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

MEJORANDO EL SISTEMA DE ASIGNACIÓN DE RECURSOS DEL DCC

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

SEBASTIÁN ANDRÉS ZAPATA ASCENCIO

PROFESOR GUÍA:
Sergio Ochoa Delorenzi

PROFESOR CO-GUÍA:
Juan Arriagada Cancino

MIEMBROS DE LA COMISIÓN:
Jose Urzúa Reinoso
María Cecilia Bastarrica Piñeyro

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN
POR: SEBASTIÁN ANDRÉS ZAPATA ASCENCIO
FECHA: 2023
PROF. GUÍA: SERGIO F. OCHOA DELORENZI

MEJORANDO EL SISTEMA DE ASIGNACIÓN DE RECURSOS DEL DCC

El Departamento de Ciencias de Computación de la Universidad de Chile tiene a cargo una serie de sistemas de software que, en un inicio, surgieron como desarrollos personales de miembros de la comunidad y pasaron a ser parte de la infraestructura del DCC. A pesar de que los productos funcionan de manera estable, existen componentes propios que afectan directamente a la gobernabilidad y capacidad de integración entre los sistemas.

Una de estas aplicaciones es el Sistema de Asignación de Recursos del DCC (SAR), que es donde se enfoca este trabajo de memoria. El SAR es utilizado actualmente por académicos y funcionarios del departamento para asignar o reservar espacios físicos, como por ejemplo, salas de reuniones o auditorios. Este sistema sufre de problemas de gobernabilidad y tiene diversas limitaciones para permitir la integración de sus datos y servicios.

El objetivo de este trabajo es mejorar la gobernabilidad y la capacidad de integración del SAR, velando porque este cumpla con una serie de requisitos que el DCC definió para asegurar su estabilidad y mantenibilidad de sus sistemas. Para ello se analizó la aplicación y se documentó cada aspecto relevante para su posterior intervención, identificando las principales restricciones y desafíos que podrían presentarse al momento de realizar los ajustes al código fuente.

Dentro de los ajustes realizados, se generó un ambiente de desarrollo que permite desplegar el sistema en cualquier máquina (servidor), con un mínimo esfuerzo. Además, se actualizaron la mayoría de las dependencias del proyecto, y se implementó una nueva API que permitirá una mejor integración con otros sistemas del DCC. También se implementó un sistema de autenticación usando las herramientas del Portal DCC, que permite acceder al SAR utilizando una cuenta de U-Pasaporte.

Para evaluar el impacto de la solución, se extendió la funcionalidad del SAR agregándole gráficos estadísticos que permiten visualizar la frecuencia de uso de los espacios físicos. El esfuerzo requerido para implementar dicha extensión fue comparado considerando el sistema legado y el nuevo. Además, se evaluó el nivel de cumplimiento de los requisitos de gobernabilidad del nuevo sistema, obteniendo un resultado altamente positivo y cumpliendo así con los objetivos planteados.

Se espera que la nueva versión del SAR siga su paso a producción y así brindar un mejor servicio al Departamento.

A mis padres Jenny y Miguel, por el infinito amor que me entregan día a día.
A mi hermano Vicente, por todas las risas y alegrías que me da.

Agradecimientos

Gracias...

A mi familia, por estar siempre presente. Por apoyarme en cada paso que doy y por el sacrificio que han entregado día a día, sé que lo hacen con un amor incondicional. Sin ellos nada de esto hubiese sido posible, los amo.

A todos aquellos que fueron compañeros (de clase, de pasillo y de carrete), alumnos, auxiliares, ayudantes, ahijados y apadrinados, pero que ahora son tremendas amistades, de verdad disfruté mucho estos 6 años en la U y todo fue gracias a ustedes.

A mis amigos de La Serena, ciudad donde crecí, por siempre darme apoyo y confianza donde sea que esté, son bacanes y espero que lo sepan, los quiero mucho.

A mis profesores guías Sergio Ochoa y Juan Arriagada, por orientarme en cada decisión al realizar este trabajo de memoria. Su gran experiencia se vio reflejada de inicio a fin, demostrando su profesionalismo.

Al equipo Duckietown 2018-2019, liderado por el profesor Matías Mattamala, haber sido alumno y auxiliar de estos cursos fue el verdadero primer paso hacia donde estoy ahora. Y gracias al profesor Miguel Campusano quien también formaba parte del equipo, por haber continuado enseñándome y siendo un guía a lo largo de mi carrera.

A Giovanni y Hans, por abrirme las puertas de su empresa y darme la oportunidad de aplicar mis conocimientos en el mundo laboral, potenciando así mi aprendizaje. También a Cristopher y José, por confiar en mí al poner en marcha un bonito proyecto en conjunto.

Finalmente, una mención especial al grupo de Werewolf Beauchef, las risas no faltaron en estos 6 años. Viva la */moshikill*.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

La capacidad de un sistema para perdurar en el tiempo involucra diversas aristas, que deben ser abordadas para el correcto funcionamiento del mismo. Muchas de estas no tienen que ver con el código fuente en sí, sino con otros criterios como su gobernabilidad y su capacidad de evolución e integración con otros sistemas.

El Departamento de Ciencias de Computación (DCC), perteneciente a la Facultad de Ciencias Físicas y Matemáticas (FCFM) de la Universidad de Chile, posee diversas aplicaciones que abordan problemáticas específicas de la operatoria de dicha unidad. Estos sistemas han ido surgiendo y evolucionando de forma poco armónica a lo largo de los años. El diseño e implementación de estos se ha hecho a partir del trabajo personal de académicos, memoristas y estudiantes, quienes han utilizado entornos ad hoc, sin seguir estándares o acuerdos acerca de cómo estructurar y documentar los sistemas.

Entre estos desarrollos, existe uno llamado *Sistema de Asignación de Recursos (SAR)* [1], que permite gestionar los distintos espacios físicos pertenecientes al departamento. Este sistema se encuentra desde el año 2017 en su versión 3.0, después de haber sido intervenida múltiples veces desde la base del trabajo personal de un memorista que mejoró el SAR previamente [2]. Dicho proyecto a su vez, fue evolucionado de una implementación anterior proveniente del año 2003 que resolvía la problemática de gestión de recursos.

En marzo del 2022, el Área de Desarrollo de Software del DCC definió una serie de criterios para poder considerar a un sistema como confiable, estable y gobernable. Estos criterios se detallan en el Anexo I. Al inicio de esta memoria, y al igual que otros sistemas del departamento, el SAR no se encontraba alineado con dichos criterios, por lo que se debía asegurar su gobernabilidad y capacidad de integración con otros sistemas con tal de seguir operando.

1.1. Justificación del trabajo

La operación actual del SAR es estable, sin embargo, su estructura no estaba clara ni documentada por diversas razones:

1. Diversos equipos y memoristas han trabajado en distintas partes de este sistema, sin mantener documentación en común.

2. No se había definido formalmente la estructura que debía respetar este software, o los

servicios que debía usar (o exponer) de cara a su integración con otros sistemas del DCC.

3. No estaba claro el alcance real del sistema, es decir, la lista de servicios que ofrece.

4. No mantenía documentación acorde a las necesidades de un desarrollador, como sus procesos o su arquitectura.

Todas estas situaciones generaban un problema de gobernanza del sistema, particularmente de mantenibilidad en el mediano y largo plazo; esto limitaba su evolución y su integración con otras aplicaciones del DCC. Debido a esto, se debía realizar un estudio exhaustivo de su arquitectura, código fuente, recursos utilizados y toda aquella información relevante que pertenezca al sistema.

El DCC busca generar un ecosistema de aplicaciones que haga uso de servicios comunes (unificados), pero el desarrollo de productos aislados representa un problema serio para el desarrollo de la infraestructura de software del departamento. Este es el problema principal abordado, para un sistema en particular: el Sistema de Asignación de Recursos (SAR).

1.2. Objetivos de la Memoria

El objetivo general de esta memoria es intervenir la estructura y el código fuente del SAR, así como también generar la documentación de soporte del mismo, para que dicho sistema se vuelva gobernable e integrable al resto del ecosistema de aplicaciones del departamento. Esto significa que el sistema debe cumplir con una serie de criterios definidos por el DCC [3] (ver Anexo I), para así mantener la estabilidad del mismo y permitir su integración con otros sistemas internos y externos de manera estructurada.

A partir del análisis del sistema actual, se espera identificar aspectos y funcionalidades faltantes para posteriormente implementarlos, además de mejorar aquellos que se determinen como deficientes, siempre en base a los criterios mencionados en el Anexo 1. Al intervenir el sistema, se buscará mostrar el impacto del trabajo realizado, en términos de capacidad de gobierno e integración del sistema SAR. Esta intervención involucra también llevar el código fuente a versiones más nuevas de la tecnología usada en el sistema legado. Esto implica el ajuste y eventualmente la reimplementación de parte de los servicios existentes.

Por último, se espera que la nueva versión del sistema sea capaz de resolver los problemas enunciados, manteniendo además el código fuente y la documentación del sistema de una manera ordenada y bien estructurada, para que el sistema cumpla los criterios mínimos de gobernabilidad establecidos por el DCC.

Para cumplir el objetivo general, se definieron los siguientes objetivos específicos:

1. Generar la documentación necesaria para cumplir los criterios de Gobernabilidad e Integración de Sistemas del DCC (ver Anexo I). Se espera facilitar el entendimiento del sistema para cualquier desarrollador o usuario del mismo.

2. Generar un repositorio de código fuente con las modificaciones y nuevas implementaciones, conteniendo también documentación acerca del “as-is” y del “to-be” del sistema.

3. Modificar la funcionalidad del sistema de legado para: 1) llevarlo a versiones actualizadas de las tecnologías que usa, y 2) cumplir con las capacidades mínimas de integración entre sistemas definidas por el DCC (ver Anexo I).

4. Implementar (al menos) un servicio que, a modo de prueba de concepto, extienda la funcionalidad actual del sistema, y de esa manera, muestre la mejora que se ha logrado en términos de gobernabilidad de la aplicación.

