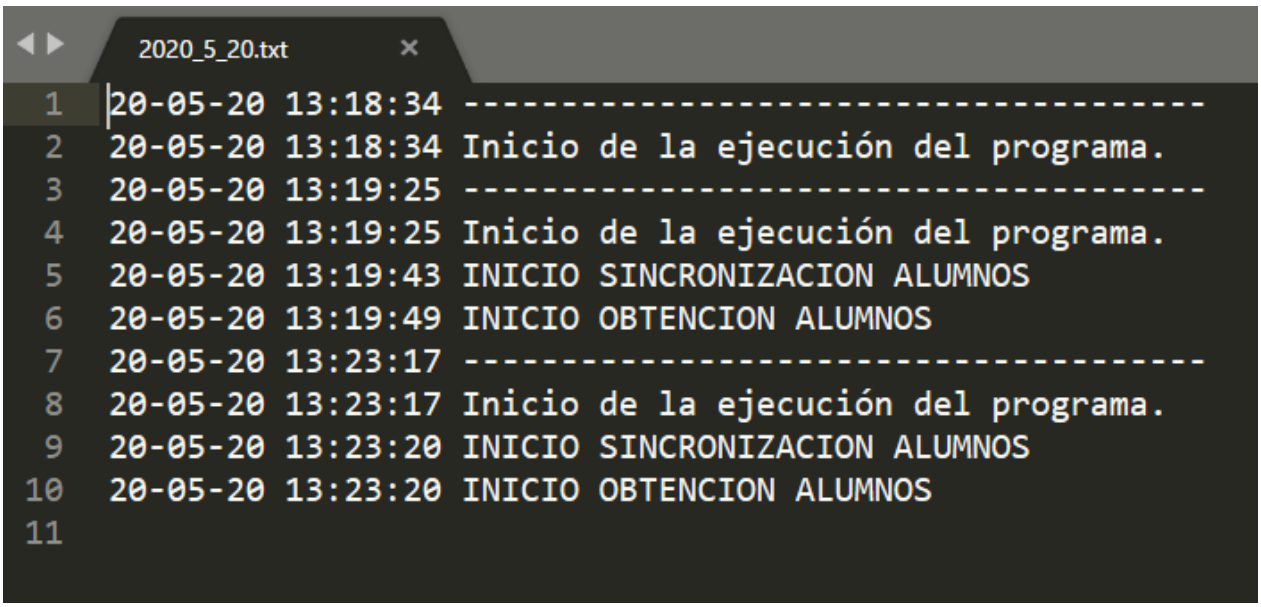


Figura 4.6: Registros de log en el archivo .txt.

base de datos. Esto, por si en un futuro se habilita de nuevo el alumno, se mantengan sus identificadores y la correspondencia con las raciones.

Para el chequeo de alumnos a actualizar, se llama al método `GetStudentsToUpdate(List<AlumnoJunaebAPI> alumnosJunaeb, List<StudentAPI> alumnosVictoria)`, que recibe la lista de alumnos de JUNAEB y la lista de alumno de la base de datos propia. En este método, se chequea cada alumno que llega de la base de datos propia, y si en la lista de alumnos de JUNAEB hay un campo de alumno que coincida en el campo `codAlumno`, se realiza una copia de este alumno a un objeto `StudentAPI`. Luego, se le agregan los campos de `habilitado = 1` y `prestador = [Prestador]`, para finalmente llamar al método `GeneralHelper.GetChangedProperties<StudentAPI>(alumno, alumnoAComparar)`.Al final, este método devuelve la lista de los alumnos a actualizar como una `List<StudentAPI>`.

`GetChangedProperties` hace una comparación profunda entre los dos objetos recibidos, llamados A y B. Esto quiere decir que: Verifica si ambos objetos son distintos a null (en caso contrario dispara un `ArgumentNullException` que dice "*You need to provide 2 non-null objects*"), obtiene el tipo T pasado como Generic, y ve que ambos A y B tengan las mismas propiedades del tipo T, y los mismos valores en dichas propiedades. De no cumplirse alguna de las condiciones, retorna false, en caso contrario, true, ambos como boolean.

Finalmente, se realiza un log que dice "*INICIO ENVIO ALUMNOS A GV*", para llamar a los métodos `AddStudents(agregarAlumnos)`, `UpdateStudents(actualizarAlumnos)`

```
public class StudentBusiness : IStudentBusiness
{
    2 references
    public void GetStudents()
    {
        GeneralHelper.Log("INICIO SINCRONIZACION ALUMNOS");
        IJunaebStudentDAO junaebStudentDAO = new JunaebStudentDAO();
        IStudentDAO studentDAO = new StudentDAO();

        GeneralHelper.Log("INICIO OBTENCION ALUMNOS");
        List<AlumnoJunaebAPI> alumnosJunaeb = junaebStudentDAO.GetStudents();
        List<StudentAPI> alumnosVictoria = studentDAO.GetStudents();

        List<StudentAPI> agregarAlumnos = GetStudentsToAdd(alumnosJunaeb, alumnosVictoria);
        List<StudentAPI> actualizarAlumnos = GetStudentsToUpdate(alumnosJunaeb, alumnosVictoria);
        List<StudentAPI> borrarAlumnos = GetStudentsToDelete(alumnosJunaeb, alumnosVictoria);

        GeneralHelper.Log("INICIO ENVIO ALUMNOS A GV");
        studentDAO.AddStudents(agregarAlumnos);
        studentDAO.UpdateStudents(actualizarAlumnos);
        studentDAO.DeleteStudents(borrarAlumnos);

        GeneralHelper.Log("FIN SINCRONIZACION ALUMNOS");
    }
```

Figura 4.7: Método principal de la capa de negocios del módulo de consultas de alumnos.

```
public List<AlumnoJunaebAPI> GetStudents()
{
    RestClient client = GeneralHelper.GetJunaebClient();
    var request = new RestRequest("RESTfulBiometrico/json/buscarFocalizado/obtenerAutorizados/"
        + ConfigurationManager.AppSettings.Get("Prestador"), Method.GET); // /User: Operation. Always use "POST"
    var response = client.Execute(request);
    var content = response.Content;
    List<AlumnoJunaebAPI> listUser = JsonConvert.DeserializeObject<List<AlumnoJunaebAPI>>(content);
    return listUser;
}
```

Figura 4.8: Captura parcial de clase JunaebStudentDAO.

y `DeleteStudents(borrarAlumnos)` de `StudentDAO`, en ese orden, con las listas respectivas de los métodos anteriores, terminando con un log "*FIN SINCRONIZACION ALUMNOS*".

En los tres métodos, el procedimiento es muy similar: se realiza una request POST a la URL por definir en UrlApiVictoria, a los endpoint "/Student/AddStudents/", "/Student/-DisableStudents/" y "/Student/UpdateStudents/", añadiendo a la request una versión serializada en JSON de las respectivas listas entregadas. La respuesta es chequeada si vuelve con un código HTTP 200 (OK). StudentAPI está definido en la carpeta ApiModels señalada en la sección 4.4.1. En caso contrario, dispara una Exception con la descripción "Error student[Add/Disable/Update]Students:" y el contenido de la Response.

```
public List<StudentAPI> GetStudents()
{
    IRestClient client = GeneralHelper.GetVictoriaClient();
    var request = new RestRequest("/Student/GetAllStudents/", Method.POST)
    {
        RequestFormat = DataFormat.Json
    };
    request.AddJsonBody(Provider);
    var response = client.Execute(request);
    if (response.StatusCode != HttpStatusCode.OK)
    {
        throw new Exception("Error studentGetAllStudents:" + response.Content);
    }
    var content = response.Content; //Get Response
    List<StudentAPI> alumnosJunaeb = JsonConvert.DeserializeObject<List<StudentAPI>>(content);
    return alumnosJunaeb;
}
```

Figura 4.9: Captura de método GetStudents de StudentDAO.

```
public static bool GetChangedProperties<T>(object A, object B)
{
    if (A != null && B != null)
    {
        Type type = typeof(T);
        foreach (PropertyInfo pi in type.GetProperties(BindingFlags.Public | BindingFlags.Instance))
        {
            object selfValue = type.GetProperty(pi.Name).GetValue(A, null);
            object toValue = type.GetProperty(pi.Name).GetValue(B, null);

            if (selfValue != toValue && (selfValue == null || !selfValue.Equals(toValue)))
            {
                return false;
            }
        }
        return true;
    }
    else
    {
        throw new ArgumentNullException("You need to provide 2 non-null objects");
    }
```

Figura 4.10: Método GetChangedProperties, que compara igualdad entre dos instancias de objetos.

4.2.3. Obtención de datos de profesores PAE

Para el proceso de obtención de datos de profesores PAE, el proceso es similar al proceso de obtención de datos de alumnos, por lo que solo se describirá de manera general el proceso.

El controlador principal llama al método GetTeachers() de TeacherBusiness. Este inicia el log con las frases “INICIO SINCRONIZACION PROFESORES” e “INICIO OBTENCION PROFESORES”, para luego ejecutar los métodos GetTeachers() de JunaebTeacherDAO y TeacherDAO.

Los datos a traer de los profesores PAE corresponden a los descritos en el Anexo A, en el documento *Implementación Servicio de Profesores Responsables*, en su apartado *Descripción y Tipo de Campos* (página 70).

En el caso de JunaebTeacherDAO, se realiza una request GET a la URL https://serviciobio-test.junaeb.cl/, al endpoint “RESTfulBiometrico/json/buscarFocalizado/obtenerPorfesore-

```
public void GetTeachers()
{
    GeneralHelper.Log("INICIO SINCRONIZACION PROFESORES");
    IJunaebTeacherDAO junaebTeacherDAO = new JunaebTeacherDAO();
    ITeacherDAO teacherDAO = new TeacherDAO();

    GeneralHelper.Log("INICIO OBTENCION PROFESORES");
    List<ProfesorJunaebAPI> profesoresJunaeb = junaebTeacherDAO.GetTeachers();
    List<TeacherAPI> profesoresVictoria = teacherDAO.GetTeachers();

    List<TeacherAPI> agregarProfesores = GetTeachersToAdd(profesoresJunaeb, profesoresVictoria);
    List<TeacherAPI> actualizarProfesores = GetTeachersToUpdate(profesoresJunaeb, profesoresVictoria);
    List<TeacherAPI> borrarProfesores = GetTeachersToDelete(profesoresJunaeb, profesoresVictoria);

    GeneralHelper.Log("INICIO ENVIO PROFESORES A GV");
    teacherDAO.AddTeachers(agregarProfesores);
    teacherDAO.UpdateTeachers(actualizarProfesores);
    teacherDAO.DeleteTeachers(borrarProfesores);

    GeneralHelper.Log("FIN SINCRONIZACION PROFESORES");
}
```

Figura 4.11: Método principal de la capa de negocios del módulo de consultas de profesores PAE.

s/getProfesorResponsableBio/[Prestador]". Este endpoint devuelve un JSON que debe ser deserializado en una lista de objetos ProfesorJunaebAPI, definido en la carpeta ApiModels señalada en la sección 4.4.1.

```
public List<ProfesorJunaebAPI> GetTeachers()
{
    RestClient client = GeneralHelper.GetJunaebClient();
    var request = new RestRequest("RESTfulBiometrico/json/buscarFocalizado/obtenerPorfesores/getProfesorResponsableBio/"
        + ConfigurationManager.AppSettings.Get("Prestador"), Method.GET); // /User: Operation. Always use "POST"
    var response = client.Execute(request);
    var content = response.Content;
    List<ProfesorJunaebAPI> listUser = JsonConvert.DeserializeObject<List<ProfesorJunaebAPI>>(content);
    return listUser;
}
```

Figura 4.12: Método GetTeachers de la clase JunaebTeacherDAO.

En el caso de `TeacherDAO.GetAllTeachers`, se realiza una request POST a la API `JunaebRawData`, al endpoint `/Student/GetAllTeachers/`. La respuesta es chequeada si vuelve con un código HTTP 200 (OK). De ser así, este endpoint devuelve un JSON que debe ser deserializado en una lista de objetos `TeacherAPI`. En caso contrario, dispara una Exception con la descripción "`Error teacherGetAllTeachers:`" y el contenido de la Response.

Luego de la obtención de datos, se chequea qué profesores se deben agregar, editar o eliminar, comparando ambas listas de profesores.

Para el chequeo de profesores a agregar, se llama al método `GetTeachersToAdd`; para el chequeo de profesores a eliminar, se llama al método `GetTeachersToDelete` y para el chequeo de profesores a actualizar, se llama al método `GetTeachersToUpdate`. En los tres, se recibe la lista de profesores de JUNAEB y la lista de profesores de la base de datos propia. Al final, estos métodos devuelven una lista de los profesores a agregar, actualizar o eliminar como una

List<TeacherAPI>.

```
2 references
public void AddTeachers(List<TeacherAPI> profesores)
{
    IRestClient client = GeneralHelper.GetVictoriaClient();
    var request = new RestRequest("/Teacher/AddTeachers", Method.POST)
    {
        RequestFormat = DataFormat.Json
    };
    request.AddJsonBody(profesores);
    var response = client.Execute(request);
    if (response.StatusCode != HttpStatusCode.OK)
    {
        throw new Exception("Error teacherAddTeachers:" + response.Content);
    }
}
```

Figura 4.13: Captura parcial de clase TeacherDAO, con el método AddTeachers. (Los otros métodos operacionales de la clase siguen la misma estructura con llamadas a distintos endpoints).

Finalmente, se realiza un log que dice "INICIO ENVIO PROFESORES A GV", para llamar a los métodos `AddTeachers(agregarProfesores)`, `UpdateTeachers(actualizarProfesores)` y `DeleteTeachers(borrarProfesores)` de `StudentDAO`, en ese orden, con las listas respectivas de los métodos anteriores, terminando con un log "FIN SINCRONIZACION PROFESORES".

En los tres métodos, el procedimiento es muy similar: se realiza una request POST a la URL por definir en UrlApiVictoria, a los endpoint "`/Teacher/AddTeachers/`", "`/Teacher/DisableTeachers/`" y "`/Teacher/UpdateTeachers/`", añadiendo a la request una versión serializada en JSON de las respectivas listas entregadas. La respuesta es chequeada si vuelve con un código HTTP 200 (OK). En caso contrario, dispara una Exception con la descripción "`Error student[Add/Disable/Update]Students:`" y el contenido de la Response.

4.2.4. Envío de datos de raciones

El controlador principal llama al método `SendRations()` de `RationBusiness`. Este inicia el log con las frases "INICIO SINCRONIZACION RACIONES" e "INICIO OBTENCION RACIONES DESDE GV", para luego ejecutar los métodos `GetLastRationSentId()` (que obtiene el id de la última ración enviada a JUNAEB) y `GetPendingRations(id)` (que obtiene los datos de todas las raciones pendientes de envío, a partir de la id entregada), ambos métodos pertenecen a `RationDAO`.

Los datos a enviar de las raciones consumidas por los alumnos corresponden a los descritos en el Anexo A, en el documento *Implementación Servicio de Inserción de Raciones RBD*, en su apartado *Descripción* (página 68).

En el caso de `GetLastRationSentId`, se realiza una request POST a la API `JunaebRawData`, al endpoint `/Ration/GetLastRationSentId/`. La respuesta es chequeada si vuelve con un código HTTP 200 (OK). De ser así, este endpoint devuelve un JSON que debe ser deserializado en un long. En caso contrario, dispara una Exception con la descripción "`Error rationGetLastRationSentId:`" y el contenido de la Response.

```
public class RationBusiness : IRationBusiness
{
    2 references
    public void SendRations()
    {
        GeneralHelper.Log("INICIO SINCRONIZACION RACIONES");
        IJunaebRationDAO junaebRationDAO = new JunaebRationDAO();
        IRationDAO rationDAO = new RationDAO();

        GeneralHelper.Log("INICIO OBTENCION RACIONES DESDE GV");
        long idUltimaRacionEnviada = rationDAO.GetLastRationSentId();
        List<RacionJunaebAPI> raciones = rationDAO.GetPendingRations(idUltimaRacionEnviada);

        GeneralHelper.Log("INICIO ENVIO RACIONES A JUNAEB");
        junaebRationDAO.SendRations(raciones);

        GeneralHelper.Log("FIN SINCRONIZACION RACIONES");
    }
}
```

Figura 4.14: Método principal de la capa de negocios del módulo de envío de raciones.

Por otro lado, `GetPendingRations` también realiza una request POST a `JunaebRawData`, pero añadiendo el id de la última ración enviada como un JsonBody. Esta request se hace al endpoint `/Ration/GetPendingRations/`. La respuesta es chequeada si vuelve con un código HTTP 200 (OK). De ser así, este endpoint devuelve un JSON que debe ser deserializado en una lista de objetos `RacionJunaebAPI`, definido al igual que todos los Models en la carpeta `ApiModels` señalada en la sección 4.4.1. En caso contrario, dispara una Exception con la descripción "`Error rationGetPendingRations:`" y el contenido de la Response.

```
public List<RacionJunaebAPI> GetPendingRations(long idUltimaRacionEnviada)
{
    IRestClient client = GeneralHelper.GetVictoriaClient();
    var request = new RestRequest("/Ration/GetPendingRations/", Method.GET)
    {
        RequestFormat = DataFormat.Json
    };
    ProviderFilter filter = new ProviderFilter()
    {
        Provider = idUltimaRacionEnviada.ToString()
    };
    request.AddJsonBody(filter);
    var response = client.Execute(request);
    if (response.StatusCode != HttpStatusCode.OK)
    {
        throw new Exception("Error rationGetPendingRations:" + response.Content);
    }
    var content = response.Content; //Get Response
    List<RacionJunaebAPI> racionesJunaeb = JsonConvert.DeserializeObject<List<RacionJunaebAPI>>(content);
    return racionesJunaeb;
}
```

Figura 4.15: Método GetPendingRations de RationDAO.

Luego de la obtención de datos, se agrega al log el mensaje "`INICIO ENVIO RACIONES A JUNAEB`", y se envía la lista de raciones a JUNAEB mediante el método `SendRations(listaRaciones)` de `JunaebRationDAO`.

`SendRations(List<RacionJunaebAPI> raciones)` realiza una request POST a la URL https://serviciobio-test.junaeb.cl/, al endpoint "`RESTfulBiometrico/json/buscarFocalizado/setListaFocaCertiPRO`". En el archivo `App.Config`, se define la URL, el usuario y contraseña como `UrlApiJunaeb`, `UsuarioApiJunaeb` y `PassApiJunaeb`. Este endpoint devuelve un JSON que debe ser deserializado en una lista de objetos `RespuestaJunaebAPI`, que está definido en la carpeta `ApiModels` señalada en la sección 4.4.1.

```
public List<RespuestaJunaebAPI> SendRations(List<RacionJunaebAPI> raciones)
{
    RestClient client = GeneralHelper.GetJunaebClient();
    var request = new RestRequest(
        "RESTfulBiometrico/json/buscarFocalizado/setListaFocaCertiPRO",
        Method.POST); // /User: Operation. Always use "POST"
    request.AddParameter("application/json", JsonConvert.SerializeObject(raciones), ParameterType.RequestBody);
    var response = client.Execute(request);
    var content = response.Content;
    List<RespuestaJunaebAPI> respuestas = JsonConvert.DeserializeObject<List<RespuestaJunaebAPI>>(content);
    return respuestas;
}
```

Figura 4.16: Método SendRations de JunaebRationDAO.