El logro de estos cuatro objetivos específicos permite demostrar que la solución alcanzada en esta memoria efectivamente mejora la gobernabilidad y capacidad de integración del sistema en cuestión.

1.3. Resumen de la Solución

La solución obtenida para solucionar el problema abordado en esta memoria, se basa en la intervención del sistema SAR en todos sus niveles, desde la documentación hasta el código fuente. Se modificaron aspectos esenciales pero mínimos para poder cumplir con los criterios de gobernabilidad e integración de sistemas del DCC.

En concreto, se generó toda la documentación necesaria para que el sistema sea gobernable. Además, se realizaron ajustes precisos en el código fuente para cumplir con los criterios de integración de sistemas, esto es, la implementación del mecanismo de autenticación oficial del DCC y la implementación de un servicio que extienda la funcionalidad del sistema. También, se integra la reconocida plataforma Docker para permitir que el SAR pueda ser replicable en cualquier máquina sin la necesidad de instalar dependencias manualmente, y sabiendo que debe ser capaz de funcionar siempre bajo las mismas condiciones.

1.4. Estructura del Documento

Este documento de memoria se compone de 7 capítulos, siendo el primero una introducción al documento mismo. El capítulo 2 entrega un análisis de las características principales del SAR, como su modelo de datos, sus usuarios y qué tan preparado está para cumplir los lineamientos deseados.

El capítulo 3 muestra con detalle los ajustes técnicos realizados al sistema con tal de cumplir los lineamientos pedidos por el DCC. El capítulo 4 muestra la extensión que se le realizó al sistema para comprobar el impacto de los ajustes realizados al mismo, además se detallan especificaciones y alcances del SAR.

El capítulo 5 presenta los resultados obtenidos luego de realizar todos los ajustes y extensiones planteadas al inicio del trabajo del memorista, se detalla el proceso de evaluación, resultados y nivel de cumplimiento del sistema con respecto a los requisitos del DCC. El capítulo 6 presenta las conclusiones obtenidas en base al trabajo realizado y se plantean los siguientes pasos a seguir para afinar el sistema y cómo proceder a futuro. El capítulo 7 contiene las referencias del documento y la bibliografía consultada durante la realización del trabajo.

Capítulo 2

Análisis del Sistema SAR

Al inicio de la memoria, el SAR no contaba con una documentación que facilite la comprensión acabada del sistema. Para abordar esa necesidad, en este capítulo se presentan los resultados de la exploración del producto, que evidencia parte del trabajo realizado por el memorista en cuanto a la generación de documentación necesaria para integrar el SAR al ecosistema del DCC. Con esto se busca dar cumplimiento a parte de los Requisitos de Gobernabilidad e Integración definidos para los sistemas del DCC, en específico cuatro: 1) Descripción del proceso que apoya el memorista, 2) Arquitectura del ecosistema, 3) Arquitectura del software y 4) Modelo de datos.

La versión inicial del SAR (aquella que se tomó al inicio del trabajo de título) se encuentra actualmente en producción, y es usada principalmente por funcionarios y profesores del departamento para llevar de mejor manera sus labores a cabo (también lo puede utilizar el Centro de Alumnos del DCC). Algunas de estas labores se ven beneficiadas por el sistema, como lo es el gestionar eficazmente los espacios físicos del departamento, para que puedan ser usados por la comunidad durante un tiempo determinado y por una razón en concreto.

2.1. Identificación del Problema y Contexto a Abordar

La Figura 1 que se muestra a continuación presenta el tablero de exploración del problema y contexto abordado por la versión legada del SAR, posteriormente se explica con más detalle cada sección de éste.

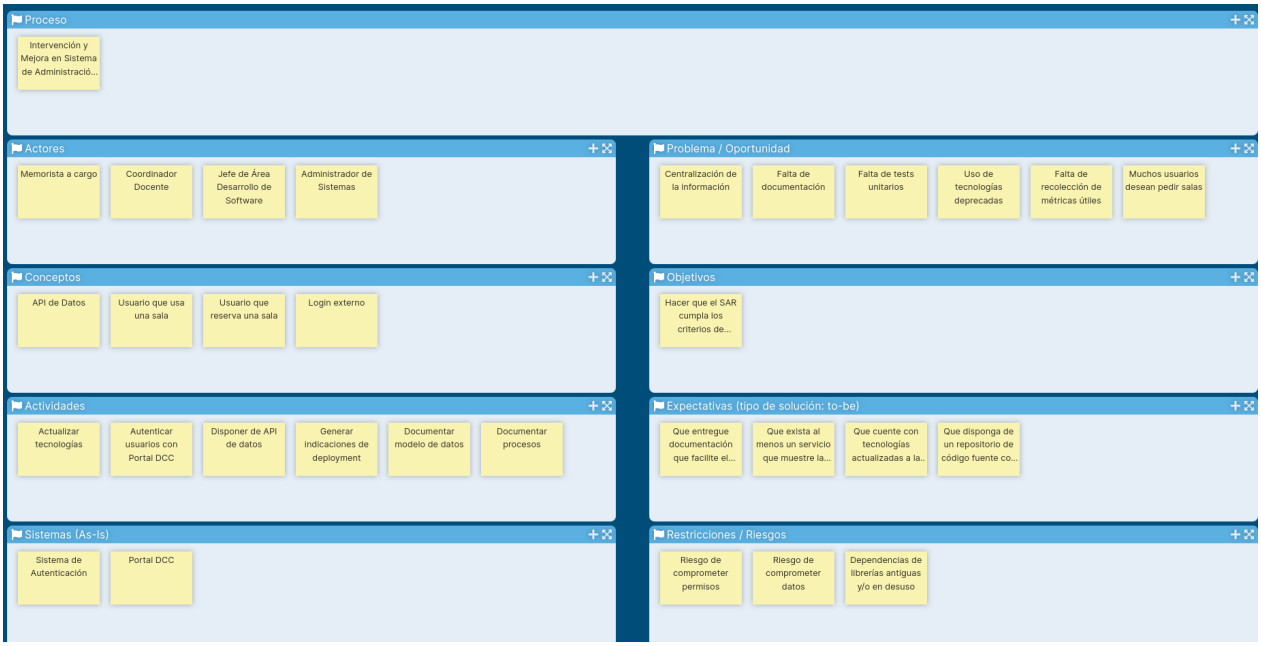

Figura 2.1: Tablero de exploración del problema y contexto para el SAR

- **Proceso**: El proceso abordado es la Intervención y Mejora en el SAR, este abarca todos los subprocesos necesarios para cumplir los objetivos propuestos, y a partir del cual, se exploran los siguientes componentes.

- **Actores**: El actor que realiza las intervención directa en el sistema es el memorista, el cual se encuentra guiado por su profesor guía (Coordinador Docente) y co-guía (Jefe del Área de Desarrollo de Software del DCC), se encargaron de guiar al memorista en el proceso de mejoras y necesidades del sistema para funcionar con el departamento, además guiar en el proceso técnico y de desarrollo del sistema, tanto en software como en servidores de testing y producción. El Administrador de Sistemas veló por ofrecer los recursos necesarios para testear el SAR en contexto de producción.

- **Problema / Oportunidad**: Los problemas identificados en el SAR son la falta de elementos mínimos necesarios para intervenirlo cuando sea necesario, principalmente falta de documentación de tecnologías y despliegue, por otro lado no existía una forma de aprender a obtener información base para explorar métricas de uso, y todos los datos se mantenían en el sistema, sin ser este alimentado por una fuente universal de datos. Por último, era difícil montar el SAR en un entorno local, dado que, se necesitaba realizar una serie de pasos para poder hacerlo funcionar, y no existía una guía de instalación actualizada (para las tecnologías deprecadas que utilizaba).

- **Conceptos**: Los conceptos que se exploran en este proceso de intervención guardan relación con los criterios pedidos por el DCC, estos son relativos a la documentación de aspectos del sistema y a los datos que guarda el mismo para ser consumidos por el departamento.

- **Objetivos**: El principal objetivo del proceso es "Hacer que el SAR cumpla los criterios de Gobernabilidad e Integración de sistemas del DCC", y se encuentra en sintonía con el objetivo general y los objetivos específicos del trabajo de la memoria.

- **Actividades**: Dentro de las actividades principales del proceso se destaca la documentación, la actualización de tecnologías a versiones más actuales y la disposición de datos al servicio del DCC.
- **Expectativas (to-be)**: Existen expectativas que se esperan del proceso y se relacionan con los objetivos específicos, de este modo se indican los macro-servicios que esta solución aportará al proceso. Se espera que:
  1. El sistema entregue documentación que facilite el entendimiento del mismo.
  2. El sistema cuente con tecnologías actualizadas a la fecha del proceso.
  3. Se disponga de un repositorio de código fuente con modificaciones e implementaciones.
  4. Exista al menos un servicio que muestre las mejoras en términos de gobernabilidad.
- **Sistemas (as-is)**: El sistema legado no contaba con fuentes externas de datos (tampoco del propio DCC), por lo que era un sistema aislado dentro en el ecosistema de desarrollos del departamento.Mientras que lo relativo a reservas pertenece correctamente al SAR, lo relativo a la información de los usuarios resulta en una mezcla de del sistema y de una fuente de verdad que pueda gobernar los datos. Uno de los criterios de integración pide precisamente que: "Todos los sistemas deben autenticar usuarios utilizando los servicios que provee el DCC". Dado esto, el único sistema del SAR que debe requerir una intervención o reemplazo es el Sistema de Autenticación, como se menciona en el tablero.
- **Restricciones / Riesgos**:
  - **Riesgos**: Existen dos riesgos asociados a la intervención de este sistema y tienen relación con comprometer ciertos datos o permisos del mismo. La API debe estar diseñada de tal forma que no se pueda acceder a datos sensibles de los usuarios, y que no se pueda acceder a datos de reservas que no correspondan al usuario que le corresponde.
  - **Restricciones**: Estas tienen con el traspaso a la nueva versión mejorada del sistema, pues los datos preexistentes deben ser migrados a la nueva versión, y no se debe perder información alguna, además que, el sistema debe seguir funcionando en producción mientras se realiza la migración.

2.2. Principales Interfaces del Sistema

En la Figura 2 se mostrará la principal interfaz de usuario del sistema: el calendario de reservas. En particular se presenta la ocupación del auditorio Picarte entre el 20 y el 26 de Junio de 2022. A través de dicha interfaz también se pueden gestionar las reservas de estos espacios al hacer click en un intervalo de tiempo, disponible si se encuentra después de la hora actual del servidor.

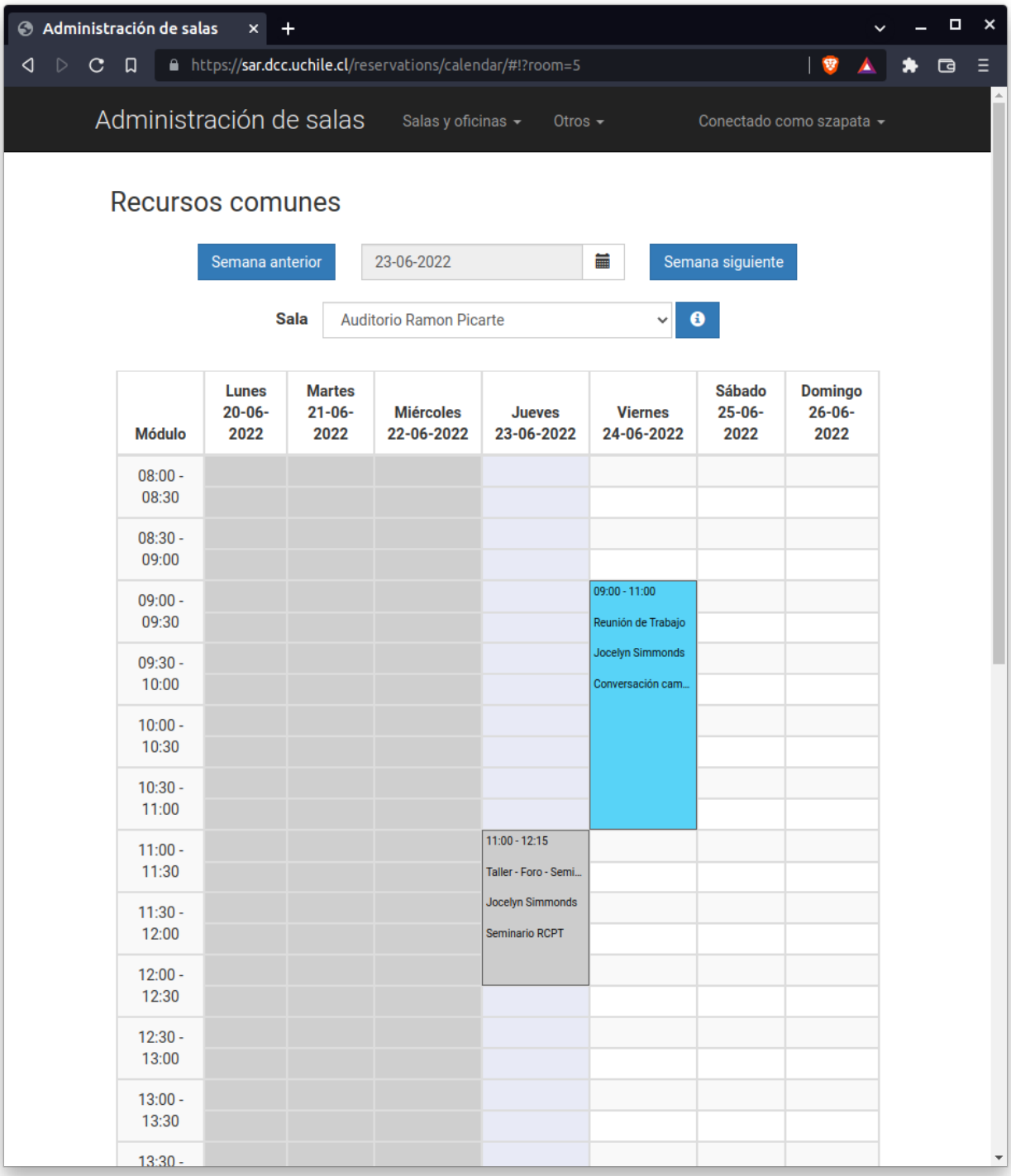

Figura 2.2: Vista principal del SAR para reserva de salas.

El SAR también funciona a modo de gestor de datos de los espacios del departamento. Esto permite encontrar información más detallada de dichos espacios, como por ejemplo, su capacidad y el contacto de quienes los administran (pueden modificarlos o eliminarlos). La Figura 3 mostrará la interfaz del sistema que entrega dicha información al usuario.

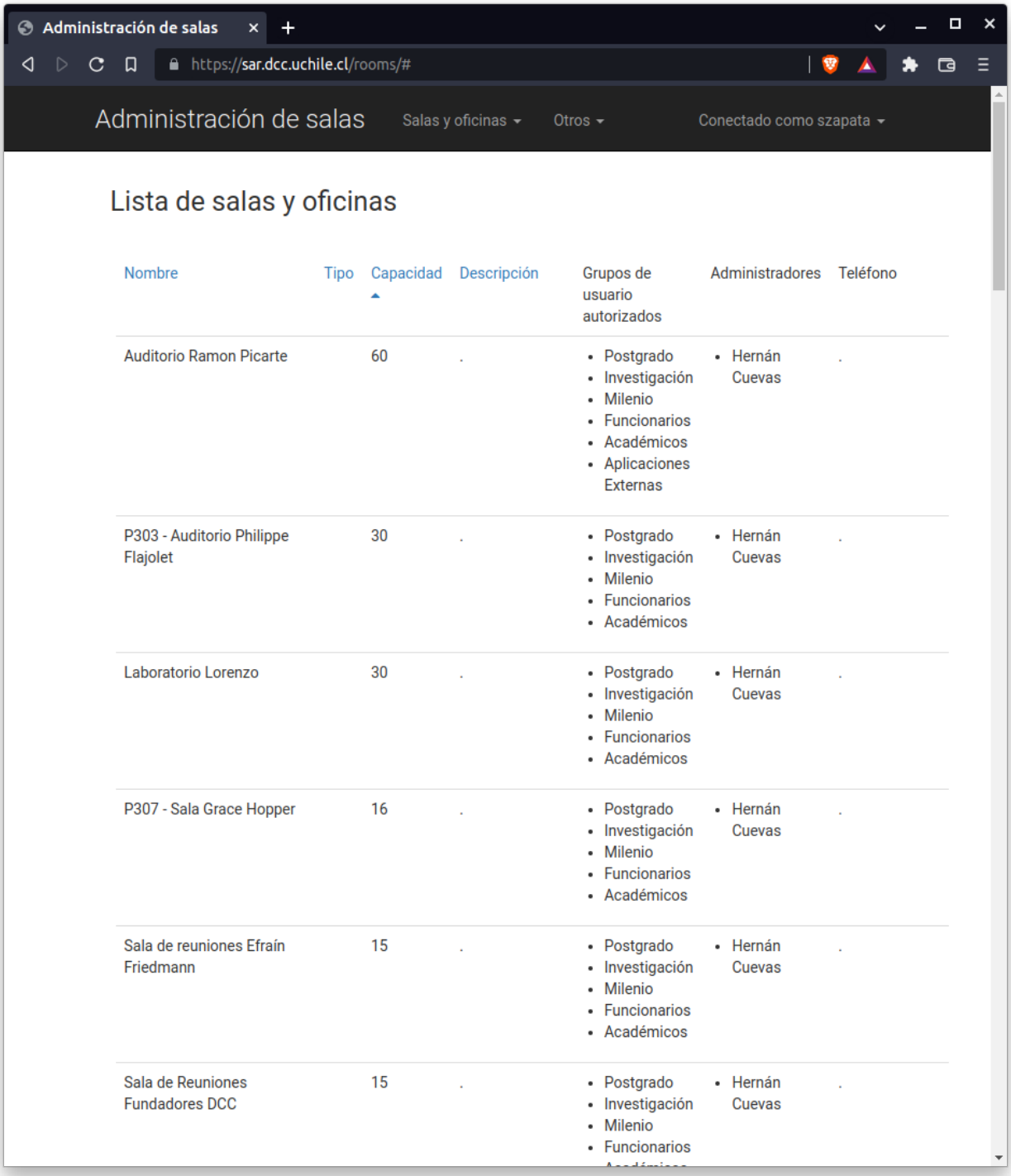

Figura 2.3: Lista de los espacios físicos del departamento.

2.3. Arquitectura del Sistema

A continuación se presentan dos diagramas: de contexto del sistema y de contenedores del sistema. Se utilizan modelos de C4 que ayudan a describir, de una manera amigable, la arquitectura de un software desde lo más general hasta lo más específico [4] (Contexto, Contenedor, Componente y Código). Estos fueron diseñados con el propósito de documentar

el SAR y no existía información previa que resuma esta información.

Un diagrama de contexto pertenece al nivel 1 y se enfoca en los actores y los sistemas internos y externos que son integrados. Un diagrama de contenedor pertenece al nivel 2 y es básicamente un zoom al cuadro del sistema del nivel 1, se realiza un énfasis en describir las tecnologías asociadas a los distintos procesos (contenedores) que corren o que son usadas, por otro lado, los actores ahora son asociados a estos procesos en cada caso.

2.3.1. Diagrama de Contexto del Sistema

El sistema es usado por cuatro usuarios directos que pueden leer, agregar, modificar y eliminar información, esto según el rol que les son asignados. Además, se identifican 2 usuarios indirectos que son registrados por el departamento para contar con una oficina o espacio fijo durante su estadía. Un usuario debe identificarse para acceder al sistema, por lo que, el SAR hace uso de un servicio interno de autenticación y entrega de credenciales, los datos se mantenían dentro del sistema. La Figura 4 muestra el diagrama que da mayor entendimiento al contexto del sistema.