Finalmente, la integración termina su ejecución con los mensajes "FIN SINCRONIZACION RACIONES" y "Fin de la ejecución del programa."

4.3. JunaebRawData

4.3.1. Aspectos Generales

JunaebRawData es una aplicación de consola que funciona como API para el proyecto IntegracionJunaeb, desarrollada en C# con el framework .NET Core 2.1. La aplicación al iniciarse, llama al método CreateWebHostBuilder, que inicializa una instancia de objeto WebHostBuilder usando Kestrel como web server, carga las configuraciones escritas en appsettings.json, carga las configuraciones de las variables de ambiente, ve los parámetros que vengan de la línea de comandos, configura el log de consola, habilita la integración de Kestrel con IIS y permite a .NET Core añadir sus configuraciones predeterminadas al proyecto. Además, se le indica al Builder que puede usar Application Insights para obtener métricas de uso en vivo al momento de ejecutarse, y se le indica que use las configuraciones de Startup definidas en Startup.cs

```
public class Program
{
    0 references | 0 exceptions
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    1 reference | 0 exceptions
    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseApplicationInsights()
            .UseStartup<Startup>();
}
```

Figura 4.17: Método principal de JunaebRawData.

Luego de este proceso, se le indica al Builder que debe construir la aplicación web y correrla hasta que se detenga el host.

La solución `JunaebRawData` cuenta con ocho proyectos:

- RawData.WebApi, es el proyecto principal. Posee los archivos Program.cs y Startup.cs para inicializar la API, además del archivo "appsettings.json" que contiene los strings de conexión a las bases de datos. Por último, tiene la carpeta con los controladores con los endpoints de alumnos, profesores PAE y raciones de casino.
- RawData.Business tiene los métodos que hacen conexión entre los controladores y los DAO.
- RawData.IBusiness tiene las interfaces de los métodos incluidos en RawData.Business.
- Los métodos de RawData.DAO realizan las llamadas a la base de datos para retornar los datos existentes.
- RawData.IDAO contiene las interfaces de los métodos de RawData.DAO.
- RawData.ServiceRegistration contiene la lista de servicios disponibles para realizar inyección de dependencias.
- RawData.ViewModels contiene las estructuras de los objetos MarcaVM y VoucherCasinoVM.
- RawData.Tests, que tiene la posibilidad de implementar tests unitarios a los métodos.

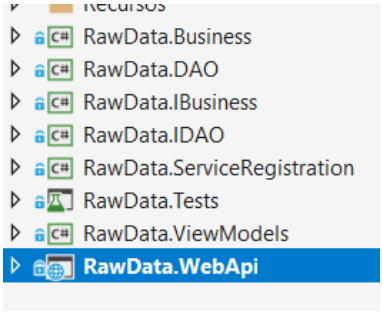


Figura 4.18: Listado de Proyectos contenidos en JunaebRawData.

Cabe destacar que `.NET Core` soporta el patrón de diseño de software de Inyección de Dependencias (DI, Dependency Injection), que es una técnica para lograr la Inversión de Control (IoC, Inversion of Control) entre clases y sus dependencias. Dado que no es el objetivo de esta memoria explicar en detalle estos conceptos, se resumirá DI en que un objeto da las dependencias de otro. En este caso específico, las interfaces proveen los métodos de las clases, para así, crear las instancias de objetos a través de las interfaces y no una instancia directa de la clase. Mientras tanto, IoC dicta que una clase no debe configurar sus dependencias estáticamente, sino a través de otra clase externa. Todo esto, para lograr mayor legibilidad y extensibilidad en el código. Una vez que la API queda corriendo, los controladores quedan a la espera de una request. Las siguientes subsecciones se dividirán en cada uno de estos controladores.

```
public static void ConfigureServices(IServiceCollection services)
{
    services.Add(ServiceDescriptor.Transient<IAlumnoBusiness, AlumnoBusiness>());
    services.Add(ServiceDescriptor.Transient<IMarcaBusiness, MarcaBusiness>());
    services.Add(ServiceDescriptor.Transient<IProfesorBusiness, ProfesorBusiness>());
    services.Add(ServiceDescriptor.Transient<IRacionBusiness, RacionBusiness>());
    services.Add(ServiceDescriptor.Transient<IVoucherCasinoBusiness, VoucherCasinoBusiness>());

    services.Add(ServiceDescriptor.Transient<IAlumnoDAO, AlumnoDAO>());
    services.Add(ServiceDescriptor.Transient<IMarcaDAO, MarcaDAO>());
    services.Add(ServiceDescriptor.Transient<IProfesorDAO, ProfesorDAO>());
    services.Add(ServiceDescriptor.Transient<IRacionDAO, RacionDAO>());
    services.Add(ServiceDescriptor.Transient<IVoucherCasinoDAO, VoucherCasinoStorageDAO>());
}
```

Figura 4.19: ServiceConfiguration en JunaebRawData.

4.3.2. StudentController

Este controlador contiene cuatro endpoints: “GetAllStudents”, que recibe un objeto que trae el id del proveedor, además de “AddStudents”, “DeleteStudents” y “UpdateStudents”, que reciben la lista de los alumnos a realizar la operación respectiva. Todos los métodos devuelven una lista de objetos AlumnoDTO, a través de un ActionResult, que empaqueta la respuesta en un objeto a través de su atributo Content, y retorna el estado de la respuesta. Dentro de cada método, se encuentra una llamada a ServiceLocator, que obtiene la instancia de IAlumnoBusiness y permite ejecutar los métodos respectivos en AlumnoBusiness. Ahí, lo único que hace cada método es llamar a su respectivo método en AlumnoDAO.

```
[Route("api/[controller]")]
[ApiController]
0 references
public class StudentController : BaseController
{
    [HttpPost("GetAllStudents")]
    0 references | 0 requests | 0 exceptions
    public ActionResult<List<AlumnoDTO>> GetAllStudents(FiltroProveedorVM filtroProveedor)
    {
        var studentBusiness = ServiceLocator.GetService<IAlumnoBusiness>();
        var retorno = studentBusiness.FindAllStudents(filtroProveedor);
        return Ok(retorno);
    }
```

Figura 4.20: Captura parcial de controlador StudentController de JunaebRawData.

Dentro de cada método de AlumnoDAO, se realiza una llamada a la base de datos, abriendo una nueva conexión SQL con los datos contenidos en la variable “ProdDB” de appsettings.json, y si el string contiene localdb, se añade a la base de datos un archivo “Calculo_Database.mdf” que actúa como base de datos local.

En el método FindAllStudents, se realiza la query

Código 4.1: Query FindAllStudents

```
1 SELECT codAlumno, codRbd, codComuna, codRegion, licitacion, curso, letraCurso, nivel,
    ↪ anhoEscolar, autorizacion, fechaAutorizacion, serDesayuno, serAlmuerzo,  serOnce,
    ↪ serCena, serColacion, serAlmuerzoViernes, rbdFicticio, mes, anho, institucion,
    ↪ alumnoFocalizado, prestador, habilitado
2 FROM AlumnosJunaeb WHERE prestador = @proveedor AND habilitado = 1;
```

```
public class AlumnoBusiness : IAlumnoBusiness
{
    private readonly IAlumnoDAO AlumnoDAO;

    0 references | 0 exceptions
    public AlumnoBusiness(IAlumnoDAO AlumnoDAO)
    {
        this.AlumnoDAO = AlumnoDAO;
    }

    2 references | 0 exceptions
    public List<AlumnoDTO> FindAllStudents(FiltroProveedorVM filtroProveedor)
    {
        return this.AlumnoDAO.FindAllStudents(filtroProveedor);
    }
```

Figura 4.21: Captura parcial de capa de negocios AlumnoBusiness de JunaebRawData.

donde proveedor es el atributo Provider en en FiltroProveedorVM.

```
public static string SqlServer
{
    get
    {
        string rst = ConfigurationHelper.GetConnectionString("ProdDB");

        if (!string.IsNullOrEmpty(rst) && rst.Contains("localdb"))
        {
            var appDataDir = ";AttachDbFilename=" + Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Calculo_Database.mdf");
            rst += appDataDir;
        }

        return rst;
    }
}
```

Figura 4.22: Configuraciones de llamada a servidor SQL en JunaebRawData.

El método AddStudents contiene la query

Código 4.2: Query AddStudents

```
INSERT INTO AlumnosJunaeb(codAlumno,codRbd,codComuna,codRegion,licitacion,curso,
    ↪ letraCurso,nivel,anhoEscolar,autorizacion,fechaAutorizacion,serDesayuno,serAlmuerzo,
    ↪ serOnce,serCena,serColacion,serAlmuerzoViernes,rbdFicticio,mes,anho,institucion,
    ↪ alumnoFocalizado,prestador,habilitado)
 VALUES
(@CodAlumno, @CodRbd, @CodComuna, @CodRegion, @Licitacion, @Curso, @LetraCurso,
    ↪ @Nivel, @AnhoEscolar, @Autorizacion, @FechaAutorizacion, @SerDesayuno,
    ↪ @SerAlmuerzo, @SerOnce, @SerCena,@SerColacion,@SerAlmuerzoViernes,
    ↪ @RbdFicticio,@Mes,@Anho,@Institucion,@AlumnoFocalizado,@Prestador,1);
```

Donde cada uno de los parámetros es un atributo de AlumnoDTO.

El método DeleteStudents contiene la query

Código 4.3: Query DeleteStudents

```
UPDATE AlumnosJunaeb SET habilitado = 0 WHERE codAlumno in (@idAlumnos);
```

Donde idAlumnos es un string que contiene todos los atributos CodAlumno de la lista pasada como parámetro, separados por coma.

El método UpdateStudents itera por cada alumno recibido a actualizar la siguiente query:

Código 4.4: Query UpdateStudents

```
UPDATE AlumnosJunaeb SET
                    codRbd = @CodRbd,
                    codComuna = @CodComuna,
                    codRegion = @CodRegion,
                    licitacion = @Licitacion,
                    curso = @Curso,
                    letraCurso = @LetraCurso,
                    nivel = @Nivel,
                    anhoEscolar = @AnhoEscolar,
                    autorizacion = @Autorizacion,
                    fechaAutorizacion = @FechaAutorizacion,
                    serDesayuno = @SerDesayuno,
                    serAlmuerzo = @SerAlmuerzo,
                    serOnce = @SerOnce,
                    serCena = @SerCena,
                    serColacion = @SerColacion,
                    serAlmuerzoViernes = @SerAlmuerzoViernes,
                    rbdFicticio = @RbdFicticio,
                    mes = @Mes,
                    anho = @Anho,
                    institucion = @Institucion,
                    alumnoFocalizado = @AlumnoFocalizado,
                    prestador = @Prestador,
                    WHERE codAlumno = @CodAlumno;
```

Donde cada uno de los parámetros es un atributo de AlumnoDTO, asignando el id del alumno a actualizar según el atributo CodAlumno.

4.3.3. TeacherController

Este controlador tiene cuatro endpoints disponibles: “GetAllTeachers”, que recibe un objeto que trae el id del proveedor, además de “AddTeachers”, “DeleteTeachers” y “UpdateTeachers”, que reciben la lista de los profesores a realizar la operación respectiva. Todos los métodos devuelven una lista de objetos ProfesorDTO, a través de un ActionResult, que empaqueta la respuesta en un objeto a través de su atributo Content, y retorna el estado de la respuesta. Dentro de cada método, se encuentra una llamada a ServiceLocator, que obtiene la instancia de IProfesorBusiness y permite ejecutar los métodos respectivos en ProfesorBusiness. Ahí, lo único que hace cada método es llamar a su respectivo método en ProfesorDAO. Dentro de cada método de ProfesorDAO, se realiza una llamada a la base de datos, abriendo una nueva conexión SQL con los datos contenidos en la variable “ProdDB” de appsettings.json, y si el string contiene localdb, se añade a la base de datos un archivo “Calculo_Database.mdf” que actúa como base de datos local.

Figura 4.23: Captura parcial de controlador TeacherController de JunaebRawData. (Nótese que los métodos están abreviados, porque la llamada a la capa de negocios es similar a StudentController).

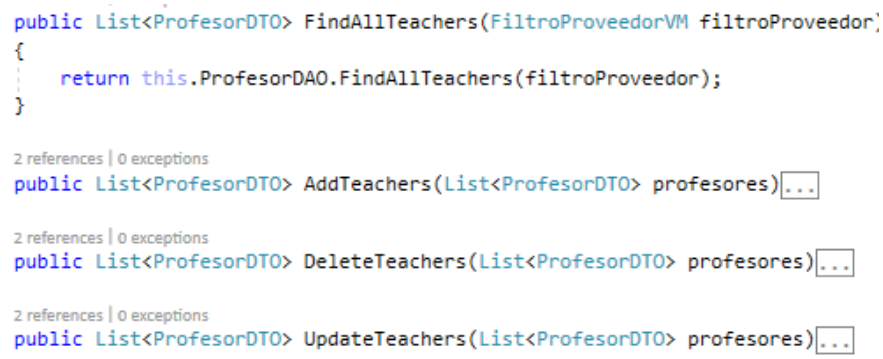


Figura 4.24: Capa de negocios ProfesorBusiness de JunaebRawData. (Nótese que los métodos están abreviados, porque la llamada a la capa de negocios es similar a AlumnoBusiness).

En el método FindAllTeachers, se realiza la query

Código 4.5: Query FindAllTeachers

```
1 SELECT idProfesor, profesorId, institucion, region, rbdId, cargo, rut, nombres,
    ↪ apellidoPaterno, email, telefono, apellidoMaterno, prestador, habilitado
2 FROM ProfesoresJunaeb WHERE prestador = @proveedor AND habilitado = 1;
```

donde proveedor es el atributo Provider en en FiltroProveedorVM.

El método AddTeachers contiene la query

Código 4.6: Query AddTeachers

```
1 INSERT INTO ProfesoresJunaeb(institucion, region, rbdId, cargo, rut, nombres,
    ↪ apellidoPaterno, email, telefono, apellidoMaterno, prestador, habilitado)
2 VALUES (@Institucion, @Region, @RbdId, @Cargo, @Rut, @Nombres, @ApellidoPaterno,
    ↪ @Email, @Telefono, @ApellidoMaterno, @Prestador, 1);
```

Donde cada uno de los parámetros es un atributo de ProfesorDTO.

El método DeleteTeachers contiene la query

Código 4.7: Query DeleteTeachers

```
UPDATE ProfesoresJunaeb SET habilitado = 0 WHERE rut in (@idProfesores);
```

Donde idProfesores es un string que contiene todos los atributos Rut de la lista pasada como parámetro, separados por coma.

El método UpdateTeachers itera por cada profesor a actualizar la siguiente query:

Código 4.8: Query UpdateStudents

```
UPDATE ProfesoresJunaeb SET
                    institucion = @Institucion,
                    region = @Region,
                    rbdId = @RbdId,
                    cargo = @Cargo,
                    nombres = @Nombres,
                    apellidoPaterno = @apellidoPaterno,
                    email = @Email,
                    telefono = @Telefono,
                    apellidoMaterno = @ApellidoMaterno,
                    prestador = @Prestador,
                    WHERE rut = @Rut;
```

Donde cada uno de los parámetros es un atributo de ProfesorDTO, asignando el id del alumno a actualizar según el atributo CodProfesor.

4.3.4. RationController

Este controlador posee dos endpoints: “GetLastRationSentId”, que recibe un objeto que contiene el id del proveedor y retorna un id correspondiente a la primera ración del día. El otro método es “GetPendingRations”, que recibe un objeto que contiene el id de la ración desde la cual obtener la lista de raciones RacionDTO a devolver. Ambos métodos retornan sus resultados a través de un ActionResult, que empaqueta la respuesta en un objeto a través de su atributo Content, y retorna el estado de la respuesta. Dentro de cada método, se encuentra una llamada a ServiceLocator, que obtiene la instancia de IRacionBusiness y permite ejecutar los métodos respectivos en RacionBusiness. Ahí, lo único que hace cada método es llamar a su respectivo método en RacionDAO.

Al igual que los dos DAOs explicados anteriormente, dentro de cada método de RacionDAO se abre una nueva conexión SQL con los datos contenidos en la variable “ProdDB” de appsettings.json, y si el string contiene localdb, se añade a la base de datos un archivo “Calculo_Database.mdf” que actúa como base de datos local.

En el método GetLastRationSentId, se realiza la query

```
[Route("api/[controller]")]
[ApiController]
0 references
public class RationController : BaseController
{
    [HttpPost("GetLastRationSentId")]
    0 references | 0 requests | 0 exceptions
    public ActionResult<long> GetLastRationSentId(FiltroProveedorVM filtroProveedor)
    {
        var rationBusiness = ServiceLocator.GetService<IRacionBusiness>();
        var retorno = rationBusiness.GetLastRationSentId(filtroProveedor);
        return Ok(retorno);
    }

    [HttpPost("GetPendingRations")]
    0 references | 0 requests | 0 exceptions
    public ActionResult<List<RacionDTO>> GetPendingRations(FiltroProveedorVM filtroProveedor)
    {
        var rationBusiness = ServiceLocator.GetService<IRacionBusiness>();
        var retorno = rationBusiness.GetPendingRations(filtroProveedor);
        return Ok(retorno);
    }
}
```

Figura 4.25: Controlador RationController de JunaebRawData.

Código 4.9: Query GetLastRationSentId

```
1 SELECT TOP (1) racionId FROM RacionJunaeb WHERE prestador = @proveedor AND
      ↪ hora LIKE '@fecha%';
```

donde fecha es la fecha donde se realiza la consulta en formato “yyyy-MM-dd”.

El método GetPendingRations contiene la query

Código 4.10: Query AddTeachers

```
1 SELECT institucion, prestador, region, comunaId, rbdId, codAlumno, nivel, fechaCerti,
      ↪ servicioCerti, hora, tipoCerti, observacion
2 FROM RacionJunaeb WHERE racionId >= @id;
```

Donde cada uno de los parámetros es un atributo de RacionDTO.

4.3.5. Archivos comunes al proyecto

Dentro del proyecto, se encuentra la carpeta Comunes, que contiene archivos necesarios para que la API funcione correctamente.