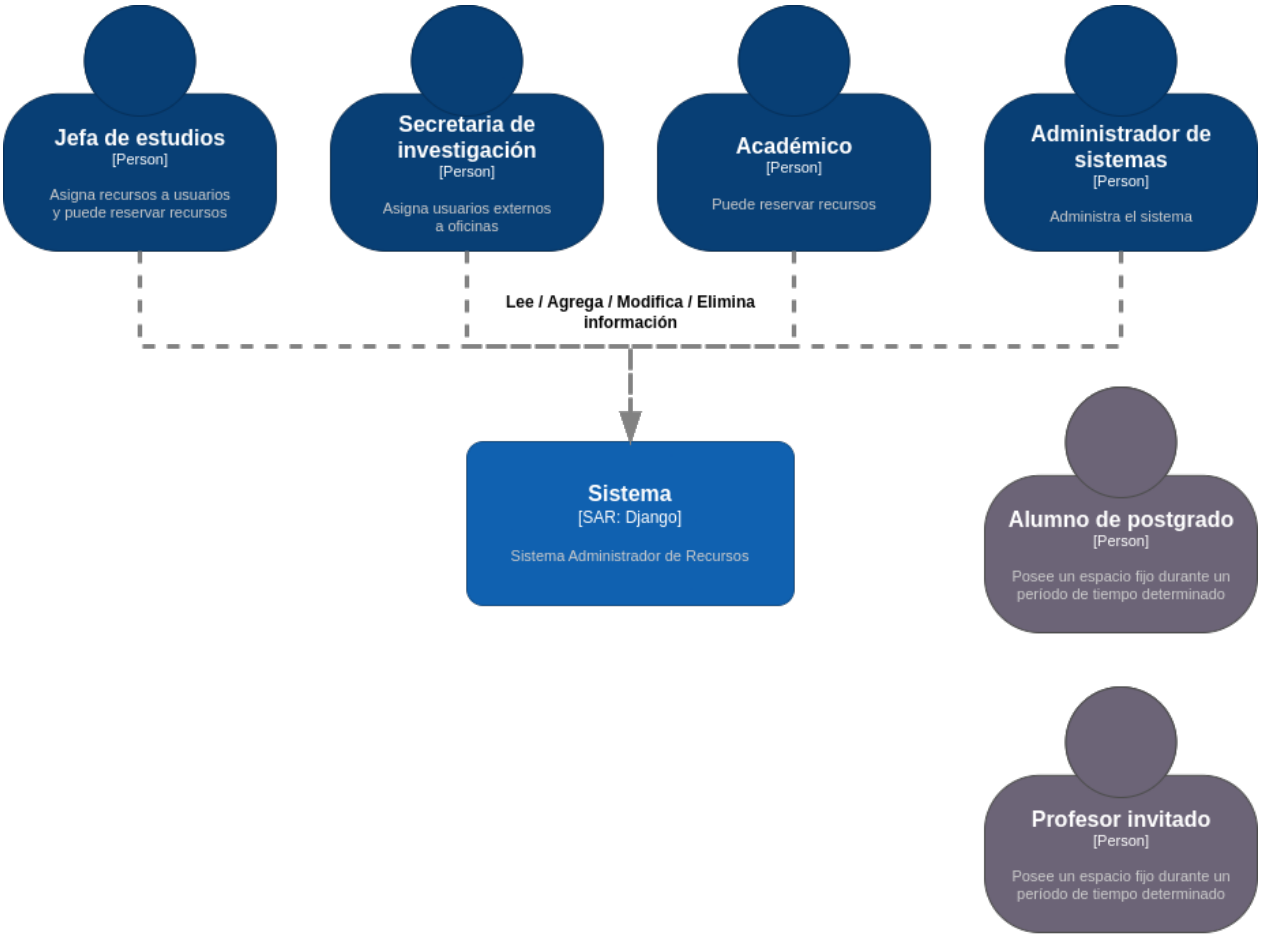

Figura 2.4: Diagrama de Contexto del SAR

2.3.2. Diagrama de Contenedores del Sistema

De la Figura 5 se desprende que el SAR contaba con tres contenedores propios para funcionar. El primero es el almacenamiento de información, se hace uso de una base de datos Postgres en su versión 9.6.24, que permite leer y escribir información relacionada a los usuarios, salas, permisos, entre otros datos.

El segundo contenedor es el reconocido framework web Django en su versión 1.11.3. Este contiene todas las funcionalidades y componentes necesarios para mantener la información, y mostrarla mediante una API que entrega contenido estático, el cual fue programado con AngularJS en su versión 1.6.8, el tercer contenedor.

Es Django el que se encarga de autenticar a los usuarios de su sistema y permite validar el acceso de un usuario.

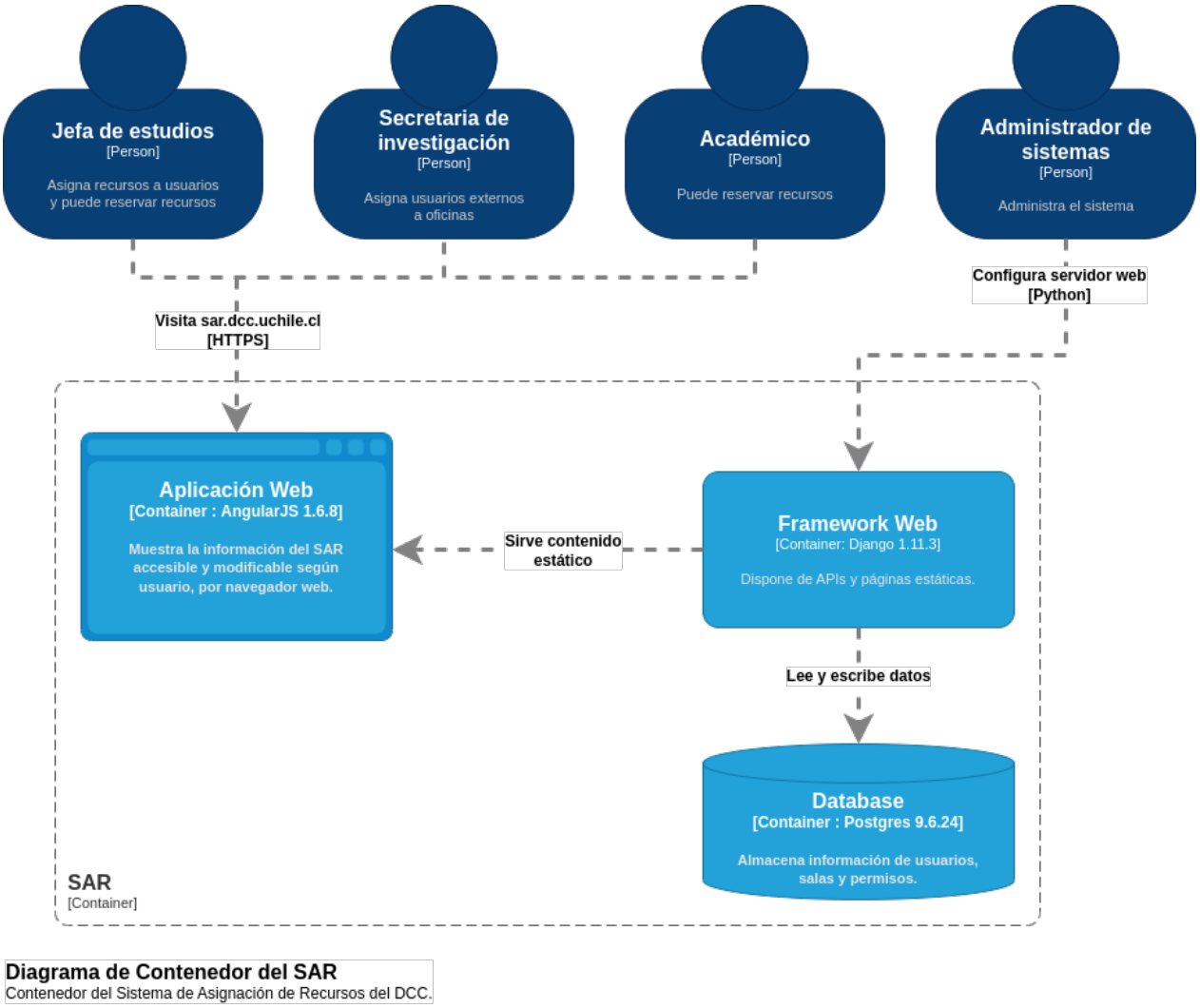

Figura 2.5: Diagrama de Contenedores del SAR

AngularJS es un framework que fue deprecado en enero del año 2022 [5], por lo que no basta con actualizar su versión, debe ser reemplazado por una nueva tecnología. Por otro lado, la versión 1.11 de Django también fue deprecada en abril del año 2020 [6].

2.4. Modelo de Datos

El modelo de datos del sistema busca controlar las reservas de salas que los usuarios piden, además de los permisos que tienen estos últimos. Se presenta a continuación un diagrama colapsado de la versión legada y muestra las relaciones entre las distintas entidades y posteriormente se describen las tablas que la componían, es relevante anticipar que el modelo cambió muy poco con la intervención (solo la tabla User). Si se desea ver el detalle de las tablas con sus atributos y llaves foráneas, se pueden consultar los Anexos II y III.