La carpeta Extensions contiene unas cuantas extensiones que ayudan a simplificar algunos métodos:

- IEnumerableExtension posee el método IsNullOrEmpty<T>(this IEnumerable<T> enu-

```
2 references
public class RacionBusiness : IRacionBusiness
{
    private readonly IRacionDAO RacionDAO;

    0 references | 0 exceptions
    public RacionBusiness(IRacionDAO RacionDAO)
    {
        this.RacionDAO = RacionDAO;
    }

    2 references | 0 exceptions
    public long GetLastRationSentId(FiltroProveedorVM filtroProveedor)
    {
        return this.RacionDAO.GetLastRationSentId(filtroProveedor);
    }

    2 references | 0 exceptions
    public List<RacionDTO> GetPendingRations(FiltroProveedorVM filtroProveedor)
    {
        return this.RacionDAO.GetPendingRations(filtroProveedor);
    }
}
```

Figura 4.26: Capa de negocios RacionBusiness de JunaebRawData.

merable), que permite chequear si una colección IEnumerable posee elementos o es nula.

- StorageExtension posee el método IList<T> ExecuteQuery<T>(this CloudTable table, TableQuery<T> query, CancellationToken ct = default, Action<IList<T» onProgress = null) where T : ITableEntity, new(), que permite que sobre una clase CloudTable ejecute una query TableQuery hacia un Storage. Esto en el caso que se decida avanzar el guardado de vouchers a un Storage. Dichas queries se detallan en VoucherCasinoDAO, separando por la PartitionKey o RowKey a buscar.
- TrimStringExtension posee el método TrimStringProperties, que retorna un objeto con sus propiedades de string (de tener) sin espacios delante y atrás del texto.

Además, posee distintos Helper que funcionan como métodos independientes:

- AdapterHelper posee el método TransformsStringToTimeSpan, que retorna un objeto TimeSpan en base a un texto en formato “HH:mm”.
- AsyncHelper tiene el método RunSync, que permite que las consultas asíncronas sean síncronas.
- CloneHelper posee el método CloneJson, que permite realizar DeepCopy de un objeto, en lugar de ShallowCopy que sucede por defecto en C#.
- ConfigurationHelper obtiene la ruta por defecto de “appsettings.json”, y lee los strings de conexión y values con los métodos GetConnectionString y Value, respectivamente.

- InsightHelper añade el log de excepciones y de métrica a las herramientas de telemetría de Application Insights.
- RetryHelper incluye el método de Execute, que realiza la ejecución de un método un número repetido de veces (por defecto, tres). La reiteración ocurre cuando hay una excepción que se dispara, lo que itera de nuevo sobre el método. Esto es usado para ver si es seguro ejecutar las consultas dirigidas a un Storage, ya sea porque no lo encuentra o si las credenciales no están correctas, o hay latencia que deje una desconexión temporal. Esto fue pensando en la expansión de este proyecto hacia el uso de un Storage en lugar de una BD para los tickets de alimentación.
- ServiceLocator permite, en conjunto con RawData.ServiceRegistration, realizar la inversión de control.

Por último, el proyecto Enum lista RawData.Commons lista algunos objetos a considerar, como los DTO AlumnoDTO, ProfesorDTO y RacionDTO, Excepciones, objetos de filtro, y constantes (como MinDateSql, la fecha mínima para consultas SQL y de Storage).

```
public static class CloneHelper
{
    2 references | 0 exceptions
    public static T CloneJson<T>(object source)
    {
        if (source == null)
        {
            return default(T);
        }

        var deserializeSettings = new JsonSerializerSettings
        {
            ObjectCreationHandling = ObjectCreationHandling.Replace,
            DateFormatString = "yyyy-MM-dd HH:mm:ss"

        };
        return JsonConvert.DeserializeObject<T>(
            JsonConvert.SerializeObject(source), deserializeSettings);
    }
}
```

Figura 4.27: Método CloneJson de clase CloneHelper.

4.4. PlataformaJunaeb

4.4.1. Aspectos Generales

La plataforma es una aplicación web desarrollada en C# con el framework .NET Core 3.1. La aplicación sigue el patrón MVC (Model View Controller). Este patrón de diseño separa la responsabilidad de los componentes en tres:

- El modelo o model se encarga de las transacciones con la base de datos, obtener los datos en bruto.
- La vista o view vela que las interfaces muestren de manera correcta de cara al usuario.

```
public static class InsightsHelper
{
    private static readonly TelemetryClient tc = new TelemetryClient();

    1 reference | 0 exceptions
    public static void LogException(Exception ex, Dictionary<string, string> properties = null)
    {
        tc.TrackException(ex, properties);
    }

    0 references | 0 exceptions
    public static void LogMetric(string metrica, Dictionary<string, string> properties = null)
    {
        var sample = new MetricTelemetry
        {
            Name = metrica
        };

        if (properties != null)
        {
            foreach (KeyValuePair<string, string> prop in properties)
            {
                sample.Properties.Add(prop.Key, prop.Value);
            }
        }

        tc.TrackMetric(sample);
    }
}
```

Figura 4.28: Métodos LogException y LogMetric de InsightHelper.

- El controlador o controller chequea que los cambios que haga el usuario en la vista se vean reflejados en el modelo.

La carpeta Models indica los modelos de datos y cómo se leerán, tanto para la vista pública, como el nombre del atributo con el que se buscará. Mientras que dentro de la carpeta Pages, se encuentran los distintos templates en Razor .cshtml, con sus respectivos controladores .cshtml.cs. Dentro de las páginas de profesores, alumnos y raciones, Details se refiere a la página que muestra un elemento en detalle, mientras que Index se refiere a la tabla general de dicha categoría.

Cuando se inicia la aplicación, esta llama al método CreateWebHostBuilder, que inicializa una instancia de objeto WebHostBuilder usando Kestrel como web server, carga las configuraciones escritas en “appsettings.json”, carga las configuraciones de las variables de ambiente, ve los parámetros que vengan de la línea de comandos, configura el log de consola, habilita la integración de Kestrel con IIS y permite a .NET Core añadir sus configuraciones predeterminadas al proyecto. Además, se le indica al Builder que puede usar Application Insights para obtener métricas de uso en vivo al momento de ejecutarse, y se le indica que use las configuraciones de Startup definidas en Startup.cs.

Dentro de Startup.cs, se indica al host que puede usar Razor para crear los template y generar vista al usuario. Además, añade como contexto de dichos template los datos del servidor SQL que se indican en “appsettings.json”, bajo la variable PlataformaJunaebContext. Para dejar el identificador simple, se deja que sea sin identificadores, por lo que no es necesario tener un usuario y contraseña en esta implementación, pero está la posibilidad de extender la funcionalidad.

Una vez que inicia obtiene 4 servicios: “Alumnos”, “Profesores”, “Raciones” y “Comunas” (este servicio opera para reconocer la comuna del servicio, dado que está por códigos numéricos en la base de datos). Luego, pobla la base de datos si es que está vacía, y finalmente, se le indica al host que corra la aplicación.

```
public static class RetryHelper
{
    private const string DEFAULT_TOTAL_ATTEMPTS = "3";

    private static readonly string TOTAL_ATTEMPTS =
        Environment.GetEnvironmentVariable("maxAttempts") ?? DEFAULT_TOTAL_ATTEMPTS;

    5 references | 0 exceptions
    public static T Execute<T>(Func<T> method)...

    0 references | 0 exceptions
    public static void Execute(Action method)
    {
        int currentRetry = 1;

        for (; ; )
        {
            try
            {
                method();
                break;
            }
            catch (Exception)
            {
                currentRetry++;
                if (currentRetry > int.Parse(TOTAL_ATTEMPTS))
                {
                    throw;
                }
            }
            Thread.Sleep(50);
        }
    }
}
```

Figura 4.29: Métodos Execute de RetryHelper, con sus constantes de máximos intentos permitidos.

4.4.2. Landing Page

En esta subsección se explica cómo se implementó la interfaz de la página principal. Además se mostrarán algunos problemas que surgieron al implementar este módulo.

La implementación de este módulo se realizó de forma incremental hasta llegar a la vista de la Figura 4.31. Inicialmente se creó una interfaz que mostraba únicamente el título de la vista. Luego, se implementa el iframe que lo contendrá.

El resto de esta vista se desarrolló en la herramienta de business analytics `Microsoft Power BI Desktop`. En ella, primero que todo se realizaron las conexiones con la base de datos que contiene los datos pertinentes a esta memoria, bajo la modalidad de conexión DirectQuery. Luego, se realizó el gráfico de barras de los estratos de alumnos, para luego hacer la matriz de raciones entregadas por región. Finalmente, se hizo un gráfico de manómetro respecto a las raciones asignadas vs raciones entregadas a los alumnos de manera diaria.

En el rincón superior izquierdo de la figura 4.31, Se pueden apreciar, como gráficos de barra, el reporte que indica la cantidad de servicios entregados en base al nivel educacional. Esta combinación se conoce como "estrato" para las empresas de reparto de alimentos, y es la principal división a la hora de tomar decisiones para estas empresas.

En la parte inferior izquierda, se encuentra la matriz de tipo de raciones entregadas por región. Esto permite conocer qué clase de alimentos se consumen en mayor o menor medida por zona geográfica, y destinar esfuerzos y recursos de parte de los administradores de JUNAEB en comunicar estos datos a las entidades responsables.

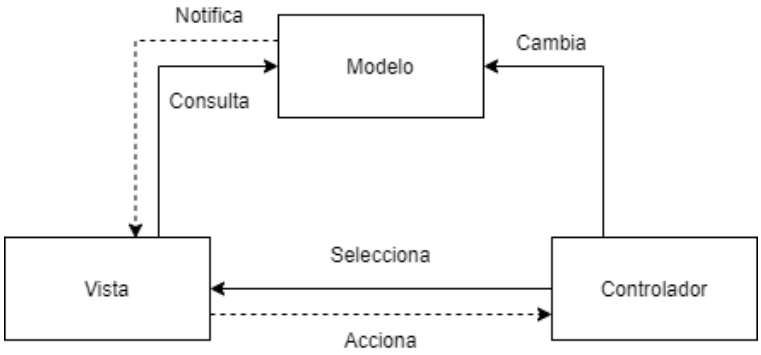
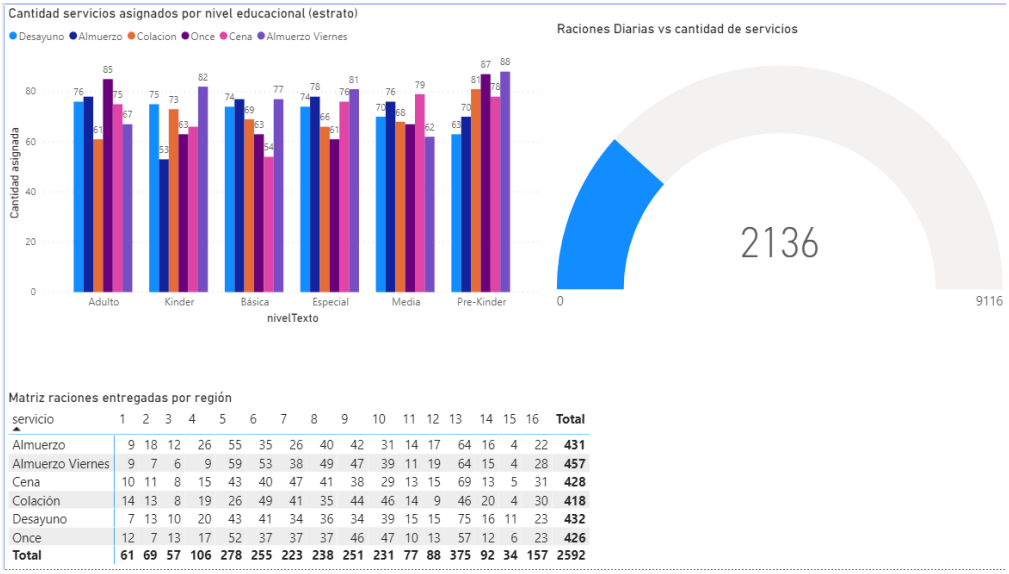


Figura 4.30: Diagrama de patrón MVC.



servicio	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Total
Almuerzo	9	18	12	26	55	35	26	40	42	31	14	17	64	16	4	22	**431**
Almuerzo Viernes	9	7	6	9	59	53	38	49	47	39	11	19	64	15	4	28	**457**
Cena	10	11	8	15	43	40	47	41	38	29	13	15	69	13	5	31	**428**
Colación	14	13	8	19	26	49	41	35	44	46	14	9	46	20	4	30	**418**
Desayuno	7	13	10	20	43	41	34	36	34	39	15	15	75	16	11	23	**432**
Once	12	7	13	17	52	37	37	37	46	47	10	13	57	12	6	23	**426**
Total	**61**	**69**	**57**	**106**	**278**	**255**	**223**	**238**	**251**	**231**	**77**	**88**	**375**	**92**	**34**	**157**	**2592**

Figura 4.31: Reportes principales de la Landing Page creados con Microsoft Power BI.

Por último, en el sector derecho de la misma figura, se encuentra el manómetro que indica la cantidad de raciones totales diarias entregadas comparado a la cantidad de raciones totales diarias destinadas a los alumnos. Esto permite a un administrador JUNAEB conocer de manera simple y rápida cuantos alimentos no están llegando a sus debidos beneficiarios.

Las principales dificultades a la hora de crear estos reportes fue no contar con la retroalimentación de las partes interesadas en el proceso. Esto, pues solo hubo una reunión con la empresa de distribución de alimentos **COAN Chile SpA.** el día 23 de enero de 2020 a las 10:00 AM, ubicada en Av. Ricardo Lyon 455. En dicha reunión, se pudo obtener la información de que el tipo de información que le interesa a COAN, y por lo general a las empresas del rubro, es la información de raciones entregadas, y la separación por estratos, debido a que el pago que realiza JUNAEB a estas depende de esa información.

Aparte de esa reunión, dado que las relaciones con JUNAEB no prosperaron, no se podía solicitar reuniones con las empresas del rubro. Además, dado lo mencionado en el capítulo 2, sección 2.4, la licitación se declaró desierta y se realizó otro proceso administrativo contemplado en la legislación vigente.

Con todo ello, se tomó el feedback entregado por el personal voluntario de GeoVictoria para poder armar un reporte que fuera comprensible y sencillo. Por otra parte, estaba la inexperiencia en el flujo completo de Power BI, pues la experiencia previa usando el sistema de business analytics era en la creación de reportes, más no en la publicación y distribución de éste.

4.4.3. Listado detallado de alumnos, profesores y raciones

En esta subsección se explica cómo se implementó la interfaz del listado detallado de alumnos. Además se mostrarán algunos problemas que surgieron al implementar este módulo. Cabe destacar que para los casos de listado detallado de profesores y de raciones, se siguió la misma secuencia de desarrollo, cambiando ligeros detalles tales como nombres de variables, o criterios de la barra de búsqueda, por lo que no se detendrá en el detalle de la implementación de estas vista.

La implementación de estos módulos se realizó de forma incremental hasta llegar a la vista de la Figura 4.32. Inicialmente se creó una interfaz que mostraba únicamente el título de la vista. Luego, se implementa la tabla en blanco con sus cabeceras. Después, se hizo el llenado de los datos, para continuar con una sección que mostrara los datos detallados de cada alumno. Finalmente, se implementó una barra de búsqueda de coincidencia total para encontrar alumnos por RBD o código de alumno, la que se modificó para aceptar coincidencias parciales.

Para acceder a esta vista, se debe seleccionar en la barra superior el link de “Alumnos”. En la parte superior, hay una barra de búsqueda para encontrar los alumnos pertenecientes a un establecimiento determinado, buscando de forma parcial por el RBD.

En la tabla, se pueden apreciar los campos que se ven en la imagen 4.32, además de un enlace a ver los datos en detalle de dicho alumno. Al entrar a dicho enlace, se pueden ver

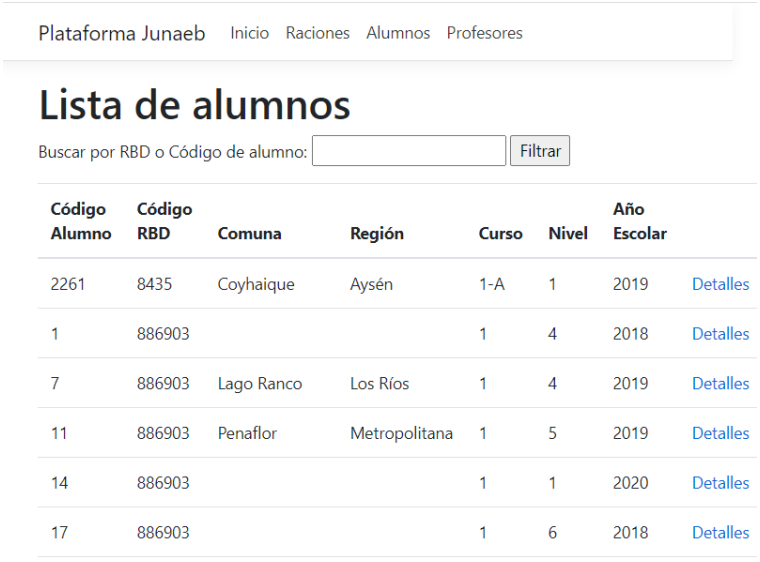


Figura 4.32: Apariencia módulo de alumnos.

todos los datos de un alumno, presente en la imagen siguiente 4.33. Se listará con mayor

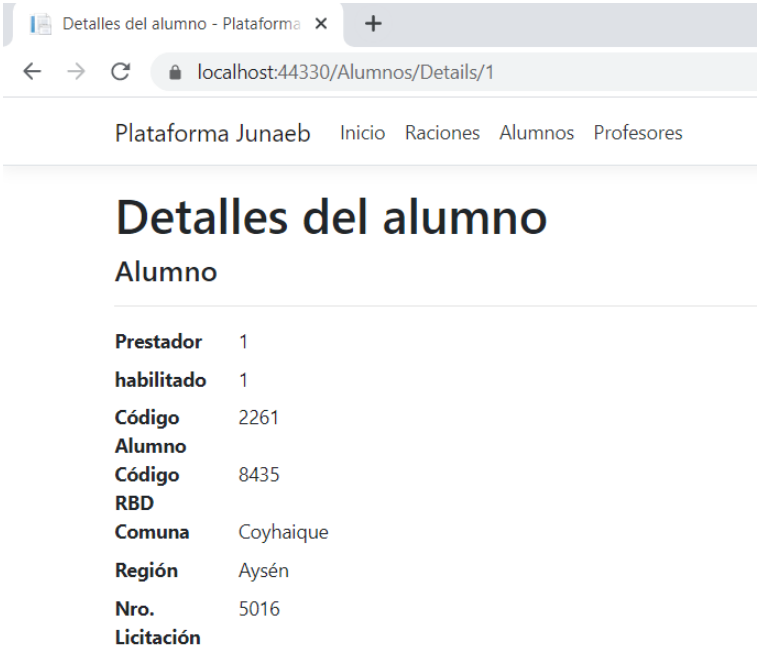


Figura 4.33: Apariencia de detalle de un alumno.

detalle los pasos para implementar esta interfaz:

- Crear los models de alumno para poder pedirlos a la base de datos y mostrarlos en el template.
- Crear la url con la que se llamará a la tabla principal. En este caso, será simplemente `/Alumnos`. Esta url está en `_Layout.cshtml`.
- Crear el template de este componente, llamado `Index.cshtml`. Esta vista llama a su controller, alojado en `Index.cshtml.cs`, el cual toma la información que le proveerá el modelo. Primero se realiza la vista sin esta llamada para comprobar que funcione.
- Luego, se crea el controller, obteniendo el contexto desde la base de datos.
- Después de rellenar exitosamente la tabla, se creó un enlace a una ventana de datos detallada. Dicha ventana se identificará con el código de alumno, por lo que la URL será `/Alumnos/[id]`, por ejemplo `/Alumnos/1`
- se procedió a crear una barra de búsqueda que filtre por el código RBD.
- Se crea el template general de la ventana detallada en `Details.cshtml`, este template contiene más datos del alumno, tales como sus raciones designadas, o la licitación a la que pertenece.
- Se crea el controller `Details.cshtml.cs`, asegurando que el id exista y que el código de la comuna exista.
- Finalmente, se procedió a mejorar la herramienta de búsqueda, añadiendo la capacidad que sea coincidencia parcial de RBD y de Código Alumno.

Muchos de los problemas que surgieron al desarrollar esta aplicación, fue más que nada por la inexperiencia al desarrollar front-end de parte propia, siendo mis experiencias personales más enfocadas al back-end. Por ejemplo, las interacciones entre la barra de búsqueda y el controlador, o el poder buscar los datos satisfactoriamente en las distintas URL. Aquello se resolvió más que nada buscando información y aprendiendo en el proceso cómo armar una aplicación web en una tecnología en la que tenía experiencia de back-end.

Capítulo 5

Validación de la implementación

Considerando la implementación descrita en el Capítulo 4, en este capítulo se describe los elementos a validar, la forma en que esta validación fue hecha o se planeó hacer, y un análisis de dichos datos. Este capítulo, al igual que el anterior, se separará por repositorios:

- Para la aplicación **IntegracionJunaeb**, se resumirán nuevamente los problemas de comunicación surgidos con JUNAEB, se detallará el plan de pruebas a seguir en conjunto en caso que se hubieran mantenido las comunicaciones, y se describirá el porqué de dichas pruebas.
- En el caso de la API **JunaebRawData**, se describirá el estado de los elementos al momento de realizar la validación, para luego describir las pruebas realizadas, y se finalizará con un análisis de los datos.
- Por último, para **PlataformaJunaeb** se mencionará el proceso de validación en reuniones con personal de GeoVictoria, se recogerán sus opiniones respecto a la aplicación web de cara al usuario y el plan de pruebas propuesto. Por último, se analizarán las respuestas para ver posibles mejoras a futuro.

En todos los casos, las pruebas de lado propio y el desarrollo de las aplicaciones se realizan en el equipo de trabajo propiedad del alumno memorista. En los casos que requieran servicios adicionales, se detallará.

5.1. IntegracionJunaeb

Al inicio del proceso de memoria, se tenía contemplado el apoyo de JUNAEB para contar con los datos de alumnos, profesores PAE y el envío de raciones de alimentación. Este proceso de colaboración fue truncado a mediados del mes de mayo año 2020, cuando JUNAEB bloqueó el acceso a los endpoints pertinentes a los webservice entregados en enero 2020.

Luego de ello, se realizó el proceso de solicitud vía correo electrónico con las personas encargadas, pero la respuesta entregada fue que se debía acudir a Consejo de Transparencia,

en la cual la solicitud de datos fue rechazada por "inexistencia de estos", aún describiendo los webservice habilitados.

Ante este escenario, las pruebas planificadas no se pudieron realizar, por lo que solamente se pasarán a describir:

Una vez finalizada la integración, se coordinaría una reunión con el Departamento de Informática de JUNAEB, para coordinar una prueba conjunta. Esto quiere decir que personal de JUNAEB confirmaría que recibieron de manera correcta los datos de raciones, y que de parte propia se recibieron los datos de alumnos y profesores. La integración se ejecutaría de manera local, en el computador propio del memorista, conectado a la red de internet de GeoVictoria. Se realizarían dos pruebas:

- Se realizaría el envío de una lista de raciones de prueba al endpoint https://serviciobio-test.junaeb.cl/RESTfulBiometrico/json/buscarFocalizado/setListaFocaCertiPRO, confirmando en el momento la recepción de los datos.
- Luego, se consultaría a https://serviciobio-test.junaeb.cl/RESTfulBiometrico/json/buscarFocalizado/obtenerProfesores/getProfesorResponsableBio/[Prestador] los datos de profesores y a https://serviciobio-test.junaeb.cl/RESTfulBiometrico/json/buscarFocalizado/obtenerAutorizados/[Prestador] los de alumnos, y se notificaría en la misma reunión que los datos fueron recibidos correctamente.

Además, se realizaron pruebas respecto a la generación de logs al pasar por los distintos módulos, aunque sin ejecutar las llamadas a los webservice. En el capítulo anterior se puede ver figuras 4.4, 4.5 y 4.6 de los logs generados de muestra, tanto vía consola como en un archivo `[fecha].txt`, siendo fecha el día de ejecución de la aplicación de integración.

Para llenar las bases de datos con datos que fueran suficientemente realistas, se procedió a rellenar las tablas con consultas personalizadas. Dichas consultas se escribieron en `Microsoft Excel` para poder realizar pequeños cambios, y de esta manera, el proceso se simplificara.

```
CREATE TABLE #ComunaTemp
(
      IdRegion VARCHAR(50),
      IdComuna VARCHAR(50)
)

DECLARE @cnt INT = 0;

WHILE @cnt < 2997
BEGIN
   INSERT INTO #ComunaTemp SELECT TOP 1 IdRegion, IdComuna FROM
ComunaJunaeb ORDER BY NEWID();
   SET @cnt = @cnt + 1;
END;

SELECT * FROM #ComunaTemp;
DROP TABLE #ComunaTemp;
```

Figura 5.1: Query SQL que genera 2997 tuplas aleatorias de IdComuna e IdRegion.

Primero, se crearon las consultas para rellenar las tablas de comunas disponibles. La información fue obtenida a través de la tabla de información de comunas y regiones del repositorio perteneciente al Ministerio de Ciencia, en su archivo `InformacionComunas.csv` [21].

Con ello, se creó la siguiente query para insertar datos en la tabla de alumnos. Para los datos de código de alumno, simplemente se siguió una secuencia lineal, mientras que las comunas de los alumnos se obtuvieron de la query de la figura 5.1. para los datos de nivel, simplemente se hizo un script en C# que imprimiera 2593 veces un entero pseudoaleatorio entre 1 y 6 (correspondientes a los códigos definidos por JUNAEB en su webservice), para el año escolar, un script similar, esta vez con un valor pseudoaleatorio entre 0 y 2, al que se le sumaría 2018 (para así abarcar los años 2018, 2019 y 2020), todas las fechas de autorización fueron definidas como el 1 de enero de dicho año, y para asignar los servicios se realizó la query random entre 0 y 1. Al final, las consultas armadas en excel columna por columna se concatenaban en un campo final, se copiaban a un archivo .sql y se ejecutaban.

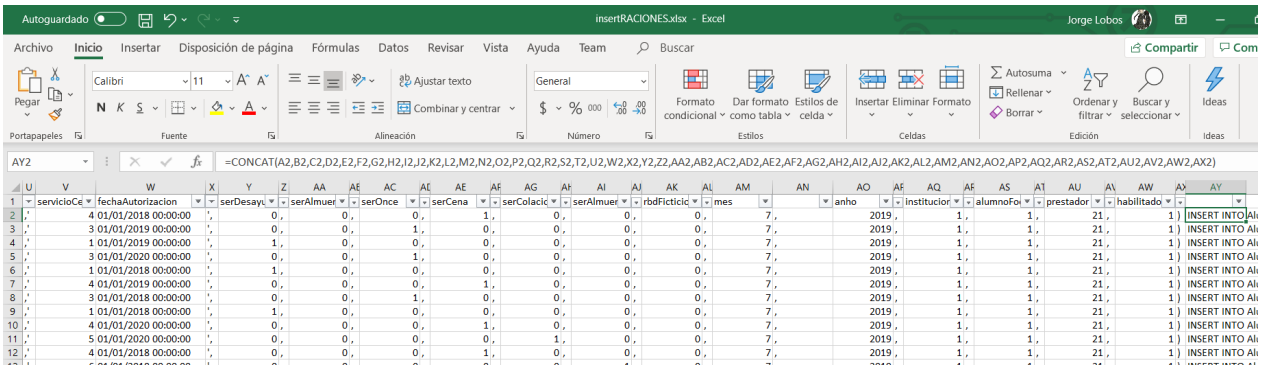


Figura 5.2: Captura parcial de plantilla Excel en la hoja destinada a consultas de alumnos.

De manera similar se llenaron los datos numéricos de las tablas de profesores, mientras que los nombres fueron llenados a mano, lo que llevó a tener solamente datos de 101 profesoras y profesores de prueba. Su correo corresponde a la letra inicial del primer nombre, el apellido paterno completo y la letra inicial del apellido materno, todos seguidos por el dominio uchile.cl y el texto completo en minúscula. Esto está destacado en la barra de escritura en la figura 5.3.

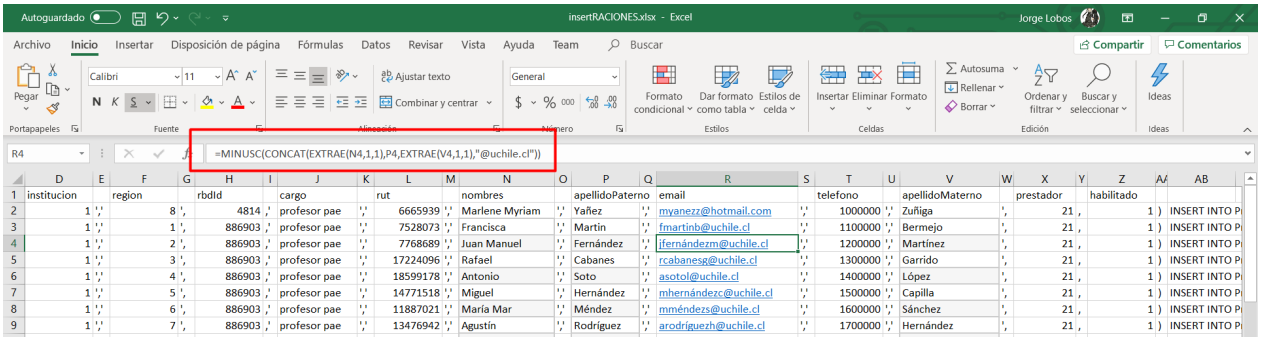


Figura 5.3: Captura parcial de plantilla Excel en la hoja destinada a consultas de profesores. Destacado en rojo la fórmula para crear correos electrónicos.

Por último, para rellenar la tabla de raciones, se eligieron los mismos criterios numéricos que el punto de alumnos, y con ello, se verifica que las raciones entregadas corresponden solamente a alumnos existentes, con su respectivo servicio de alimentación. Dicha correlación

se puede verificar en el campo servicioCerti en la figura 5.4 y en los campos de servicio en la figura 5.2.

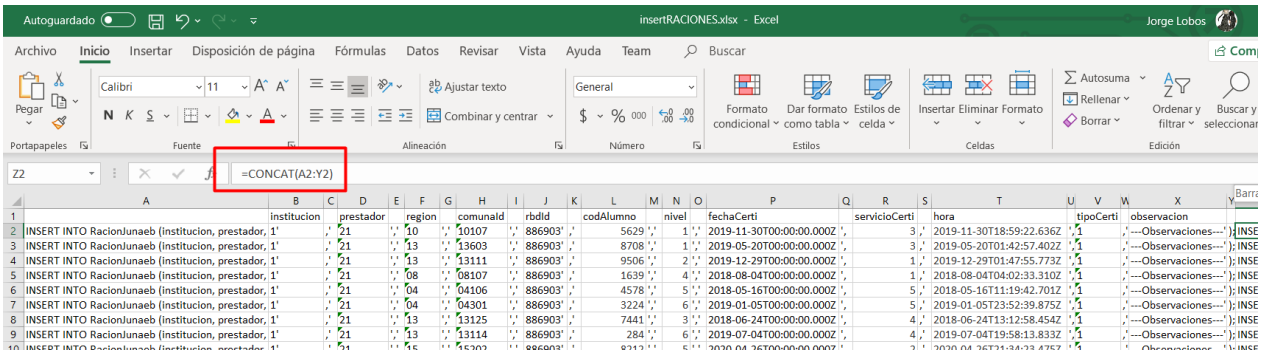


Figura 5.4: Captura parcial de plantilla Excel en la hoja destinada a consultas de raciones.

El objetivo de estas pruebas era validar que la comunicación entre JUNAEB y la integración propia fuera exitosa, para en un futuro poder alimentar la base de datos propia con información de alumnos beneficiarios y profesores PAE, al dejar en el Programador de Tareas de Windows, en un servidor dispuesto a aquello, una tarea que se ejecute entre una a seis veces al día con esas dos peticiones. Además, se enviaría seis veces al día, en horarios por coordinar con JUNAEB relativos a los servicios de alimentos entregados por el PAE, los datos de raciones entregadas a los alumnos. Se decide ese rango de frecuencias para la recepción de datos de alumnos y profesores dado que todos los días es posible que se modifique la lista de beneficiarios del programa PAE, por lo que define así la cota superior, y dado que queremos tener los usuarios ya procesados antes de que se envíe el conjunto de datos de raciones de alumnos a JUNAEB, se establece así la cota inferior. Cabe destacar que, al igual que el rango horario de raciones mencionado anteriormente, queda a futuro coordinar con JUNAEB el criterio que la organización posee para actualizar sus datos de alumnos y profesores.

5.2. JunaebRawData

Para realizar las pruebas de validación de la aplicación de integración, se crearon requests HTTP en el programa Postman v7.33.1 [6].

Postman es un software que facilita el testeo de API, dado que permite generar en pocos pasos consultas apuntando a la API elegida. Los beneficios de Postman incluyen:

- Creación de request HTTP, soportando distintos sistemas de autentificación, cookies, certificados, además del cuerpo de la request. Además, se pueden guardar las respuestas obtenidas.
- Se pueden organizar las request creadas en grupos llamados **Colecciones**. Así, las request pueden heredar propiedades de la colección, tales como credenciales de identificación y variables. Dichas variables pueden tener un valor a usar en las credenciales, en la URL o incluso en el cuerpo de la request.

- Postman permite crear equipos de trabajo (cosa que no fue el caso en esta memoria, pero es posible hacerlo en cuanto se extienda fuera del alcance de ésta). Las `Colecciones` y requests se pueden compartir con los equipos de trabajo que se tengan registrados.

Se realizaron pruebas del funcionamiento de la API JunaebRawData corriendo la API en el equipo local del memorista, sin navegadores abiertos más que una pestaña de Google Chrome que indicaba que la API estaba en ejecución recibiendo peticiones a través del puerto 60388. Esto se hizo creando una `Colección` en Postman llamada **JunaebRawData**, que tuviera las requests HTTP necesarias para realizar las pruebas.

En la figura 5.5, se puede ver la interfaz de una request en Postman. En la barra gris superior se puede apreciar el tipo de Request HTTP junto a la dirección en la que se hace. Abajo de ello, los headers están pre-rellenados, para que así solo se tenga que llenar el cuerpo de la request.

Como fue descrito en el capítulo 4, el endpoint recibe un objeto JSON que contiene como parámetro el proveedor `Provider` que consultará por los alumnos a los que corresponde dicho proveedor hacer entrega de alimentos.

Luego de ello, se le da al botón azul **Send**, y realiza la request. En ella, abajo del cuerpo de envío, al medio a la derecha de color verde se puede ver el estado de la respuesta, el tiempo que se demoró en procesar la request, y el tamaño de la respuesta. En este caso, respondió `200 OK`, con un tiempo de 3.26 segundos, y 1004.54 KB. Además, en la parte de abajo, se puede apreciar el cuerpo de la respuesta.

En las figuras 5.6 y 5.7, se realizan peticiones bastante similares a la de la figura 5.5, por lo que no se ahondará en ellas.

Con estas pruebas mencionadas en las subsecciones 1 y 2 de este capítulo, se verifica que el método en que se comunicarán los datos entre el sistema propuesto y los webservices de JUNAEB funcionen de manera correcta. A futuro, se podrán ampliar los servicios tanto en la API como en la integración de ser necesario, contando ya con una estructura base y un método de comunicación efectivo con la cual canalizar peticiones a la base de datos, y se comprueba que se cumple el objetivo específico 1 propuesto en el capítulo 1.

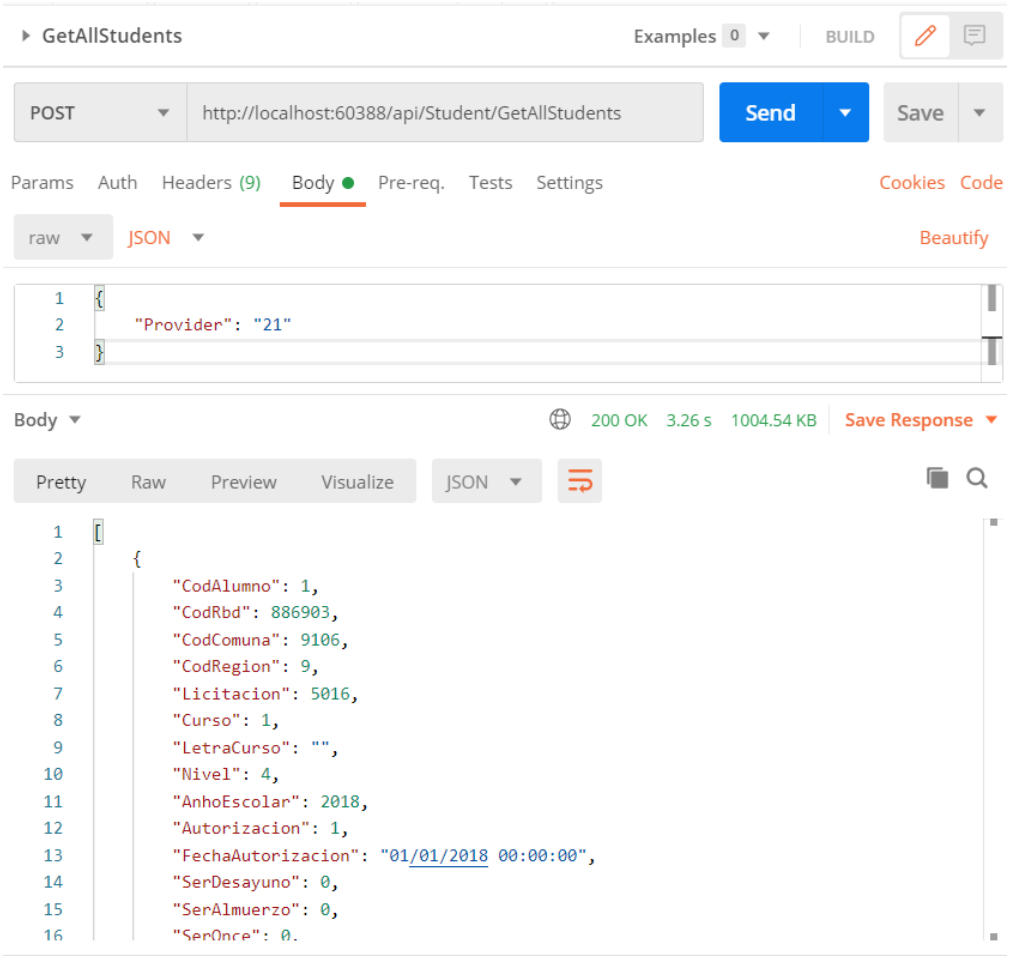


Figura 5.5: Prueba exitosa de obtención de alumnos vía JunaebRawData.