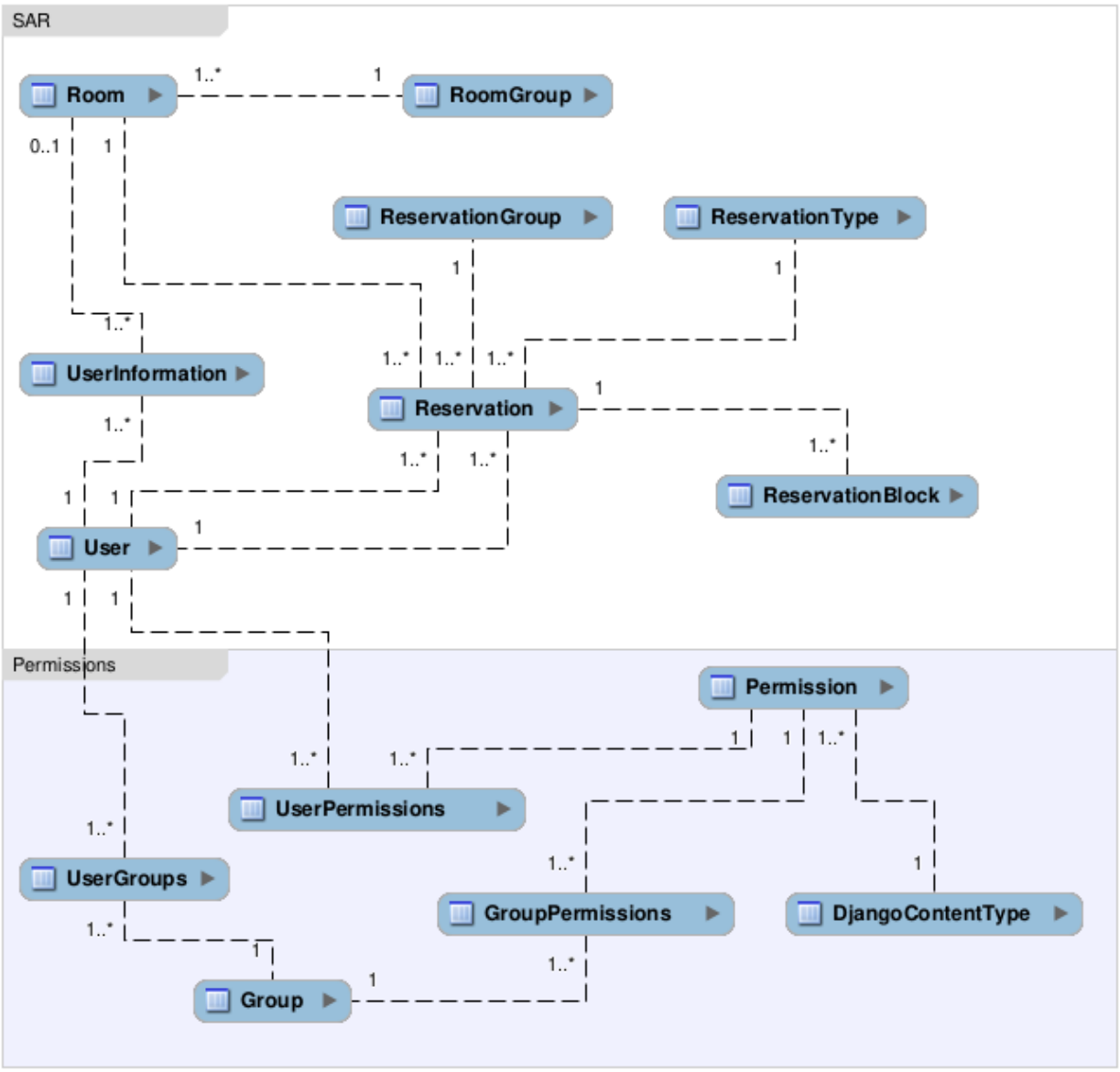

Figura 2.6: Modelo de datos del sistema legado

La Figura 6 muestra dos secciones del modelo de datos: *SAR* y *Permissions*. La primera sección corresponde al sistema en sí, mientras que la segunda evidencia el sistema de permisos.

La entidad *Reservation* es central en el sistema y maneja todas las reservaciones, esto es,

cuando un usuario *User* pide una sala *Room* (para el mismo usuario u otro distinto). Estas reservaciones pertenecen a un grupo de reservaciones *ReservationGroup*, tienen un determinado tipo *ReservationType* y pueden contener múltiples bloques de reserva *ReservationBlock*, estos últimos son de 15 minutos cada uno.

Las salas *Room* pertenecen a un grupo de salas *RoomGroup* (estos grupos son oficinas, salas de clase, salas de reuniones, etc.) y pueden estar asignadas de manera fija a un usuario mediante la entidad *UserInformation* que contiene información específica de una persona inscrita en el sistema, y complementa la tabla *User* proporcionada por Django.

En cuanto a los permisos, estos sirven para asignar responsabilidades a usuarios en el sistema, cada permiso *Permission* puede ser asignado a un usuario mediante la relación *UserPermissions*, y también a un grupo *Group* mediante la relación *GroupPermissions*. De esta manera, un usuario puede pertenecer a un grupo de permisos, como también tener permisos asignados directamente para agregar, editar y/o eliminar información específica del sistema. La entidad *DjangoContentType* es propia de Django y permite una mejor gestión de estos permisos.

2.5. API del Sistema

El sistema legado cuenta con URLs que facilitan los procesos del sistema, pero son muy específicas para cada caso, pues fueron pensadas exclusivamente para realizar operaciones en el SAR y se les dota de muchas responsabilidades, sin dar paso a la modularización o abstracción de funcionalidades.

No se cuenta con un CRUD (Create, Read, Update, Delete) que permita realizar operaciones básicas a los datos, todo esto hace que una aplicación externa no pueda integrarse adecuadamente a este sistema. Tampoco existe documentación que permita entender el funcionamiento de las URLs, por lo que se debe analizar el código fuente para comprender los flujos asociados.

2.6. Repositorio de Código Fuente

Existe un repositorio que ha versionado adecuadamente el sistema en el tiempo. Este pertenece a la organización en GitHub del Área de Desarrollo del DCC, y posee una única rama principal donde vive el sistema legado que fue tomado por el memorista.

El archivo README.md documentado en el repositorio funciona como guía para instalar el sistema, pero no es suficiente para comprender otros aspectos relevantes como la arquitectura del sistema, el despliegue en producción, configuración de variables de entorno o su stack tecnológico. Tampoco se cuenta con una sección de requisitos previos para la instalación, lo que la hace poco confiable y propensa a errores.

2.7. Nivel de Cumplimiento con los Lineamientos de Desarrollo del DCC

La exploración realizada entregó un entendimiento más profundo del sistema, qué busca resolver una problemática y muestra el diseño que la resuelve, sin entrar en aspectos más técnicos. A continuación, se indica el nivel de cumplimiento inicial que tenía el SAR, respecto a los lineamientos de gobernabilidad e integración definidos para los sistemas del DCC (descritos en detalle en el Anexo 1). Esta tabla se le fue entregada al memorista para completarla y apoyar así el proceso de integración y gobernabilidad del sistema.

Tabla 2.1: Nivel de cumplimiento inicial del SAR.

Ítem Evaluado	**Nivel de Cumplimiento**
La arquitectura del ecosistema debe estar documentada y actualizada.	No cumple
La arquitectura de software debe estar documentada y actualizada.	No cumple
El modelo de datos debe estar documentado y actualizado.	Cumple
El sistema debe contar con una API, la cual debe estar documentada y actualizada para acceder a los datos.	No cumple
La autenticación de usuarios debe realizarse usando UPasaporte.	No cumple
El código fuente debe estar apropiadamente versionado y respaldado en un repositorio.	Cumple
El stack tecnológico utilizado debe estar apropiadamente indicado.	No cumple
El stack tecnológico debe estar alineado con las definiciones del DCC.	No cumple
El sistema debe contar con las indicaciones pertinentes para que un tercero pueda levantarlo en un ambiente de desarrollo.	Cumple

El sistema legado cumple con un 33 % de los criterios definidos por el departamento para ser integrado a su ecosistema, esto es un porcentaje bajo por lo que, el sistema no estaba preparado para ser puesto en producción si se hubiese contado con la existencia de los lineamientos del DCC.

2.8. Desafíos para Extender el SAR

Antes de realizar el trabajo de memoria, existían desafíos claros que guardaban relación con la gobernanza a la hora de intervenir el sistema y extenderlo:

- **Falta de documentación de las tecnologías utilizadas**: La ausencia de esta información tan valiosa significa no saber cuáles de estas deben ser actualizadas de forma manual, además de generar potenciales riesgos de seguridad en caso de necesitar rapidez

para actuar frente a una brecha de seguridad en una librería de terceros. Un sistema gobernable debe controlar las tecnologías que utiliza junto con sus versiones, también aplica a las dependencias.

- **Falta de documentación para desplegar el sistema**: El repositorio del sistema legado indica ciertas instrucciones para montar un entorno local de desarrollo, sin embargo, falla a la hora de indicar pre-requisitos del sistema o las versiones de los programas que se deben utilizar. Esto genera una barrera de entrada que dificulta los primeros despliegues del SAR. Por esto mismo, las instrucciones no son replicables en todas las máquinas y no se puede asegurar que funcionen aunque se sigan a la perfección.
- **Falta de documentación para consumir datos del sistema**: El sistema legado no cuenta con documentación sobre la API existente, por lo que no se puede saber qué datos pueden ser accesados, ni la forma en la que estos pueden ser obtenidos. Lo anterior también aplica a datos enviados desde una fuente externa, ya sea para autenticarse o para crear/integrar aplicaciones que requieran hacer uso del SAR. En particular, supondrá un desafío a la hora de extender el sistema al generar métricas de uso de las reservas.

Capítulo 3

Ajustes al Sistema SAR

Es importante destacar que no se ajustó el proceso que apoya el sistema. Sin embargo, sí se modificó parte de la funcionalidad del SAR. Los ajustes a la plataforma para mantener su operatividad fueron diversos, y principalmente se enfocaron en facilitar la creación de un ambiente de desarrollo, actualizar las tecnologías de la plataforma y mejorar la autenticación de usuarios.

3.1. Generando un Ambiente de Desarrollo y de Producción

Para poder realizar el trabajo, se necesitaba un ambiente de desarrollo que permitiera realizar pruebas de forma rápida. Para disponer de este, se consideró actualizar los archivos y las tecnologías que generan las configuraciones iniciales del sistema, en particular las dependencias del backend y del frontend, para apoyar de esta manera una implementación mayor de la plataforma en cuanto a su despliegue en entorno de desarrollo y de producción. Para esto, se realizaron los ajustes que se describen a continuación.

3.1.1. Poetry para backend

Las dependencias de Python se gestionaban mediante un archivo llamado *requirements.txt* que contenía las versiones de las librerías que se utilizaban en el sistema. Esta forma es poco práctica para el manejo de dependencias, ya que no permite manejar de forma automática las versiones de las mismas, ni las dependencias de las dependencias. Para solucionar esto, se implementó Poetry, una herramienta que permite manejar las dependencias de un proyecto de Python, permitiendo crear entornos virtuales dependiendo de la necesidad.

La adición de Poetry al sistema permite además, fijar las versiones de las dependencias recursivamente, lo que significa que los desarrollos realizados en Django (Python) no cambiarán su comportamiento, permitiendo llevar el desarrollo a cualquier máquina sin el riesgo de que las dependencias cambien.

3.1.2. Bower para frontend

Para gestionar las dependencias que utiliza el frontend del sistema legado, se utilizaba Bower, el cual (al igual que Poetry) gestiona las dependencias de un proyecto. Sin embargo, Bower es una herramienta mantenida pero deprecada según sus autores [7], por lo que

recomiendan migrar a una tecnología más actual como Yarn o NPM. Esta migración a otra herramienta quedó pospuesta como trabajo futuro, dada la gran cantidad de dependencias que deben ser migradas.

El frontend fue lo único que contiene tecnologías y herramientas desactualizadas, dado que una migración de esta magnitud requiere de un tiempo mayor al disponible para el proyecto y los objetivos eventualmente se escapan del trabajo requerido. Sin embargo, se realizaron los ajustes necesarios para que el sistema pueda ser desplegado haciendo uso de estas tecnologías y en particular de bower. Un análisis más profundo del porqué no fue actualizado el frontend se verá más adelante en el documento.

3.1.3. Docker para despliegue de aplicaciones

Lo que marca una gran diferencia en el despliegue de la aplicación es el uso de Docker para tal fin. Con esto ya no es necesario instalar las dependencias del sistema en la máquina local, sino que se puede hacer uso de un contenedor que las tenga instaladas y configuradas. El trabajo en este punto se centró en generar dos imágenes, una de desarrollo y otra de producción. Al ser el SAR un sistema monolítico, cada imagen contiene todo el sistema y sus dependencias, por lo que no es necesario instalar nada más que las imágenes de Docker.

Se generó un archivo Dockerfile que contiene las instrucciones necesarias para construir la imagen. Se divide en tres etapas que pueden ejecutarse en paralelo. La primera se basa en Node 8.7, para instalar todas las dependencias del frontend declaradas en el archivo de configuración de Bower. La segunda realiza el mismo proceso con Poetry, basándose en Python 3.10.4 y guardando las librerías utilizadas en el backend. La tercera etapa descarga librerías usadas en el runtime y descarga las dependencias de las primeras dos etapas. De esta forma se tiene el sistema SAR completamente funcional, pero alojado en una imagen ultra liviana de Python 3.10.4 (versión Alpine) y que posee lo más mínimo para funcionar.

Tener una imagen liviana del SAR permite que el despliegue de la aplicación sea más rápido y pueda transportarse a cualquier lugar, sabiendo que funcionará de la misma forma, reduciendo además tiempos de espera en caso de necesitar reiniciar la aplicación o mejorar la versión de la misma. La imagen de desarrollo pesa 271MB y la de producción pesa 186MB, cualquier máquina que pueda ejecutar Docker y disponga de este espacio, tendrá a su disposición una versión funcional del SAR corriendo un solo comando.

3.2. Actualización de tecnologías y librerías

Como se mencionó, el sistema legado cuenta con tecnologías y librerías con versiones muy desactualizadas, lo cual es siempre considerado un problema de seguridad. Dentro de los objetivos específicos de la memoria se encuentra el actualizar las tecnologías del sistema, para lo cual se realizaron los cambios que se indican a continuación.

3.2.1. Actualización del backend

La decisión de subir la versión de Python se tomó debido a la necesidad de actualizar Django. La versión que se utilizaba era la 1.11.3 que fue liberada en julio del año 2017 y deprecada en el año 2020. Por esta razón, se decidió actualizar Django a su versión soportada

más actual como lo es la 4.1, pero como esta versión no funciona en Python 3.5, se decidió subir también la versión de este último a la 3.10, que seguirá recibiendo soporte hasta el año 2026 [8].

La versión 4.1 de Django no es LTS y será soportada hasta diciembre del año 2023 [9], pero se optó por este cambio debido a que la futura versión 4.2 LTS está pronta a llegar (abril 2023) y tendrá 3 años de soporte. Se consideró mejor que el trabajo a futuro actualice prontamente la librería de una versión minor (4.1 -> 4.2) que de una versión major (3.2 -> 4.2), considerando el trabajo que conllevan estas actualizaciones.

No hubo mayores complicaciones en el proceso de actualización, los cambios máximos fueron el cambio de nombre de una función por otra, pero no afectó el funcionamiento del sistema.

3.2.2. Actualización del frontend

El frontend del SAR es un caso particular que debe ser estudiado con mucho detalle, ya que se trata de una aplicación que fue desarrollada en AngularJS (no confundir con Angular), una tecnología creada por Google en el año 2010 y discontinuada en enero de 2022 [5]. Debido a que actualizar la versión del framework no es una opción, se debe optar por un cambio radical de tecnología, a una más reciente y con un soporte duradero. El estudio, diseño y futura aplicación de una nueva tecnología frontend en el SAR involucraría cambios que se encuentran fuera del alcance de esta memoria (inclinada por el lado documental e interviniendo el contexto del sistema, pero con las menores intervenciones posibles al código) por lo que se decidió mantener el frontend en AngularJS.

Al ser Bower un gestor de paquetes deprecado, varias de las librerías hosteadas podrían seguir actualizándose en otro gestor, dejando de funcionar en el SAR. Por esta razón, se decidió fijar a versiones específicas de las librerías utilizadas en el frontend, para evitar que se actualicen y dejen de funcionar en el sistema. Esta solución temporal permitirá que el sistema siga funcionando sin problemas, pero no se podrá aprovechar de las mejoras que se realicen en las librerías utilizadas.

3.2.3. Actualización de la base de datos

La base de datos del sistema legado utiliza Postgres 9.6, la cual es una versión que ya no recibe soporte desde el año 2021 [10]. Si bien nunca se llegaron a realizar cambios en la base de datos, en el ambiente de desarrollo con Docker se levanta una base de datos con Postgres 13.3, esta es la versión actual que utiliza el Área de Desarrollo del DCC y es la que será utilizada para la migración del sistema legado.

Con la nueva versión en mente, se decidió trabajar en el sistema legado con la versión 13, para evitar problemas de compatibilidad en el futuro. Esta actualización no implicó mayores complicaciones, ya que la versión 13 es compatible con la 9, por lo que no se tuvo que realizar ningún cambio en el código. La migración de los datos puede realizarse sin problemas y se generó un script para apoyar en esta tarea.

3.3. Descripción de la API de Datos

Parte de los objetivos del trabajo de memoria, es hacer que el sistema cuente con una API que permita la integración con otros sistemas, facilitando el consumo de datos del SAR. Para esto, se decidió utilizar la librería Django REST Framework, la cual permite generar APIs de forma rápida y sencilla. Esta librería no se encontraba en el sistema legado, por lo que se decidió agregarla a las dependencias del proyecto para poder cumplir con este objetivo.

3.3.1. Interfaz de sólo lectura

Como se mencionó, la primera versión de la API no hacía uso de la popular librería Django REST Framework, y solo se entregaban vistas directas en el sistema, pensadas particularmente para acciones que se realizan en el frontend. Estas vistas además, realizan demasiadas instrucciones por vez y esto resulta ser problemático a la hora de saber qué está haciendo cada consulta por detrás. Ejemplo de esto es el endpoint que permite realizar la reserva de una sala. Este endpoint posee 46 líneas de código y realiza otras consultas más a la base de datos, llamando a funciones de utilidad que son aún más largas. Es necesario disponer de endpoints y funciones que cumplan una sola responsabilidad.

En el proceso de negociación de la solución con los actores involucrados directamente con el SAR, se acordó que la nueva API (versión 2) será, en un inicio, solo de lectura, sin poder ejecutar otras acciones. Con esto en mente, se generaron vistas en Django para obtener información de cada uno de los nueve modelos de la base de datos, y se agregaron restricciones para que solo usuarios autenticados puedan acceder a la API. También, se puede filtrar la búsqueda en algunos modelos para facilitar la misma y se protegen campos sensibles como algunos datos de los usuarios. Esta API se encuentra disponible en la URL /api/v2/.

Como esta nueva API es solo lectura, fue necesario mantener la versión anterior, que se encuentra disponible y activa, para que el frontend que la consume pueda seguir funcionando sin problemas y no deban hacerse cambios abismales en su código.

3.3.2. Documentación de la API

La documentación de la API del sistema legado es nula y, por lo mismo, se dificulta el proceso a la hora de extender el sistema con nuevas funcionalidades. Por esto, la nueva versión de la API contará con su documentación correspondiente, la cual se generará utilizando la librería *drf-yasg*. Esta librería permite generar documentación de forma automática, a partir de los modelos y vistas creados en Django que se especifiquen. Esta documentación se encuentra disponible en la URL /api/v2/docs/.

Para la visualización de la documentación se utiliza Swagger UI, la cual permite probar las distintas llamadas a la API de forma sencilla, además de permitir ver el tipo de datos de entrada y de salida de cada una de ellas. Además, se puede ver la información de los modelos y sus campos, lo que facilitará el proceso de desarrollo de la API.

3.4. Indicaciones para el despliegue del Sistema

Como parte de los requisitos pedidos por el DCC para la integración del sistema, se pide generar indicaciones adecuadas para el deployment. Las instrucciones del sistema legado quedaron obsoletas dada la migración a Docker, a continuación se enumeran las indicaciones del sistema anterior y del nuevo sistema.