5.3. PlataformaJunaeb

A medida que se iba finalizando el desarrollo de la integración y la API, comenzaba el desarrollo de la plataforma web para poder visualizar los datos, por lo que se empezaron a realizar reuniones periódicas con personal de GeoVictoria para validar que las vistas fueran cómodas de usar.

En las primeras reuniones, se pedía al voluntario que revisara el apartado visual de las listas detalladas de alumnos y profesores, dado que debido a la falta de experiencia desarrollando interfaces, éstas no eran atractivas al usuario. Fue esto una de las críticas que se realizó en la primera iteración, específicamente, mencionando que ambos módulos se veían "toscos" dada la falta de una biblioteca visual que se encargara del CSS del sitio.

Este problema fue solucionado usando `Bootstrap` como framework visual. Por otra parte, se encontró el sitio "Muy blanco, vendría bien definir colores para enfocar la atención en lo que necesitas, y un logo para el inicio". Dichas tareas fueron dejadas de lado para revisar aspectos funcionales de los tres repositorios.

En la segunda reunión, se mostraron estas correcciones a ambos módulos, lo que fue

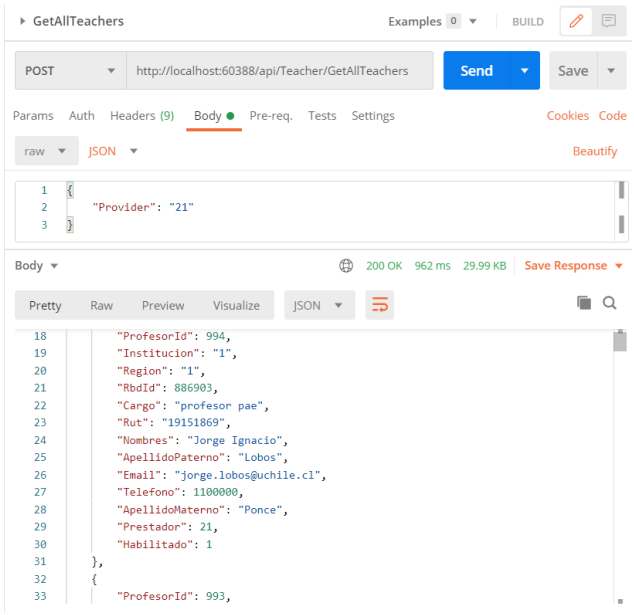


Figura 5.6: Prueba exitosa de obtención de profesores PAE vía JunaebRaw-Data.

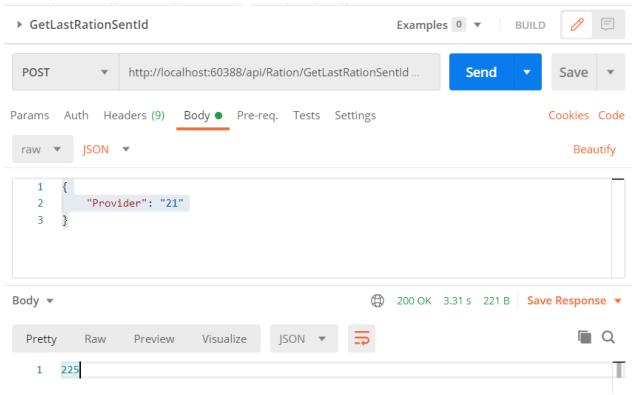


Figura 5.7: Prueba exitosa de obtención de última ración enviada vía JunaebRawData

percibido de mejor manera por el voluntario. Por otra parte, se trató la apariencia de la página principal y la herramienta de business analytics utilizada. El voluntario expresó que "los datos están bien, pero muy juntos, por lo que sería bueno mejorar la distribución de los espacios". Esto, pues se tenían cuatro gráficos juntos, quedando la cantidad de raciones por región como un extra. Se decidió combinar sus datos con la tabla de raciones entregadas por servicio, para formar la matriz de raciones por servicio entregadas por región.

En la última reunión, se discutieron aspectos más técnicos respecto al sitio, como mejorar la visibilidad, un sistema de identificación de ser necesario, y la implementación de cambios estructurales a los datos, para soportar un sistema de paginación.

Esto, para reducir los tiempos de carga en el sitio web, y mejorar la visibilidad de los elementos, pues en los tres módulos de detalles, en la carga inicial de los sitios `/Alumnos`, `/Profesores` y `/Raciones`, carga todos los datos de la sección que corresponde (por ejemplo,

Figura 5.8: Apariencia inicial de Listado de alumnos, sin estilos CSS.

todas las raciones). Ello implicaría problemas de escalabilidad en el caso de que el sitio web creciera.

Estas reuniones se realizaron principalmente para ver qué podría requerir un usuario de la plataforma. El personal de GeoVictoria cumplió el rol de este usuario con el rol de Administrador(a) en las historias de usuario descritas en el capítulo 3, por lo que se comprobaron las dificultades en el uso de la plataforma.

Finalmente, las dificultades surgidas a lo largo de este proceso de pruebas fue la falta de experiencia del memorista, tanto en el ámbito de front-end, como en el uso de `Microsoft Power BI`, por lo que a la vez que se arreglaban los problemas, el memorista seguía aprendiendo. Por otro lado, como se explicaba en la sección de IntegraciónJunaeb, dado que las comunicaciones fueron cortadas, se debieron usar otros métodos para llenar las bases de datos, generando datos propios.

Plataforma Junaeb Inicio Raciones Alumnos Profesores

Lista de alumnos

Buscar por RBD o Código de alumno: Filtrar

Código Alumno	Código RBD	Comuna	Región	Curso	Nivel	Año Escolar	
2261	8435	Coyhaique	Aysén	1-A	1	2019	Detalles
1	886903			1	4	2018	Detalles
7	886903	Lago Ranco	Los Ríos	1	4	2019	Detalles
11	886903	Penaflor	Metropolitana	1	5	2019	Detalles
14	886903			1	1	2020	Detalles
17	886903			1	6	2018	Detalles

Figura 5.9: Apariencia actual de Listado de alumnos, usando el framework Bootstrap para los estilos CSS.

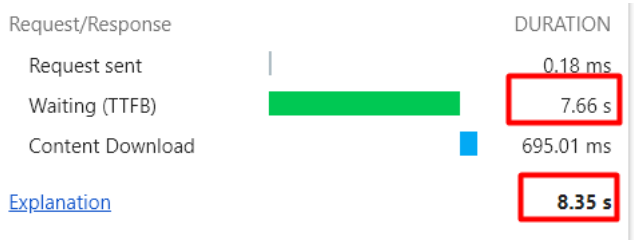


Figura 5.10: Tiempo de carga del listado general de raciones entregadas.

Capítulo 6

Conclusiones

Considerando la validación descrita en el Capítulo 5, en este capítulo se finaliza esta memoria haciendo un resumen del trabajo desarrollado, luego pasar a conclusiones técnicas y personales, para terminar con la labor futura a ejecutar en los componentes realizados en esta memoria.

6.1. Resumen del trabajo realizado

Esta sección detalla el trabajo efectuado por el memorista, el cumplimiento total, parcial o incumplimiento de los objetivos específicos mencionados la sección 1.4 del capítulo 1, y las causas y soluciones propuestas a los problemas que causaron aquello.

Como objetivo general, se planificó diseñar, construir y validar un sistema que centralice y facilite el análisis de las raciones repartidas a los alumnos mediante el programa PAE del JUNAEB. Este objetivo fue cumplido parcialmente, pues con el sistema creado, a pesar de ya estar los componentes diseñados y habiendo realizado las pruebas descritas en el capítulo anterior, faltan las pruebas conjuntas con esta entidad, poner en marcha la publicación en producción de los componentes, y realizar pruebas piloto del funcionamiento de los componentes en un entorno real.

Los conflictos de comunicación con JUNAEB se dieron debido al cese de comunicación de las entidades correspondientes, y el cierre de respuesta de los webservices. Luego de ello, se realizó la petición de reapertura, pero un mes después responden que se debe realizar petición al Consejo de Transparencia, quienes responden que dicha información no existe y que se debería destinar profesionales adicionales para solucionar el requerimiento, por lo que fue rechazado. En la sección de anexos, se incluye la información de los webservices pertinentes, y la petición realizada al Consejo de Transparencia, con su respectiva respuesta.

Respecto a los objetivos específicos, el primero de estos menciona la creación de las interfaces, modelo y API del sistema, lo que se tradujo en la planificación de los componentes IntegracionJunaeb, JunaebRawData y PlataformaJunaeb, además de la base de datos en la que es posible alojar la información de los alumnos, profesores PAE y raciones entregadas.

En términos de los alcances de esta memoria, este objetivo está cumplido, pero se deja la posibilidad de que en el futuro, se agreguen más funcionalidades, como las descritas en el capítulo 3 en la sección 3.3.

Luego, se mencionan las pruebas de las API de los reloj control como segundo objetivo específico. Este objetivo no fue cumplido, ya que luego de averiguar con el área comercial de GeoVictoria el procedimiento en el que cada empresa obtiene sus reloj control, se dio a conocer que cada empresa de alimentos puede, o comprar directamente a GeoVictoria los relojes con el software perteneciente a casino (que se muestran en el capítulo 1, figura 2.1a), o realizar una compra directa de los huelleros a la empresa ZKTeco y realizar el chequeo de software con GeoVictoria, por lo que habría que realizar pruebas dependientes de cada proveedor de alimentos. Además, el contexto de pandemia impidió cualquier reunión física para poder hacer pruebas a los equipos, sin arriesgar la integridad del memorista.

Para el ítem test de stress, este objetivo fue cumplido de forma parcial, dado que, como se mencionó en el capítulo anterior, no se pudo probar en conjunto a JUNAEB el ingreso o modificación de datos de alumnos y profesores PAE, o el envío de raciones al webservice habilitado para aquello. Dado esto, se tomó como medida de contingencia la creación de queries SQL que inyecten datos a las tablas correspondientes. La ejecución de estos scripts tomó menos de 5 segundos cada uno, por lo que en una estimación inicial, la escritura bruta de la información no supondría problema a una base de datos. Ello, sin perjuicio de que, al habilitar JUNAEB nuevamente las comunicaciones en algún momento, se puedan realizar pruebas más rigurosas. De esta manera, no solo se probaría la escritura en la base de datos, sino que también la estabilidad de los componentes, y de haber cambios, estos pueden volver a integrarse con los webservices de manera simple, limpiando la base de datos y usando los datos provistos por ellos para cargar los datos reales, implicando un bajo esfuerzo de actualización.

Luego, se menciona como objetivo la implementación de una aplicación que reciba como entrada los datos enviados por los reloj control al motor de base de datos, y que genere, automáticamente o a pedido, consultas o reportes con la información de las raciones repartidas. Este objetivo fue parcialmente cumplido, dado que a pesar de haber utilizado la herramienta de business analytics `Microsoft Power BI` para crear las queries de consulta y que se muestre como el reporte principal en la Landing Page de PlataformaJunaeb, estas no cuentan con filtros de cara al usuario que este pueda editar para realizar las distintas consultas, sino que se actualiza diariamente.

El último objetivo propuesto fue validar el trabajo realizado una vez cada 6 semanas con personal de Pruebas de GeoVictoria, revisando los tiempos de ejecución, los datos de entrada y la estabilidad del sistema. Dicho objetivo fue cumplido, dado que efectivamente, se realizaron tres reuniones, descritas en la sección dedicada a PlataformaJunaeb en el capítulo 5. En estas reuniones, el personal entregó feedback respecto a aspectos visuales, de estabilidad y de experiencia de usuario para mejorar la plataforma.

6.2. Conclusiones acerca del trabajo realizado

En esta sección, se realizarán conclusiones técnicas y personales acerca del trabajo desarrollado.

Respecto al desarrollo de los distintos componentes de la aplicación, se aplicó exitosamente el uso del patrón moderador para separar las responsabilidades de comunicación con los webservices externos, consultas e inyección a la base de datos, y la plataforma visual. Dicho patrón permitió desacoplar los componentes y a pesar de que no se previeron los problemas ocurridos con JUNAEB, el diseño escogido fue el adecuado para enfrentar de mejor manera dicha eventualidad, permitiendo la continuación del proyecto de memoria.

El mantener tres repositorios distintos puede parecer una tarea compleja en un inicio, pero el que estuvieran separados por responsabilidades permitió que los cambios producidos simplificaran el desarrollo sobre ellos, al darse un desarrollo más focalizado respecto a las distintas funcionalidades propuestas al inicio de este proceso de memoria. Además, en caso de realizar modificaciones en el futuro, los cambios quedarán más localizados: al querer realizar un cambio solamente en la capa visual, únicamente se tendrá que cargar el proyecto PlataformaJunaeb. De querer agregar nuevas funcionalidades JUNAEB al momento de retomar las comunicaciones, se deberá modificar solamente el componente IntegracionJunaeb, en caso de usar los mismos datos. En caso contrario, de necesitar nuevos tipos de datos, se pueden crear nuevos endpoints y/o controladores a la API JunaebRawData. En todos los casos, no sería necesario ejecutar componentes que no estén relacionados a las pruebas a realizar.

Por otra parte, las dificultades surgidas producen nuevos aprendizajes, planteando la pregunta *¿Qué implica crear datos realistas para este proyecto de memoria?* Esto, respecto a la creación de datos dummy para llenar la base de datos y poder probar los componentes ante la falta de datos provistos por JUNAEB. Primero, la respuesta más sencilla es usar datos que parezcan pertenecientes a los usuarios que alimentan estos datos, pero ¿Qué restricciones tienen dichos datos? ¿Cuáles casos son más probables que se den, y cuáles se descartan? y ¿Es necesario generar datos completamente reales, o tengo un objetivo detrás de ello? son preguntas que iban surgiendo a lo largo del proceso de generación de la data dummy. Tomando en cuenta que los principales objetivos de dichos datos eran demostrar el comportamiento de la plataforma web al personal voluntario de GeoVictoria, y que existieran pruebas de soporte de los datos, se debía realizar un proceso extenso de creación de datos, pero lo suficientemente adaptable como para que no impactara en el proceso de desarrollo de los componentes.

Dado esto, se tomó en cuenta que los datos debían *parecer* realistas, pero sin mucha profundidad, cosa que los conceptos a mostrar en cada uno de los apartados fueran los correctos. Los nombres no debían ser "Usuario 1" o similares, sino nombres de personas reales, y el volumen de los datos de prueba debía ser similar a las estimaciones realizadas al inicio del proceso de memoria, lo que concluyó en la generación de poco más de 2.500 campos de raciones, simulando alumnos firmando con su huella digital la entrega de raciones.

Por otra parte, personalmente logré un aprendizaje en el área de front-end con esta memoria, dado que la gran mayoría de mis experiencias han sido hasta el momento en back-end. Esto me permitió aprender la complejidad del trabajo con usuarios. Al probar una plataforma, hay varios aspectos que se podrían haber mejorado: Haber aumentado la periodicidad de

las reuniones con tal que los voluntarios no perdieran la costumbre de usar la plataforma y reaprender el proceso cada seis semanas, el tener un plan de pruebas que se ajustara de mejor manera a cada etapa del desarrollo, o definir tópicos de antemano, son elementos básicos ante estas situaciones, pero que en definitiva la falta de experiencia mermó.

Por último, aunque `Microsoft Power BI` es una herramienta de business analytics que permite una rápida visualización de cara al usuario, tener flexibilidad respecto a la creación de consultas y ser multiplataforma, al ser una herramienta privativa [3], el alojar el servicio en la nube tiene un costo asociado para que se pueda crear un enlace al reporte dentro de un `iframe`. Por otro lado, la experiencia de este proceso completo es algo nuevo, siendo que los conocimientos previos que se tenían respecto al desarrollo de reportes era simplemente la creación, para luego traspasarlos a otros desarrolladores encargados de la publicación. Esta memoria permitió descubrir que la curva de aprendizaje de esta herramienta es lo suficientemente elevada para incluso requerir certificaciones [2] en algunos empleos.

6.3. Trabajo futuro

Respecto al trabajo futuro a realizar, uno de los aspectos más importantes es ejecutar los tests de integración con los webservices de JUNAEB en cuanto se vuelva a tener acceso a estos. Por otra parte, para mejorar aún más la escalabilidad de los datos, se puede generalizar la lógica detrás de estos para estandarizar el tipo de objetos recibidos.

Otro de los aspectos a tratar con JUNAEB son la realización de pruebas de usabilidad del sitio. A pesar de que el personal de Pruebas de GeoVictoria tiene conocimientos de como testear un sitio web, no son el usuario final a quien va dirigido el sitio, sino solamente fueron voluntarios.

Por último, en el capítulo 5 se describieron las falencias de usabilidad en la plataforma: Debido a que la cantidad de datos es estática, hay que pensar un esquema de carga dinámica de datos, porque la cantidad irá creciendo diariamente, lo que puede traer problemas como tiempos de cargas elevados, visualización de datos que no se tenían esperados, y uso de recursos adicionales en la aplicación. Una de las soluciones propuestas es crear un objeto que encapsule un conjunto de datos y realice paginación de estos.

Bibliografía

[1] Mediator design pattern, 1 2012. URL https://web.archive.org/web/20201002133951/https://sourcemaking.com/design_patterns/mediator.

[2] Browse certifications and exams - power bi, 9 2015. URL https://docs.microsoft.com/en-us/learn/certifications/browse/?expanded=power-platform&products=power-bi.

[3] Términos de licencia del software de microsoft microsoft power bi para windows, 9 2015. URL https://powerbi.microsoft.com/es-es/windows-license-terms/.

[4] .net framework 4.7.1, 12 2017. URL https://dotnet.microsoft.com/download/dotnet-framework/net471.

[5] Microsoft.aspnet.razor 3.2.7, 11 2018. URL https://www.nuget.org/packages/Microsoft.AspNet.Razor/.

[6] Postman, the collaboration platform for api development, 9 2019. URL https://www.postman.com/.

[7] Kestrel web server implementation in asp.net core, 5 2020. URL https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-3.1.

[8] What is power bi?, 9 2020. URL https://docs.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview.