3.4.1. Sistema legado: Instrucciones para levantar el sistema en desarrollo

Para el sistema legado los pasos eran los siguientes:

1. Copiar y renombrar el archivo de configuración del proyecto en Django.
2. Instalar dependencias de Python en la máquina.
3. Instalar dependencias de Bower en la máquina.
    a) (Opcional) Migrar un dump a la base de datos: este paso se realizaba con un script automático que se ejecutaba en la máquina.
4. Inicializar el sistema: Esto comprendía realizar las migraciones, crear un superusuario, cargar algunos fixtures y, en caso de error, realizar un flush de la base de datos.
5. Levantar el servidor de desarrollo de Django en la máquina.

Estos pasos fueron diseñados en el año 2017 y no se habían actualizado desde entonces, cuando fueron replicados en el año 2022 en una máquina personal de un desarrollador, se encontraron con que no funcionaban. Si bien el sistema podía ser levantado, el frontend no funcionaba correctamente y todo intento de comunicación con la API del backend fallaba. Esto se debió a que con el paso del tiempo, algunas librerías se actualizaron y se rompieron algunas funcionalidades que construían los archivos estáticos del frontend y no se podía avanzar de la pantalla de login.

Un parche temporal a la solución anterior fue fijar las versiones de las librerías que se encontraban en conflicto, pero esto no era una solución a largo plazo, ya que seguían desactualizadas. Este no era el único problema, dado que las herramientas para instalar las dependencias (Pip para Python y Bower para AngularJS) también cuentan con actualizaciones nuevas y no se podía garantizar que funcionaran en una máquina u en otra, incluso entre sistemas operativos distintos. Por esto, se decidió migrar el sistema a Docker, para que el sistema pueda ser levantado en cualquier máquina con Docker instalado.

Respecto a las variables de entorno, estas no forman parte de las indicaciones de levantamiento. Sin embargo, si ninguna es establecida, el sistema arrojará errores y no se levantará. Estas eran relativas a la conexión de la base de datos y el modo de ejecución del sistema (desarrollo o producción).

3.4.2. Sistema nuevo: Instrucciones para levantar el sistema en desarrollo

Se utiliza un archivo Makefile para facilitar el proceso de levantamiento del sistema. Este archivo contiene los siguientes comandos:

- **make dev**: Levanta el sistema en modo desarrollo. Remueve cualquier contenedor que se haya estado ejecutando con esta aplicación. Se reinicia la información de la base de datos, luego levanta el sistema, ejecuta un script que carga un dump preexistente a la base de datos y realiza las migraciones correspondientes. Si se realizan cambios en archivos estáticos o en el backend, estos son reconocidos y no es necesario reiniciar el sistema.

- **make prod**: Realiza el mismo proceso, pero sin cargar datos ni realizar migraciones. Luego de levantar el sistema, colecciona los archivos estáticos generados y los comprime para reducir el tamaño de descarga de cada vista de la aplicación. Es importante destacar que esta compresión a la vez ofusca los scripts, por lo que no se puede acceder a ellos directamente.

- **make migrations**: Genera las migraciones correspondientes a los cambios realizados en los modelos de Django. Se debe ejecutar cada vez que se modifiquen los modelos.

- **make migrate**: Aplica las migraciones encontradas en el proyecto, para así sincronizar los datos de la base de datos con los modelos de Django. Este comando es llamado por "make dev".

- **make loadfixtures**: Carga al sistema las fixtures predefinidas del proyecto. Esto puede ser salas existentes, tipos de reservaciones, grupos de usuarios, etc...

- **make loaddump**: Ejecuta un script en el contenedor de Postgres, para así cargar los datos del dump que se encuentra en una carpeta definida del repositorio. Este comando es llamado por "make dev". Más instrucciones se encuentran en el README del proyecto.

- **make createsuperuser**: Crea un superusuario para acceder al panel de administración del sistema en Django.

- **make down**: Detiene el sistema, elimina los volúmenes creados para la persistencia de datos y las redes internas creadas por Docker.

Estos comandos facilitan el proceso de levantamiento del sistema, ya que no es necesario conocer los comandos de Docker para realizar las tareas. Típicamente, se espera que un desarrollador utilice el único comando "make dev"para empezar a intervenir en el SAR o testearlo.

Respecto a las variables de entorno, estas se especifican en las instrucciones y se encuentran en un archivo de ejemplo, estas guardan relación con la conexión con la base de datos, con variables de producción como llaves secretas y un apartado especial para la autenticación con el sistema externo con el Portal DCC que se define en la siguiente sección.

3.5. Autenticación de Usuarios

El sistema de autenticación del SAR también fue ajustado para cumplir con los lineamientos del DCC. Como en ellos se indica: *"Todos los sistemas deben autenticar usuarios utilizando los servicios que provee el DCC. Cada sistema debe implementar su propio esquema de roles y permisos (Autorizaciones)"* (Ver Anexo I).

Del análisis previo realizado al SAR se desprende que este contaba con su propio sistema de autenticación de usuarios basado en Django, y se gestionaba a través de un backend interno. Para cumplir el requisito, se extendió este sistema con el uso de una aplicación SSO desarrollada en Django por el departamento [11], esta permite comunicarse con el Portal DCC y U-Pasaporte y provee todas las herramientas necesarias para una integración simple y directa con cualquier proyecto, pues se configura a través de variables de entorno y de la exposición de un endpoint en particular que fue configurado para tal propósito.

La adición del Portal DCC como sistema externo al proyecto implica un cambio en los diagramas de contexto y de contenedor del SAR, los cuales se muestran en la Figura 7 y Figura 8 respectivamente.