[9] Cristóbal. Secretario General JUNAEB 2017 Acevedo Ferrer. Aplicación del acta de la supervisión de la gestión operativa y propuesta técnica del pae para la licitación id 85-10-lp14, 3 2017. URL https://www.junaeb.cl/wp-content/uploads/2017/07/REX-481-Aprueba-IT-Lic.-1014-2017-rev-02.pdf.

[10] Alibaba Group. Fr402 alibaba gold supplier zkteco high resolution biometric time attendance system, 2019. URL https://www.alibaba.com/product-detail/FR402-Alibaba-Gold-Supplier-ZKteco-High_60390458718.html?spm=a2700.7724838.2017115.13.564711237wzp4Zf.

[11] BadgePass. Campus card systems, 2019. URL https://www.badgepass.com/products-and-solutions/campus-card-systems.

[12] BadgePass. Badgepass shop id systems, 2019. URL https://www.badgepass.com/shop-online/shop-id-systems.

[13] Capture Technologies. Capture technologies k-12 education, 2019. URL https://web.archive.org/web/20190119103831/https://capturet.com/markets/k-12-education/.

[14] Unidad de licitaciones JUNAEB. Suministro de raciones alimenticias, 6 2018. URL

http://www.mercadopublico.cl/Procurement/Modules/RFB/DetailsAcquisition.aspx?qs=H/izg++HE8RdlaU7SUDcbA==.

[15] Fulcrum Biometrics. Irishield-usb bk2121u iris camera, 2020. URL https://www.fulcrumbiometrics.com/IriShield-USB-BK-2121U-p/102152.htm.

[16] Andrew Hudson. U maryland uses touchless biometrics for dining hall access, 1 2019. URL https://www.cr80news.com/news-item/u-maryland-uses-touchless-biometrics-for-dining-hall-access/.

[17] IdentiSys. Complete campus onecard operations solution, 2019. URL https://www.identisys.com/markets-served/higher-education/campus-onecard-system.

[18] JUNAEB. JUNAEB programa de alimentación escolar (pae), 12 2014. URL https://www.junaeb.cl/programa-de-alimentacion-escolar.

[19] JUNAEB. JUNAEB ¿Cómo funciona el SINAE?, 3 2015. URL http://web.archive.org/web/20201114194601/https://www.junaeb.cl/como-funciona-el-sinae.

[20] Francisco Mardones, Cruz Bustamante, Jorge Penjeam, and Pedro Vega. Informe final de evaluación, programa de alimentación escolar, ministerio de educación, junaeb, 8 1997. URL https://www.dipres.gob.cl/597/articles-140908_informe_final.pdf.

[21] Ministerio de Ciencia, Tecnología, Conocimiento e Innovación. Informacioncomunas.csv, 2020. URL https://github.com/MinCiencia/Datos-COVID19/blob/master/input/Otros/InformacionComunas.csv.

[22] L. O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003. doi: http://dx.doi.org/10.1109/JPROC.2003.819611.

[23] Subsecretaría de Educación Superior, Ministerio de Educación. Beca de alimentaciÓn (baes), 4 2017. URL http://portal.beneficiosestudiantiles.cl/becas-y-creditos/beca-de-alimentacion-baes.

[24] Jaime Tohá. Oficio fiscalización respuesta licitación junaeb 85-27-lr18, 3 2019. URL https://www.camara.cl/verDoc.aspx?prmTIPO=OFICIOFISCALIZACIONRESPUESTA&prmID=76213&prmNUMERO=295&prmRTE=0.

[25] Marcelo. DIPRES Villena. Evaluación de impacto de los programas de alimentación de la junaeb, del ministerio de educación, 12 2013. URL https://www.dipres.gob.cl/597/articles-141186_informe_final.pdf.

[26] Genevieve Warren, Gordon Hogenson, Saisang Cai, Mike Jones, Jennifer Kirsch, Rick van den Bosch, Mike Jacobs, Mark Seemann, David Coulter, and Colin Robertson. Code metrics values, 11 2018. URL https://docs.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2019.

Anexo A

Anexos

A.1. Webservices de JUNAEB

A continuación se incluyen los documentos que detallan cómo funcionan los webservices de JUNAEB. Primero, en las páginas 67 y la mitad izquierda de la página 68, se incluye el servicio de inserción de raciones, que permite a JUNAEB recibir estos datos para que tengan registros de las raciones diarias consumidas por los alumnos.

Luego, en la mitad derecha de la página 68 y la página 69, se agrega el documento que contiene las instrucciones para llamar al webservice de matrículas. Este webservice tiene los datos de los alumnos pertenecientes al PAE.

Finalmente, en la página 70 se lista la información del webservice de profesores responsables.

Implementación Servicio de Inserción de Raciones RBD:

- https://serviciobio-test.junaeb.cl/RESTfulBiometrico/json/buscarFocalizado/setListaFocaCerti

Content-Type: application/json
Method HTTP: POST
Authorization Type: Basic Auth
(Las credenciales fueron enviadas en correos previos. Estas se mantienen sin modificaciones)

Listado de campos solicitados:

ENVÍO RACIÓN, SUJETA A VALIDACIÓN POR NORMAS DE DUPLICIDAD, CAMPOS NULOS:


```
{
        "institución":  1,
        "prestador":  21,
        "region":  7,
        "comunaId": 127,
        "rbdId": 886903,
        "codAlumno": 9999,
        "nivel": 1,
        "fechaCerti": "2018-07-03T00:00:00.000Z",
        "servicioCerti": 1,
        "hora": "2018-07-03T13:10:25.000Z",
        "tipoCerti": 1,
        "observación": "---Observaciones---"
}
```


Descripción (Todos los campos son obligatorios):

- **institución**: 1 JUNAEB / 2 JUNJI / 3 INTEGRA
- **prestador**: código particular para cada proveedor
- **región**: Número de Región (Formato Entero)
- **comunaId**: Código Identificatorio Comuna (Formato Entero)
- **rbdId**: Código identificatorio del Establecimiento asociado a la entrega de Raciones
- **codAlumno**: Código identificador del Alumno
- **nivel**: 1 Básica / 2 Media / 3 Kinder / 4 Pre-Kinder / 5 Adulto / 6 Especial
- **fechaCerti**: Fecha Certificación, puede incluir hora.
- **servicioCerti**: código identificatorio del servicio o ración entregada (1 Desayuno / 2 Almuerzo / 3 Once / 4 Cena / 5 Colación / 6 Tercer Servicio)
- **hora**: Fecha Certificación, incluye hora de entrega de la ración.
- **tipoCerti**: 1 - ingreso vía Web Services
- **observación**: comentarios adicionales en la entrega

JUNAEB | Junta Nacional de Auxilio Escolar y Becas

Respuesta EXITOSA (RACIÓN INSERTADA):


```
{
   "codError": 0,
   " codAlumno ": 9999,
   "obsError": []
}
```


Respuesta DETECCIÓN CAMPO NULO (RACIÓN NO INSERTADA)

ObsError: Arreglo con los campos nulos detectados.


```
{
    "codError": -1,
    " codAlumno ": 9999,
    "obsError": [
          {
             "obs": "INSTITUCIÓN"
          },
          {
             "obs": "NIVEL"
          }
    ]
}
```


Respuesta RACIÓN DUPLICADA (RACIÓN NO INSERTADA):

Cabe destacar que se contempla como "Envío duplicado" aquel envío en el cual, TODOS sus campos ya se encuentran registrados en la Base de Datos, solo basta con modificar un campo del total de campos solicitados para que este ingreso no cuente como duplicado.


```
{
      "codError": 1,
      " codAlumno ": 9999,
      "obsError": [
         {
            "obs": "RACIÓN DUPLICADA"
         }
      ]
}
```

Respuesta AUTENTICACIÓN NO VÁLIDA (RACIÓN NO INSERTADA):


```
{
    "codError": -2,
    " codAlumno ": 0,
    "obsError": [
      {
        "obs": "LOGIN INVALIDO"
      }
    ]
}

{
    "codError": -2,
    " codAlumno ": 0,
    "obsError": [
      {
        "obs": "CREDENCIALES NO ENVIADAS"
      }
    ]
}
```


Códigos de Error:

- -1 = Envío con campo(s) nulo(s)
- 0 = Ingreso sin Observaciones
- 1 = Envío duplicado
- -2 = autentificación no válida

JUNAEB | Junta Nacional de Auxilio Escolar y Becas

Implementación Servicio de Obtención de Matrículas RBD

- https://serviciobio-test.junaeb.cl/RESTfulBiometrico/json/buscarFocalizado/obtenerAutorizados/{codPrestador}

Content-Type: application/json
Authorization Type: Basic Auth
Method HTTP: GET

(Las credenciales fueron enviadas en correos previos. Estos se mantienen sin modificaciones)

REQUEST (HEADERS):

- Authorization: Basic (Uso de credenciales)
- Content-Type: application/json

RESPONSE:

Array de Datos Ejemplo:


```
[{
    "codAlumno": 15572550,
    "codRbd": 6770,
    "codComuna": 14101,
    "codRegion": 14,
    "licitacion": 3715,
    "curso": 1,
    "letraCurso": "A",
    "nivel": 1,
    "anhoEscolar": 2018,
    "autorizacion": 0,
    "fechaAutorizacion": "20/07/2019 15:51:52 PM",
    "serDesayuno": 1,
    "serAlmuerzo": 1,
    "serOnce": 0,
    "serCena": 0,
    "serColacion": 0,
    "serAlmuerzoViernes": 1,
    "rbdFicticio": 9999,
    "mes": 8,
    "anho": 2018,
    "institucion": 1,
    "alumnoFocalizado": 1
}]
```

Descripción y Tipo de Campos:

- **codAlumno** (NUMÉRICO): Código identificador del Alumno.
- **codRbd** (NUMÉRICO): Código identificador del Establecimiento.
- **codComuna** (NUMÉRICO): Código identificador de la Comuna de ubicación del Establecimiento.
- **codRegion** (NUMÉRICO): Código identificador de la Región de ubicación del Establecimiento.
- **licitacion** (NUMÉRICO): Número de Licitación activa.
- **curso** (NUMÉRICO): Numeral identificatorio del Curso del Alumno.
- **letraCurso** (TEXTO): Letra identificatoria del Curso del Alumno.
- **nivel** (NUMÉRICO): Código identificador del Nivel Educacional activo del Alumno.
- **anhoEscolar** (NUMÉRICO): Año escolar vigente.
- **autorizacion** (NUMÉRICO): Identificador de Autorización de Alimentación Apoderado.
- **fechaAutorizacion** (TEXTO): Fecha de registro autorización Apoderado.
- **serDesayuno** (NUMÉRICO): Estado Servicio de Desayuno del Alumno.
- **serAlmuerzo** (NUMÉRICO): Estado Servicio de Almuerzo del Alumno.
- **serOnce** (NUMÉRICO): Estado Servicio de Once del Alumno.
- **serCena** (NUMÉRICO): Estado Servicio de Cena del Alumno.
- **serColacion** (NUMÉRICO): Estado Servicio de Colación del Alumno.
- **serAlmuerzoViernes** (NUMÉRICO): Estado Servicio de Almuerzo Viernes del Alumno.
- **rbdFicticio** (NUMÉRICO): Código de RBD ficticio
- **mes** (NUMÉRICO): Mes de última carga Planilla SIGE
- **anho** (NUMÉRICO): Año de última carga Planilla SIGE
- **institución** (NUMÉRICO): Código de Institución asociada al RBD
- **alumnoFocalizado** (NUMÉRICO): Identificador si el Alumno es Focalizado según SIGE.

JUNAEB | Junta Nacional de Auxilio Escolar y Becas

Representación de Códigos:

Nivel:

1. Básica
2. Media
3. Kinder
4. Pre-Kinder
5. Adulto
6. Especial

Autorización:

0. No Autorizado por Apoderado
1. Autorizado por Apoderado

Servicios (ser):

0. No considerado en el Servicio.
1. Considerado en el Servicio.

Institución:

1.- JUNAEB
3.- JUNJI
4.- INTEGRA

Alumno Focalizado:

0. Alumno NO Focalizado.
1. Alumno Focalizado.

Implementación Servicio de Profesores Responsables:

- https://serviciobio-test.junaeb.cl/RESTfulBiometrico/json/buscarFocalizado/obtenerProfesores/getProfesorResponsableBio/{codPrestador}

Content-Type: application/json
Method HTTP: GET
Authorization Type: Basic Auth

(Las credenciales fueron enviadas en correos previos. Estas se mantienen sin modificaciones)

REQUEST (HEADERS):

- Authorization: Basic (Uso de credenciales)
- Content-Type: application/json

RESPONSE:

Array de Datos Ejemplo:


```
[
  {
    "profesorId": 995,
    "institucion": "1",
    "region": "8",
    "rbdId": 4814,
    "cargo": "profesor pae",
    "rut": "6665939",
    "nombres": "Marlene Myriam",
    "apellidoPaterno": "Yañez",
    "email": "myanezz@hotmail.com",
    "telefono": 1000000,
    "apellidoMaterno": "Zuñiga"
  }
]
```


JUNAEB | Junta Nacional de Auxilio Escolar y Becas

Descripción y Tipo de Campos:

- **profesorId** (NUMÉRICO): Código identificador del Profesor PAE.
- **institucion** (TEXTO): Código identificador de la Institución asociada.
- **region** (TEXTO): Código identificador de la Región de ubicación del RBD.
- **rbdId** (NUMÉRICO): Código identificador del Establecimiento.
- **cargo** (TEXTO): Profesor PAE.
- **rut** (TEXTO): Rol único Nacional del Profesor PAE.
- **nombres** (TEXTO): Nombre Profesor PAE.
- **apellidoPaterno** (TEXTO): Apellido Paterno Prof. PAE.
- **email** (TEXTO): Dirección electrónica Prof. PAE.
- **telefono** (NUMÉRICO): Fono de contacto Prof. PAE.
- **apellidoMaterno** (TEXTO): Apellido Materno Prof. PAE.

Representación de Códigos:

Institución:

1.- JUNAEB
3.- JUNJI
4.- INTEGRA

A.2. Petición a Consejo de Transparencia para habilitación de webservices y denegación

En este anexo, se incluyen los documentos relativos al proceso de comunicación con el Consejo de Transparencia para la reapertura de los webservices de JUNAEB, luego de que el acceso a los datos fuera negado.

En el primer documento, en la página 72, se muestra el acuso de recibo de la petición. En este acuso, se detalla el nombre del responsable, dirección de correo electrónico, la solicitud propiamente tal, y las observaciones relativas a la petición.

Finalmente, entre las páginas 73 y 75, se detalla lo relativo a la denegación de reapertura de estos servicio.

ACUSE DE RECIBO DE SOLICITUD DE ACCESO A LA INFORMACIÓN

LEY DE TRANSPARENCIA

AJ009T0002074

Fecha: 27/05/2020 Hora: 17:26:29

Portal de Transparencia – Ingreso Electrónico – Solicitud de Información

1. Contenido de la Solicitud

Nombre y apellidos o razón social:	Jorge Lobos Ponce
Tipo de persona:	Natural
Dirección postal y/o correo electrónico:	jorge.lobos@ug.uchile.cl
Nombre de apoderado (si corresponde):	
Solicitud realizada:	Buenas tardes: Mi nombre es Jorge Lobos, soy memorista del Departamento de Ciencias de la Computación de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. Actualmente estoy en un proceso de memoria que requiere información que JUNAEB puede disponibilizar, con el objetivo de realizar un mejor catastro del destino de las raciones entregadas por ustedes, en base a licitaciones públicas realizadas por ustedes en base a requerimientos de sistemas biométricos para llevar registro. Actualmente, mi propuesta de memoria se llama "IMPLEMENTACIÓN DE SISTEMA DE VALIDACIÓN Y REPORTE PARA EL REGISTRO DE ENTREGAS DE RACIONES PAE DE JUNAEB" a cargo de la profesora Jocelyn Simmonds y dado que va a ser una memoria pública, el software finalizado estaría a su disposición. Lo necesario para poder probar las visualización de datos de la plataforma de reporte es una muestra de datos de -Alumnos -Profesores PAE -Raciones Con la que alimentar la base de datos propia, que presenten ustedes mediante alguna de las siguientes opciones: -Plataforma -Documento (.csv, .xlsx u otro) -Endpoint de API test (Web Service) De esa manera, poder verificar que todo funcione de buena manera. Por ello, quisiera pedir una muestra de datos de test. Muchísimas gracias de antemano, quedo atento a su respuesta
Observaciones:	El caso ideal de entrega de información serían WebServices que contengan los siguientes formatos de entrega (GET) o recepcion (POST): - Servicio GET Alumnos (JUNAEB entrega): Arreglo de datos de todos los alumnos, cada uno conteniendo la siguiente información: ▪ identificadorAlumno: Alguna forma de identificar anónimamente a un alumno (integer en base a un id anónimo, o string en base a un hash) ▪ institución: Institución a cargo del programa PAE del establecimiento: Integra, JUNAEB, JUNJI u otro (integer mediante un código o string del nombre de la institución) ▪ rbd: Rol Base Datos del Establecimiento (integer) ▪ región: Código Único Territorial de la región (integer) ▪ comuna: Código Único Territorial de la comuna (integer) ▪ curso: Curso del Alumno (integer, por ejemplo 4 para cuarto). ▪ letraCurso: Letra del Curso del Alumno (string, vacío o nulo si no presenta letra). ▪ nivelEducacional: Pre-Kinder, Kinder, Básica, Media, Adulto, Especial (string, o integer mediante un código) ▪ anho: Año escolar vigente (integer o string). ▪ fechaAutorizacion: Fecha de autorización Apoderado para que alumno presente registro biométrico. (string) ▪ servicioDesayuno: Indica si alumno recibe Desayuno. (integer o boolean) ▪ servicioAlmuerzo: Indica si alumno recibe Almuerzo. (integer o boolean) ▪ servicioOnce: Indica si alumno recibe Once. (integer o boolean) ▪ servicioCena: Indica si alumno recibe Cena. (integer o boolean) ▪ servicioColacion: Indica si alumno recibe Colación. (integer o boolean) ▪ servicioOtro: Indica si alumno recibe algún otro servicio. (integer o boolean) ▪ empresaPrestador: Empresa que presta servicio de alimentación (string o integer). -Servicio GET Profesores PAE (JUNAEB entrega): Arreglo de datos de todos los profesores PAE, cada uno conteniendo la siguiente información: ▪ identificadorProfesor: Alguna forma de identificar a un profesor PAE único (pude ser integer en base a un id anónimo o string en base a un hash) ▪ institución: Institución a cargo del programa PAE del establecimiento: Integra, JUNAEB, JUNJI u otro (integer mediante un código o string del nombre de la institución) ▪ rbd: Rol Base Datos del Establecimiento (integer)

▪ región: Código Único Territorial de la región (integer)
▪ comuna: Código Único Territorial de la comuna (integer)
▪ rut: Rol único Nacional del Profesor PAE. (string)
▪ nombres: Nombre(s) del Profesor PAE. (string)
▪ apellidoPaterno (TEXTO): Apellido Paterno Prof. PAE. (string)
▪ apellidoMaterno: Apellido Materno del Prof. PAE. (string)
▪ email: Correo electrónico Prof. PAE. (string, vacío o nulo en caso de no existir)
▪ telefono: Fono de contacto del Prof. PAE. (string)
- Servicio POST Raciones (JUNAEB recibe):
▪ institución: Institución a cargo del programa PAE del establecimiento: Integra, JUNAEB, JUNJI u otro (integer mediante un código o string del nombre de la institución)
▪ región: Código Único Territorial de la región (integer)
▪ comuna: Código Único Territorial de la comuna (integer)
▪ rbd: Rol Base Datos del Establecimiento (integer)
▪ identificadorAlumno: Alguna forma de identificar anónimamente a un alumno (integer en base a un id anónimo, o string en base a un hash)
▪ nivelEducacional: Pre-Kinder, Kinder, Básica, Media, Adulto, Especial (string, o integer mediante un código)
▪ tipoRación: Desayuno, almuerzo, once, cena, colación, otro. (string o integer, con su respectivo código)
▪ fechaRacion: Fecha y hora de entrega de ración (string).
▪ empresaPrestador: Empresa que presta servicio de alimentación (string o integer).

Archivos adjuntos:	
Medio de envío o retiro de la información:	Correo electrónico
Formato de entrega de la información:	PDF
Sesión iniciada en Portal:	NO
Vía de ingreso en el organismo:	Vía electrónica

De acuerdo a su requerimiento, este organismo procederá a verificar lo siguiente:

a) Si su presentación constituye una solicitud de información.
b) Si nuestra institución es competente para dar respuesta a ésta.
c) Si su solicitud cumple con los requisitos obligatorios establecidos en el artículo 12 de la Ley de Transparencia.

2. Fecha de entrega vence el: 24/06/2020

El plazo máximo para responder una solicitud de información es de veinte (20) días hábiles. De acuerdo a su presentación la fecha máxima de entrega de la respuesta es el día **24/06/2020**. Se informa además que excepcionalmente el plazo referido podrá ser prorrogado por otros 10 días hábiles, cuando existan circunstancias que hagan difícil reunir la información solicitada, conforme lo dispone el artículo 14 de la Ley de Transparencia.

Informamos además que la entrega de información eventualmente podrá estar condicionada al cobro de los costos directos de reproducción. Por su parte, y de acuerdo a lo establecido en el artículo 18 de la Ley de Transparencia, el no pago de tales costos suspende la entrega de la información requerida.

En caso que su solicitud de información no sea respondida en el plazo de veinte (20) días hábiles, o sea ésta denegada o bien la respuesta sea incompleta o no corresponda a lo solicitado, en aquellos casos que la ley lo permite usted podrá interponer un reclamo por denegación de información ante el Consejo para la Transparencia www.consejotransparencia.cl dentro del plazo de 15 días hábiles, contado desde la notificación de la denegación de acceso a la información, o desde que haya expirado el plazo definido para dar respuesta.

3. Seguimiento de la solicitud

Con este código de solicitud: **AJ009T0002074**, podrá hacer seguimiento a su solicitud de acceso a través de los siguientes medios:

a) Directamente llamando al teléfono del organismo: 22 630 0521
b) Consultando presencialmente, en oficinas del organismo "Junta Nacional de Auxilio Escolar y Becas (JUNAEB)", ubicadas en Monjitas N° 565, piso 6, Santiago, en el horario Lunes a viernes de 08:30 a 17:30 hrs.
c) Digitando código de solicitud en www.portaltransparencia.cl opción 'Hacer seguimiento a solicitudes'

4. Eventual subsanación

Si su solicitud de información no cumple con todos los requisitos señalados en el artículo 12 de la Ley de Transparencia, se le solicitará la subsanación o corrección de la misma, para lo cual tendrá un plazo máximo de cinco (5) días hábiles contados desde la notificación del requerimiento de subsanación. En caso que usted no responda a esta subsanación dentro del plazo señalado, se le tendrá por desistido de su petición.

GOBIERNO DE CHILE
JUNTA NACIONAL DE AUXILIO
ESCOLAR Y BECAS

DENIEGA SOLICITUD DE ACCESO A INFORMACION QUE INDICA Y ORDENA PUBLICACION DEL ACTO ADMINISTRATIVO QUE LO DISPONE.

RESOLUCIÓN EXENTA 1740

SANTIAGO, 17 JUN 2020

V I S T O:

Lo dispuesto en el artículo 8°, inciso segundo de la Constitución Política de la República; en el Decreto con Fuerza de Ley N° 1-19653 de 2001 del Ministerio Secretaría General de la Presidencia, que fija el texto refundido, coordinado y sistematizado de la Ley N° 18.575 de Bases de la Administración del Estado, en la Ley N° 19.880 de Bases de los Procedimientos Administrativos que Rigen los Actos de los Órganos de la Administración del Estado, en la Ley N° 20.285, sobre Acceso a la Información Pública y su Reglamento, aprobado por Decreto N° 13 de 2009, del Ministerio Secretaría General de la Presidencia, en la Instrucción General N°3, del 2009, Sobre el Índice de Actos y Documentos Calificados Como Secretos o Reservados del Consejo Para La Transparencia; en la Ley N° 15.720, que crea la Junta Nacional de Auxilio Escolar y Becas; en el Decreto Supremo N° 5.311 de 1968 del Ministerio de Educación, que fija el Reglamento General de la Junta Nacional de Auxilio Escolar y Becas; en el Decreto Ley Nº 180 de 1973 que declara en receso el Consejo de JUNAEB cuyas facultades otorga a su Secretario General; en el Decreto Supremo N° 5 de 2018 del Ministerio de Educación que designa a don Jaime Tohá Lavanderos en calidad de Secretario General de JUNAEB, en las Resoluciones N.º 7 y 8, de 2019, ambas de la Contraloría General de la República, que imparte normas sobre exención del trámite de Toma de Razón y determinación de los montos en unidades tributarias mensuales, a partir de las cuales los actos que se individualizan quedarán sujetos a Toma de Razón y a Controles de Reemplazo cuando corresponda, y en lo requerido por don Jorge Lobos Ponce, a través de solicitud de acceso a la información pública, Nº AJ009T-00002074 de 27 de mayo de 2020.

CONSIDERANDO:

1.- Que, con fecha 27 de mayo de 2020, por medio de la Solicitud de Acceso a la Información Pública Nº AJ009T-00002074 efectuada electrónicamente a través del Portal de Transparencia del Estado, don Jorge Lobos Ponce requirió lo siguiente:

"Mi nombre es Jorge Lobos, soy memorista del Departamento de Ciencias de la Computación de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile.
Actualmente estoy en un proceso de memoria que requiere información que JUNAEB puede disponibilizar, con el objetivo de realizar un mejor catastro del destino de las raciones entregadas por ustedes, en base a licitaciones públicas realizadas por ustedes en base a requerimientos de sistemas biométricos para llevar registro.
Actualmente, mi propuesta de memoria se llama "IMPLEMENTACIÓN DE SISTEMA DE VALIDACIÓN Y REPORTE PARA EL REGISTRO DE ENTREGAS DE RACIONES PAE DE JUNAEB" a cargo de la profesora Jocelyn Simmonds y dado que va a ser una memoria pública, el software finalizado estaría a su disposición.
Lo necesario para poder probar las visualización de datos de la plataforma de reporte es una muestra de datos de -Alumnos -Profesores PAE -Raciones Con la que alimentar la base de datos propia, que presenten ustedes mediante alguna de las siguientes opciones: -Plataforma -Documento (.csv, .xlsx u otro) -Endpoint de API test (Web Service)
De esa manera, poder verificar que todo funcione de buena manera. Por ello, quisiera pedir una muestra de datos de test. Muchísimas gracias de antemano, quedo atento a su respuesta"
Observaciones
"El caso ideal de entrega de información serían WebServices que contengan los siguientes formatos de entrega (GET) o recepción (POST): - Servicio GET Alumnos (JUNAEB entrega): Arreglo de datos de todos los alumnos, cada uno conteniendo la siguiente información: ▪ identificador Alumno: Alguna forma de identificar anónimamente a un alumno (integer en base a un id anónimo, o string en base a un hash) ▪ institución: Institución a cargo del programa PAE del establecimiento: Integra, JUNAEB, JUNJI u otro (integer mediante un código o string del nombre de la institución) ▪ rbd: Rol Base Datos del Establecimiento (integer) ▪ región: Código Único Territorial de la región (integer) ▪ comuna: Código Único Territorial de la comuna (integer) ▪ curso: Curso del Alumno (integer, por ejemplo 4 para cuarto). ▪ Letra Curso: Letra del Curso del Alumno (string, vacío o nulo si no presenta letra)
Nivel educacional Pre kínder, Kínder, Básica, Media, Adulto, Especial (string, o integer mediante un código) ▪ año: Año escolar vigente (integer o string). ▪ fecha Autorización: Fecha de autorización Apoderado para que alumno presente registro biométrico. (String) ▪ servicio Desayuno: Indica si alumno recibe Desayuno. (Integer o boolean) ▪ servicio Almuerzo: Indica

si alumno recibe Almuerzo. (Integer o boolean) ▪ *servicio Once: Indica si alumno recibe Once. (Integer o boolean)* ▪ *servicio Cena: Indica si alumno recibe Cena. (Integer o boolean)* ▪ *servicio Colación: Indica si alumno recibe Colación. (Integer o boolean)* ▪ *servicio Otro: Indica si alumno recibe algún otro servicio. (Integer o boolean)* ▪ *Empresa Prestador: Empresa que presta servicio de alimentación (string o integer).*

Servicio GET Profesores PAE (JUNAEB entrega): Arreglo de datos de todos los profesores PAE, cada uno conteniendo la siguiente información: ▪ identificador Profesor: Alguna forma de identificar a un profesor PAE único (pude ser integer en base a un id anónimo o string en base a un hash) ▪ institución: Institución a cargo del programa PAE del establecimiento: Integra, JUNAEB, JUNJI u otro (integer mediante un código o string del nombre de la institución) ▪ rbd: Rol Base Datos del Establecimiento (integer) ▪ región: Código Único Territorial de la región (integer) ▪ comuna: Código Único Territorial de la comuna (integer) ▪ Rut: Rol único Nacional del Profesor PAE. (String) ▪ nombres: Nombre(s) del Profesor PAE. (String) ▪ apellido Paterno (TEXTO): Apellido Paterno Prof. PAE. (String) ▪ apellido Materno: Apellido Materno del Prof. PAE. (String) ▪ Email: Correo electrónico Prof. PAE. (String, vacío o nulo en caso de no existir) ▪ Teléfono: Fono de contacto del Prof. PAE. (String) - Servicio POST Raciones (JUNAEB recibe): ▪ institución: Institución a cargo del programa PAE del establecimiento: Integra, JUNAEB, JUNJI u otro (integer mediante un código o string del nombre de la institución) ▪ región: Código Único Territorial de la región (integer) ▪ comuna: Código Único Territorial de la comuna (integer) ▪ rbd: Rol Base Datos del Establecimiento (integer) ▪ identificador Alumno: Alguna forma de identificar anónimamente a un alumno (integer en base a un id anónimo, o string en base a un hash) ▪ nivel Educacional: Pre-Kínder, Kínder, Básica, Media, Adulto, Especial (string, o integer mediante un código) ▪ tipo Ración: Desayuno, almuerzo, once, cena, colación, otro. (String o integer, con su respectivo código) ▪ fecha Ración: Fecha y hora de entrega de ración (string). ▪ Empresa Prestador: Empresa que presta servicio de alimentación (string o integer)"

2.- Que, el artículo 8° de la Constitución Política establece que: "Son públicos los actos y resoluciones de los órganos del Estado, así como sus fundamentos y los procedimientos que utilicen. Sin embargo, solo una ley de quórum calificado podrá establecer la reserva o secreto de aquellos o de estos, cuando la publicidad afectare el debido cumplimiento de las funciones de dichos órganos, los derechos de las personas, la seguridad de la Nación o el interés nacional".

3.- Que, a su vez el artículo 5° la Ley N° 20.285 presume pública toda información que obre en poder de la Administración, salvo en los casos de excepción por alguna de las causales de reserva o secreto establecidas en su artículo 21 y en otras leyes de quorum calificado.

4.- Que, la Ley Nº 20.285 en su artículo 21, Nº 1, literal c), dispone, en lo pertinente con esta solicitud, que se podrá denegar total o parcialmente el acceso a la información: "Tratándose de requerimientos de carácter genérico, referidos a un elevado número de actos administrativos o sus antecedentes o cuya atención requiera distraer indebidamente a los funcionarios del cumplimiento regular de sus labores habituales".

5.- Que, la petición del requirente comprende un gran volumen de información vinculada al Programa de Alimentación Escolar que administra JUNAEB, que no se encuentra sistematizada ni clasificada en los términos que solicita el requirente. De tal forma que, para satisfacer su requerimiento sería necesario destinar como mínimo a dos funcionarios con dedicación exclusiva a esa tarea durante un mes, por las características propias de la información objeto de esta solicitud.

6.- Que, en consecuencia y en atención al elevado número de antecedentes que constituye la información solicitada, el Servicio se encuentra imposibilitado de acceder a la petición de la requirente, ya que no dispone del recurso humano suficiente que pueda destinar a realizar estas funciones de manera preferente, sin que ello implique distraer indebidamente a los funcionarios del cumplimiento regular de sus labores habituales.

7.- Que, a mayor abundamiento y en atención a la crisis sanitaria que afecta al país en la actualidad, es más difícil aún destinar a un número determinado de funcionarios para que se dediquen a realizar una función de manera exclusiva, en circunstancias que las prioridades de este momento requieren contar con la mayor cantidad de recursos humanos disponibles para el cumplimiento de los objetivos institucionales. Por lo tanto:

RESUELVO:

ARTÍCULO PRIMERO: DENIÉGASE el acceso a la información pública requerida por don **Jorge Lobos Ponce**, solicitud Nº AJ009T-00002074, de fecha 27 de mayo de 2020, de acuerdo con el artículo 21, N°1, literal c) de la Ley 20.285, Sobre Acceso a la Información Pública, singularizada en el considerando 1.- del presente acto administrativo.

ARTÍCULO SEGUNDO: INCORPÓRESE en el Índice de Actos y Documentos calificados como secretos o reservados, del Portal de Transparencia la presente resolución denegatoria, una vez que se encuentre **ejecutoriada,** de conformidad a lo establecido en el artículo 23 de la Ley Nº 20.285 y en la Instrucción General Nº3 del

2008, del CPLT.

ARTÍCULO TERCERO: De acuerdo con lo prescrito en el artículo 24 y siguientes de la Ley N° 20.285, en contra de la presente resolución el requirente tendrá derecho a recurrir ante el Consejo Para La Transparencia, solicitando amparo a su derecho de acceso a la información, dentro del plazo de quince días, contado desde la notificación de la presente resolución.

ARTÍCULO CUARTO: NOTIFIQUESE la presente resolución a don **Jorge Lobos Ponce** a la casilla electrónica indicada en su presentación.

ANÓTESE,

Junta Nacional de Auxilio Escolar y Becas
SECRETARIO GENERAL

JAIME TOHÁ LAVANDEROS
SECRETARIO GENERAL
JUNTA NACIONAL DE AUXILIO ESCOLAR Y BECAS

JUNTA NACIONAL DE AUXILIO ESCOLAR Y BECAS
JEFE GABINETE
CHILE

ALA/macm

Junta Nacional de Auxilio Escolar y Becas
DIRECCIÓN

Distribución:
- Gabinete
- Departamento Jurídico
- Oficina de Partes

JUR N° 7615-2020

A.3. Diagramas entidad-relación adicionales

A continuación, se presentan diagramas entidad-relación desarrollados para el sistema, tanto actual como a futuro. En ellos, se detalla el diseño de la base de datos con las que se comunica actualmente el componente JunaebRawData en el caso de la figura 3.3 descrita en la sección 3.3, además de las tablas propuestas para objetivos futuros, fuera del alcance de esta memoria. Dichos objetivos también están presentes en la sección 3.3

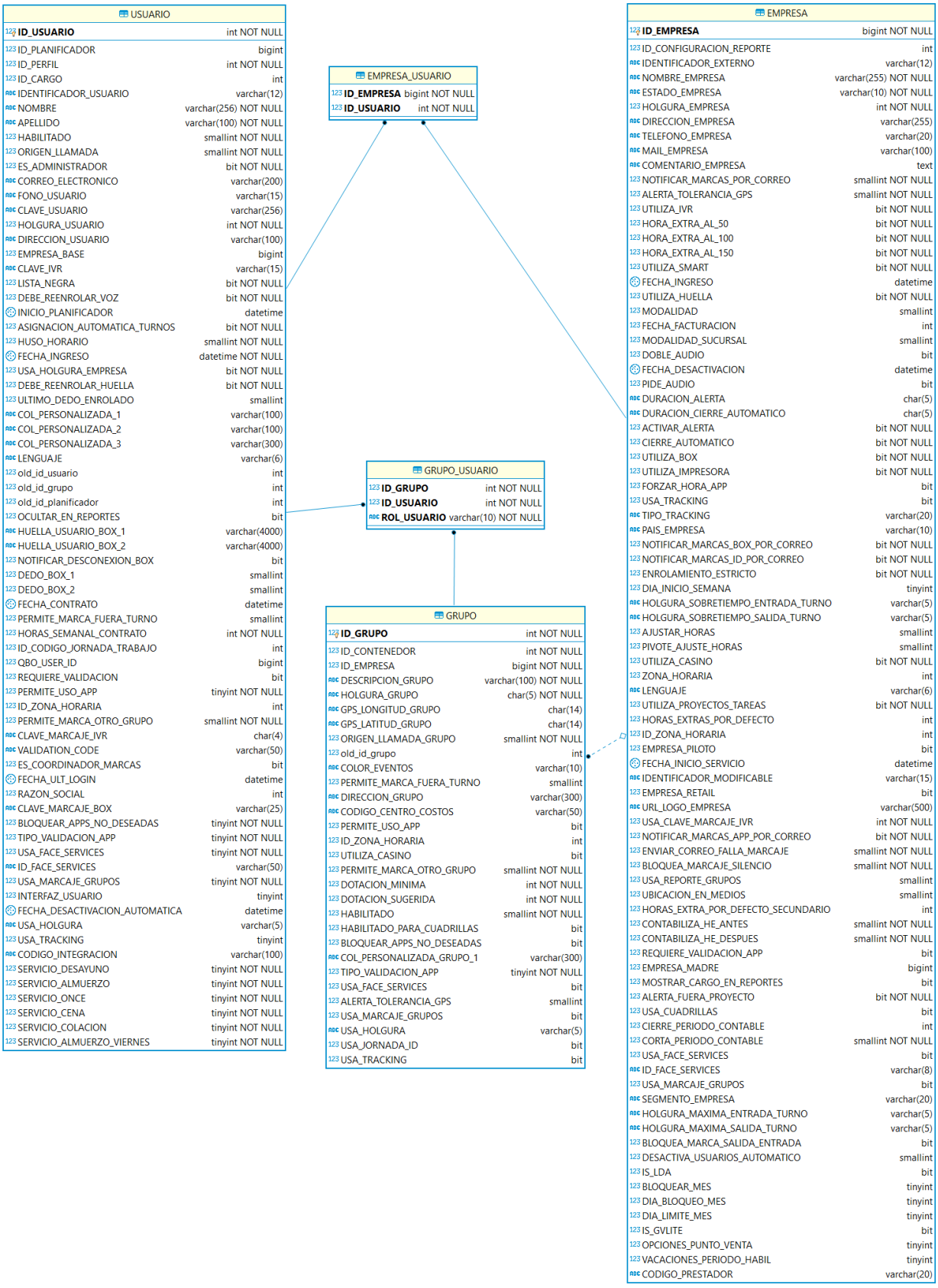


Figura A.1: Diagrama entidad-relación que muestra la relación entre usuarios, empresas y grupos.

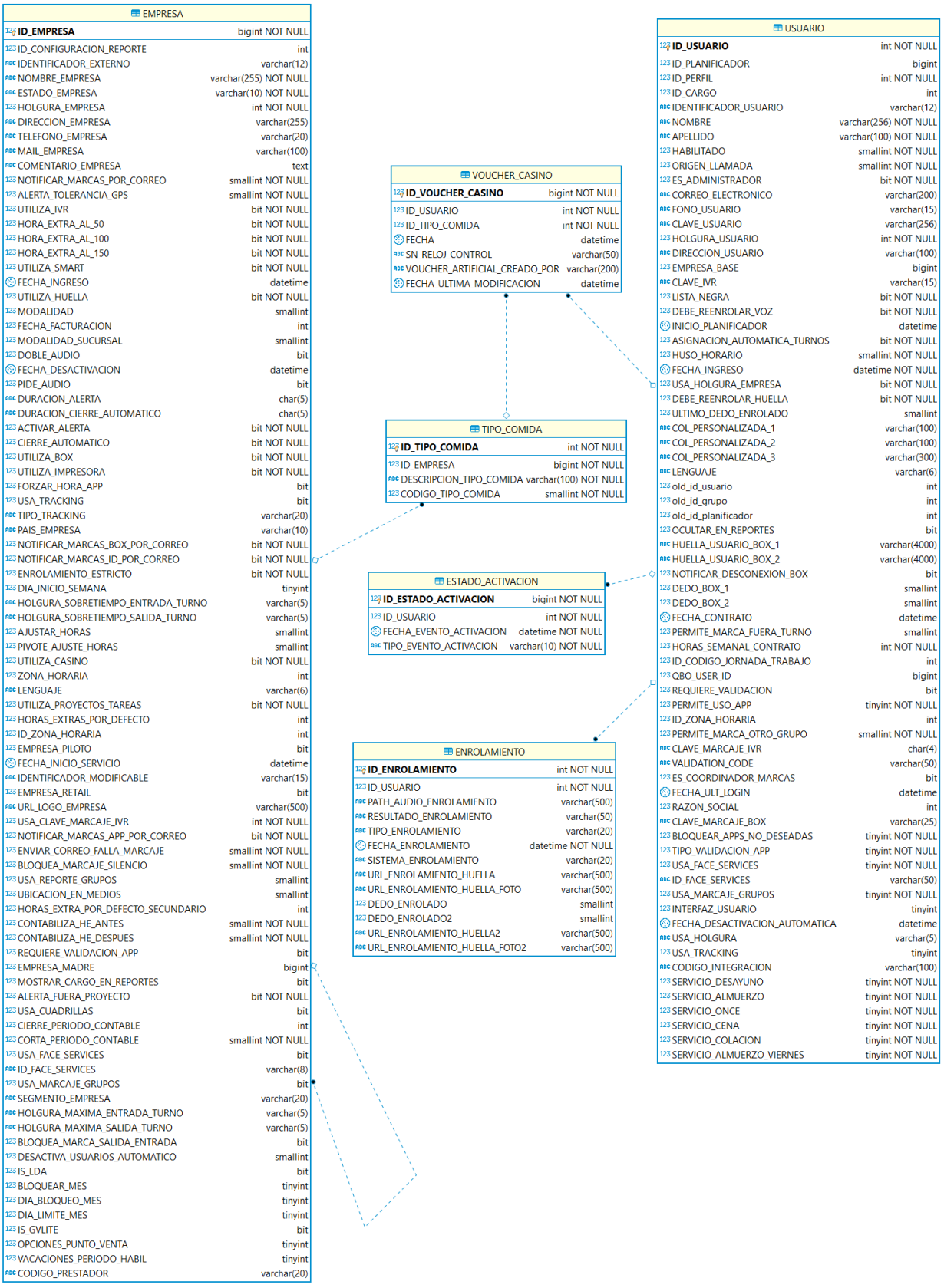


Figura A.2: Diagrama entidad-relación que muestra la relación entre usuarios, su enrolamiento, activación y marcaje.

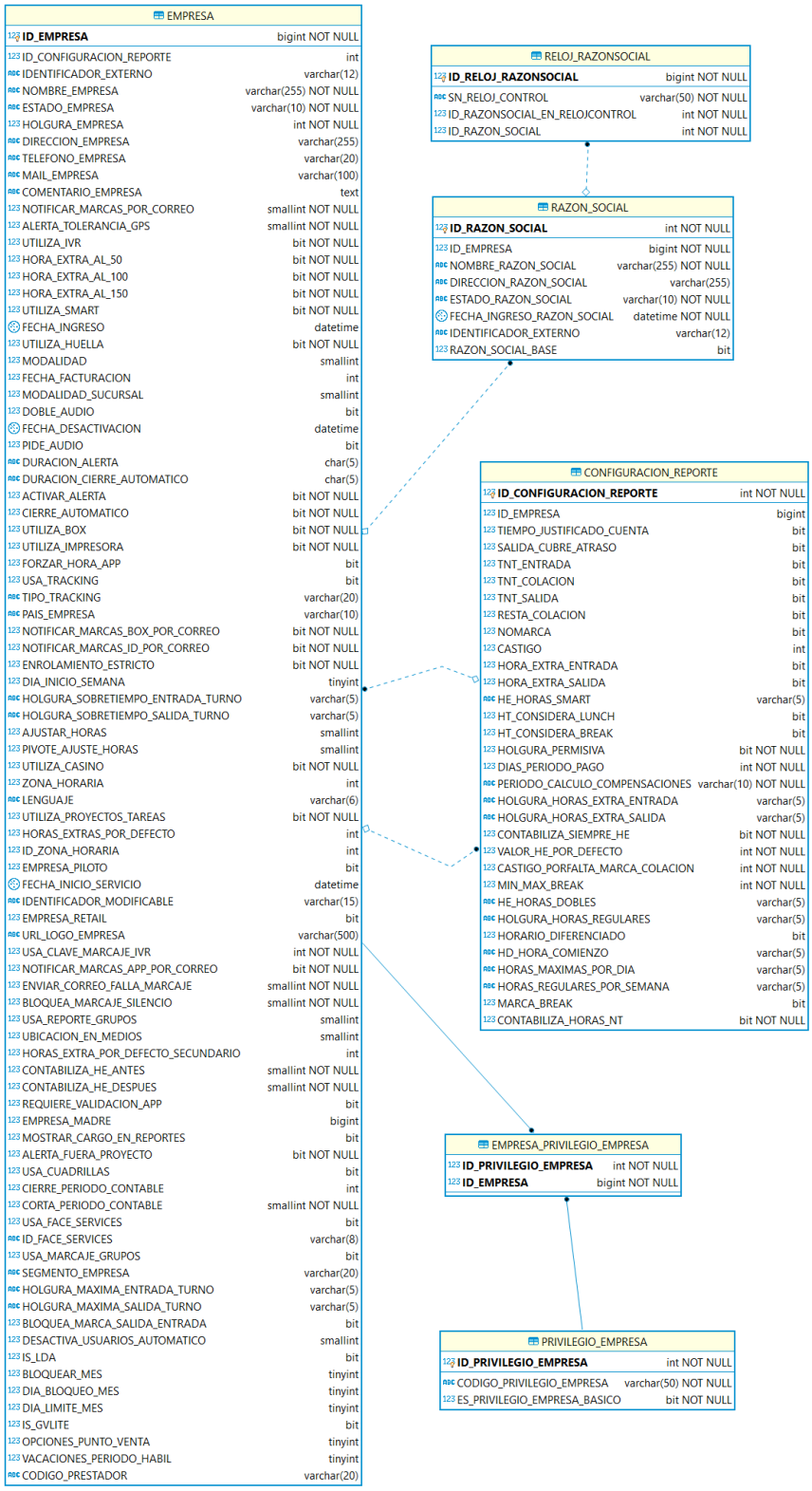


Figura A.3: Diagrama entidad-relación que muestra la relación entre las empresas, su configuración de reportes, su razón social y los privilegios con los que cuenta.

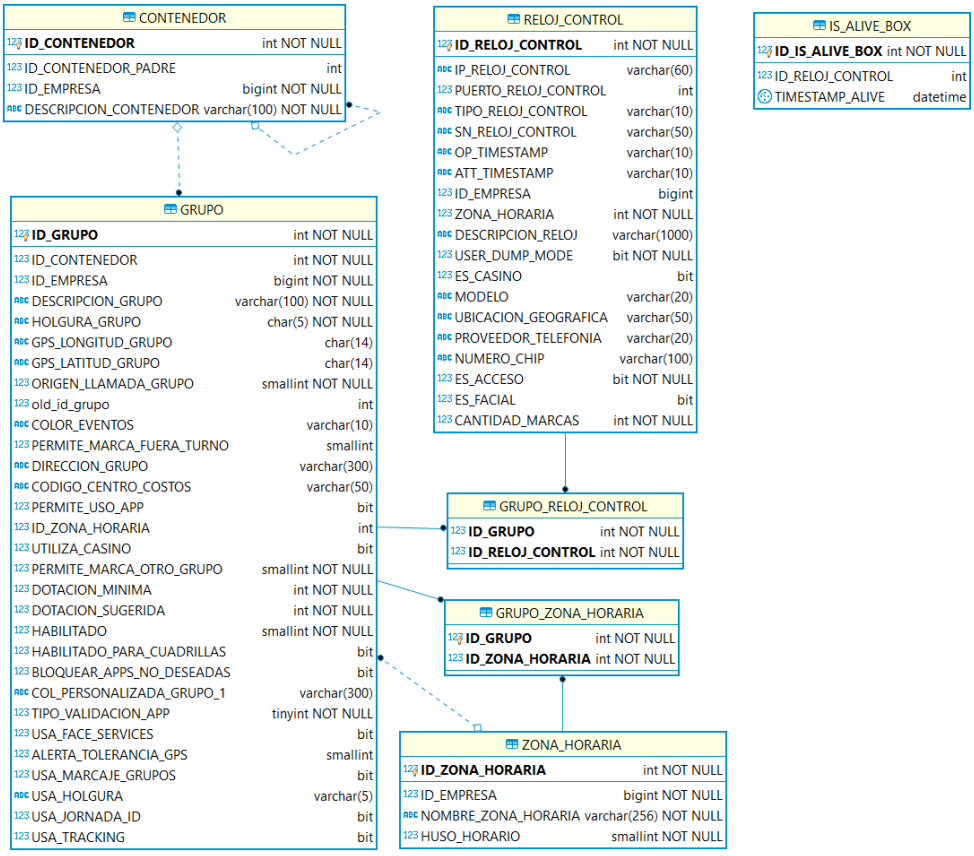


Figura A.4: Diagrama entidad-relación que muestra la relación entre grupos y contenedores.

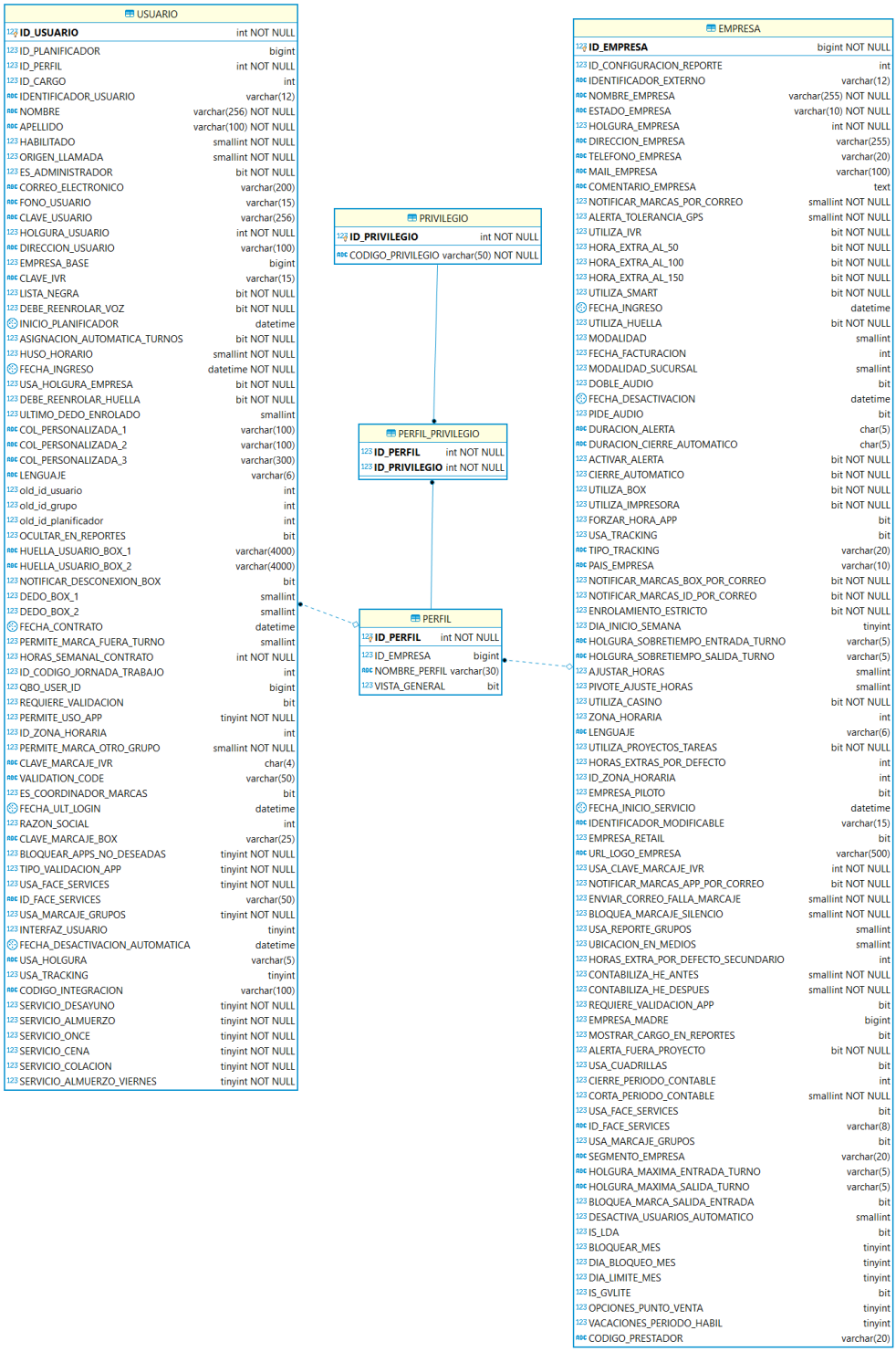


Figura A.5: Diagrama entidad-relación que muestra cómo están agrupados los perfiles.

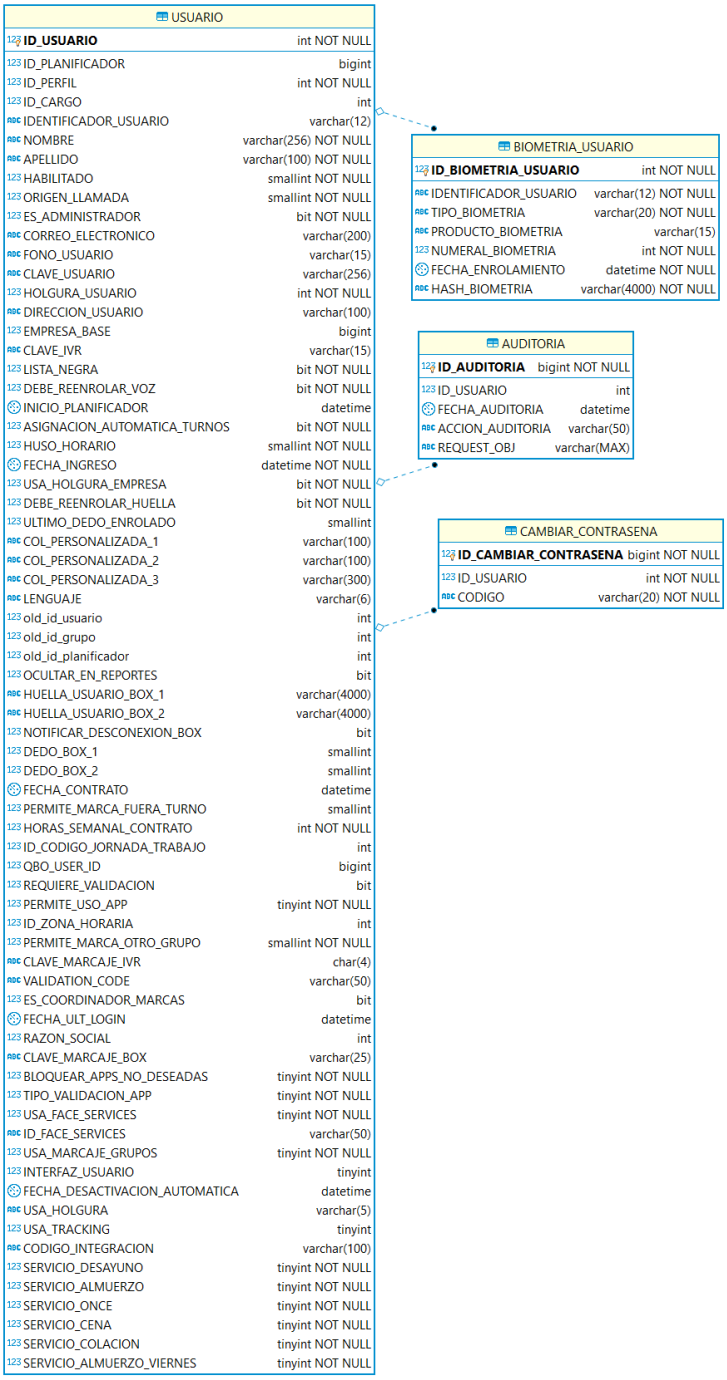


Figura A.6: Diagrama entidad-relación que muestra cómo están agrupados los datos de registro.