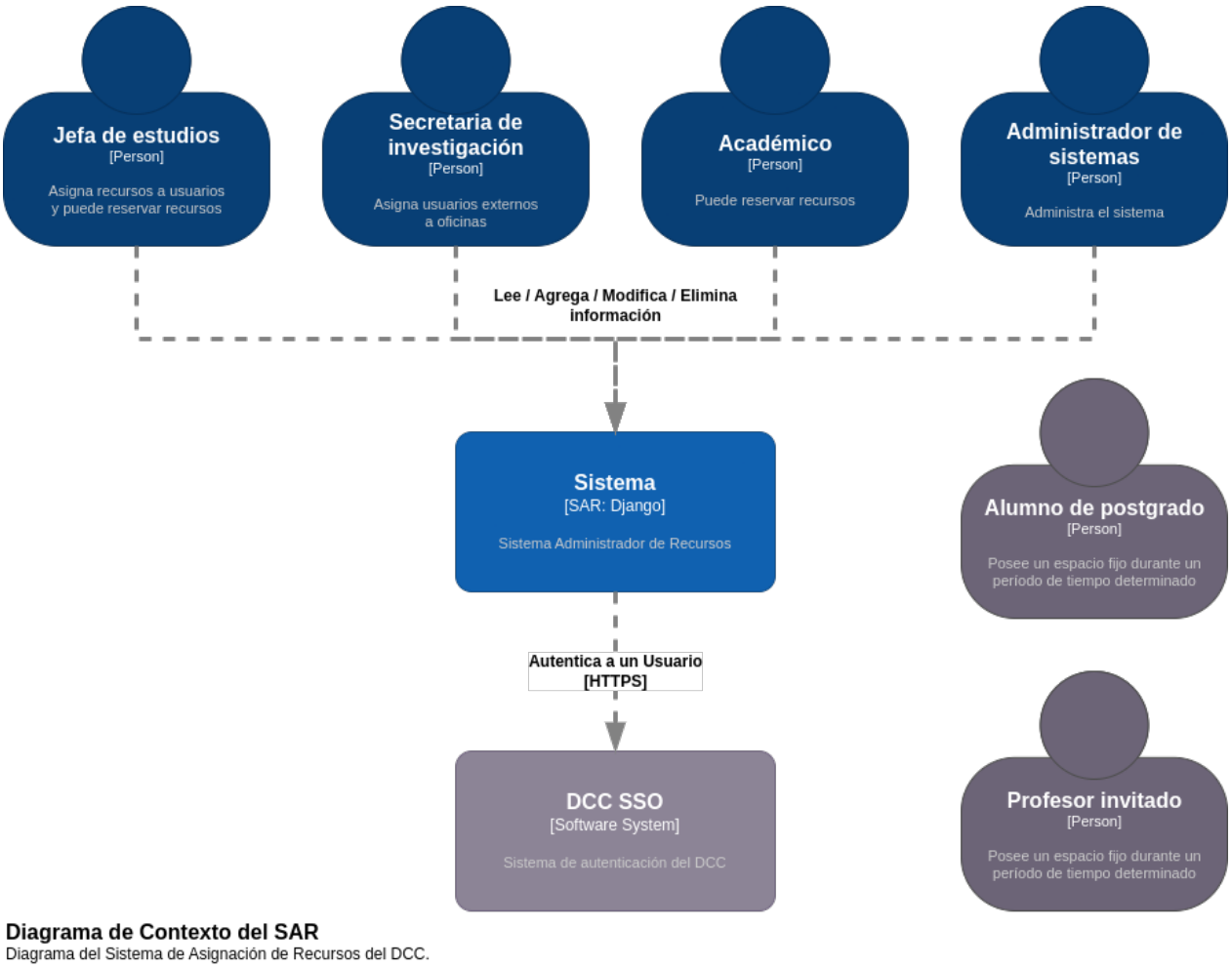

Figura 3.1: Nuevo Diagrama de Contexto del SAR

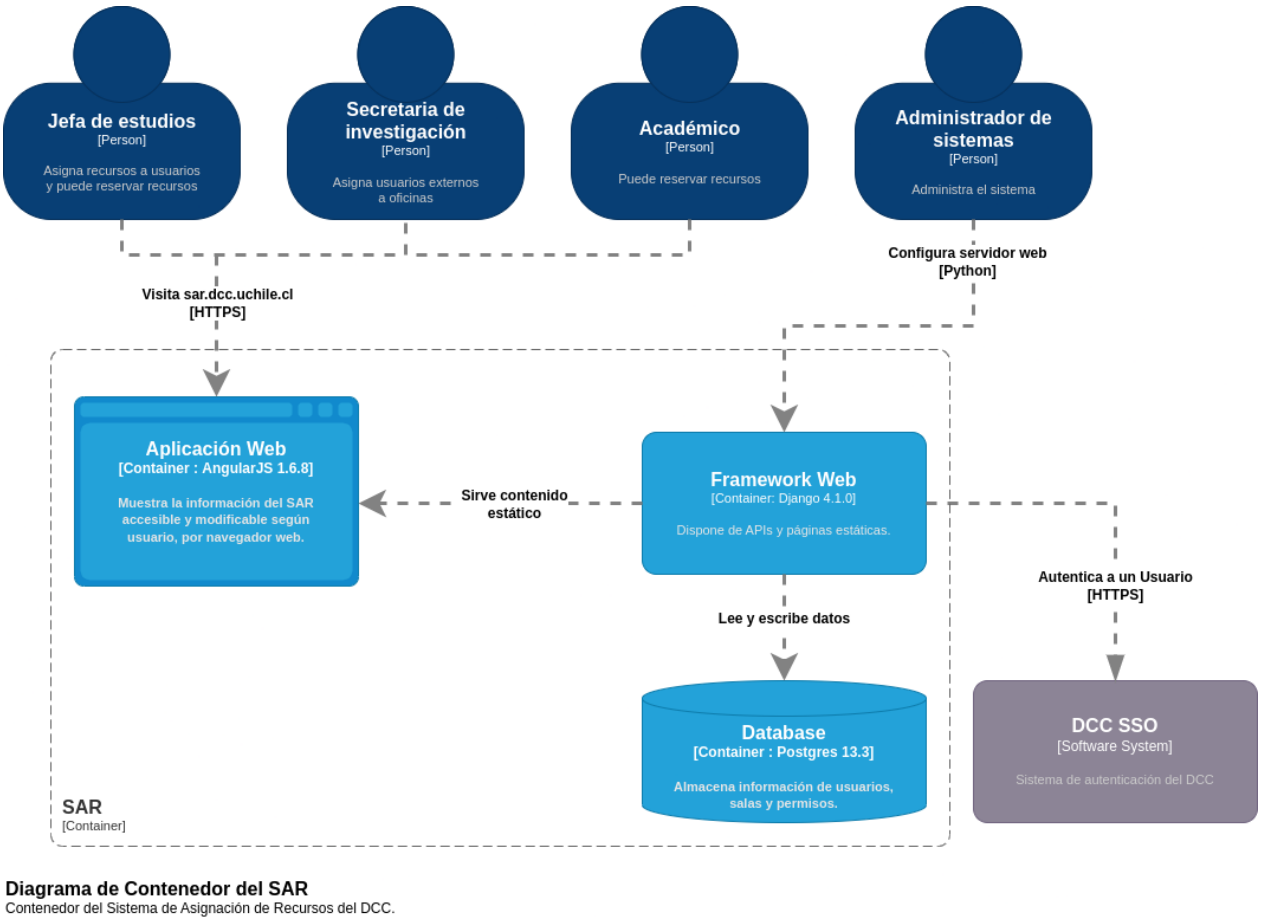

Figura 3.2: Nuevo Diagrama de Contenedor del SAR

3.6. Modelo de Datos

El modelo de datos también tuvo ajustes necesarios para su integración con el ecosistema del DCC. En particular se hicieron dos cambios relativos a la tabla *UserInformation*:

- **Se agrega el atributo rut**: Este es un string de 12 caracteres que representa el RUT de una persona, sin dígito identificador. Por ahora este atributo puede ser nulo o ser un string vacío, esta decisión fue tomada porque no todos los usuarios del SAR tienen un RUT asociado. Se solicitó una lista de nombres y RUTs de los usuarios registrados pero no se logró completarlos a todos, además, en el SAR existen usuarios abstractos como el Centro de Alumnos del DCC o un usuario de Reservas Externas.
- **Se elimina el atributo *first_login***: Este atributo se utilizaba para determinar si un usuario era nuevo, para así redireccionarlo a un formulario para ingresar parte de su información personal. Como esta información vendrá desde el Portal DCC, este atributo ya no es necesario. Los usuarios que deseen realizar acciones en el sistema y no tengan información asociada, deberán ingresar a la aplicación pero debiéndose autenticar con el Portal DCC (más precisamente desde U-Pasaporte) para registrar su información. Un mensaje de aviso aparecerá luego de iniciar sesión (Ver Anexo IV.2).

Ninguna otra tabla fue modificada, la nueva tabla del modelo de datos puede verse en el Anexo III.

3.7. Repositorio del Código Fuente

El sistema legado ya cuenta con un repositorio en GitHub que versiona su código fuente, todo el desarrollo realizado en el proyecto se encuentra en este repositorio, solo los usuarios con permisos pueden acceder a él. Todo el trabajo realizado en esta memoria se encuentra en una rama (memoria2022-sz) diferente de la principal, cada hito merecía la creación de una nueva rama y un posterior Pull Request a la rama del memorista para combinar las intervenciones de código necesarias.

Los ajustes del repositorio y estructura del código fuente sufrieron cambios menores:

- Reestructuración de carpetas para funcionar de mejor manera con Docker.
- Adición del submódulo de GitHub que contiene el código fuente de la aplicación SSO.
- Reemplazo total del archivo README.md para entregar documentación del deployment y otras informaciones de utilidad.

Capítulo 4

Extensión al Sistema SAR

Gracias a las múltiples mejoras en aspectos específicos del sistema y a la generación de documentación, siempre en base a los requisitos del DCC, es posible el diseño de nuevas extensiones de manera rápida y eficiente. Como se mencionó en los objetivos de la memoria, la implementación de un servicio que extienda las funcionalidades del SAR, puede evidenciar el progreso de este último, en términos de gobernabilidad e integración. Es por esto que, a efectos de cumplir este objetivo, se implementó un nuevo servicio que provee métricas de uso de los espacios del departamento.

El servicio implementado es una sección de gráficos estadísticos, a modo de dashboard, que permite recoger y entregar información de valor proveniente del mismo sistema, de forma precisa, medidas históricas del uso de los espacios. Con esto, se entrega una mejor interpretación de cómo estos recursos son aprovechados, generando oportunidades de mejora en la gestión de los mismos.

En concreto, esta nueva sección se aprovecha de la extensión del servicio de APIs del sistema y del rápido despliegue (sea en un entorno de desarrollo o de producción) del mismo. La extensión implementada considera la desagregación de la información entregable por parte de la nueva API, además que fue desarrollada considerando la nueva documentación vía Swagger que indica cómo los datos son accesados.

4.1. Necesidades de la Extensión

Para desarrollar un gráfico estadístico con métricas útiles, se deben considerar los siguientes puntos:

1. Se desea resolver una pregunta; por ejemplo, ¿cuánto se usa el auditorio Picarte?
2. Se necesita comprender qué datos se necesitan para resolver la pregunta.
3. Se necesita acceder a los datos pertinentes, cumpliendo con los requisitos del sistema para dicho acceso.
4. Se necesita tener una forma de procesar la información entregada por la API de datos.
5. Se necesita tener una forma de visualizar la información procesada.

6. La extensión necesita pertenecer al mismo sistema. Esto también significa pertenecer al mismo stack tecnológico.

7. Al ser una extensión del sistema, se necesita contar con una forma de desplegar los nuevos cambios.

Con esto en mente, se generaron gráficos estadísticos que resuelven dos preguntas: "¿Cuáles son las salas más y menos reservadas?" y "Para una sala, ¿en qué hora, día o mes es más utilizada?. A continuación se muestra el resultado de ambos gráficos.

4.2. Estadísticas de Uso de Salas

Para acceder a la vista de estadísticas, se introdujo una nueva sección en el menú de navegación del SAR: Estadísticas. Este menú puede verse en la Figura 9 y despliega las opciones de dos vistas: 1) frecuencia de reservas por sala y 2) frecuencia de reservas por horario.

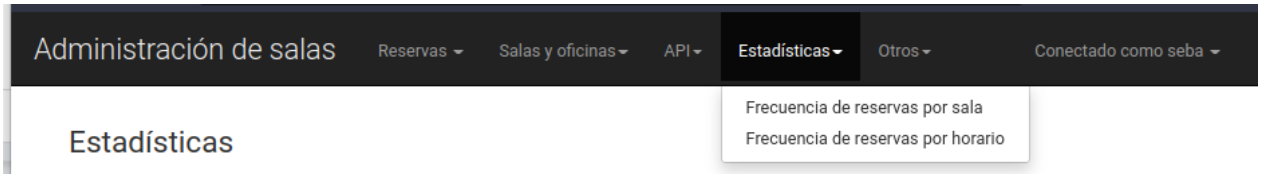

Figura 4.1: Menú de navegación del SAR con la nueva sección de Estadísticas.

Esta extensión fue desarrollada utilizando AngularJS 1.6.8, la versión que se encuentra deprecada, porque si bien es cierto que se debe migrar a un framework estable. Esta nueva funcionalidad es una prueba de concepto que busca mostrar la facilidad de extender el sistema con las mejoras y ajustes realizados, y poder así evaluar la gobernabilidad del mismo.

4.2.1. Frecuencia de Reservas por Sala

Esta vista muestra la cantidad de reservas realizadas en cada sala del departamento, desde el 1 de enero de 2014 hasta el 1 de enero de 2023. La información se muestra en forma de gráfico de barras, la altura de estas es proporcional a la cantidad de reservas realizadas. Al pasar el cursor sobre una de las barras, se muestra la cantidad de reservas realizadas en esa sala. La Figura 10 evidencia algunos resultados de utilidad, como por ejemplo, las salas más y menos pedidas a lo largo del tiempo. Cabe destacar que existe información desde el 2014.

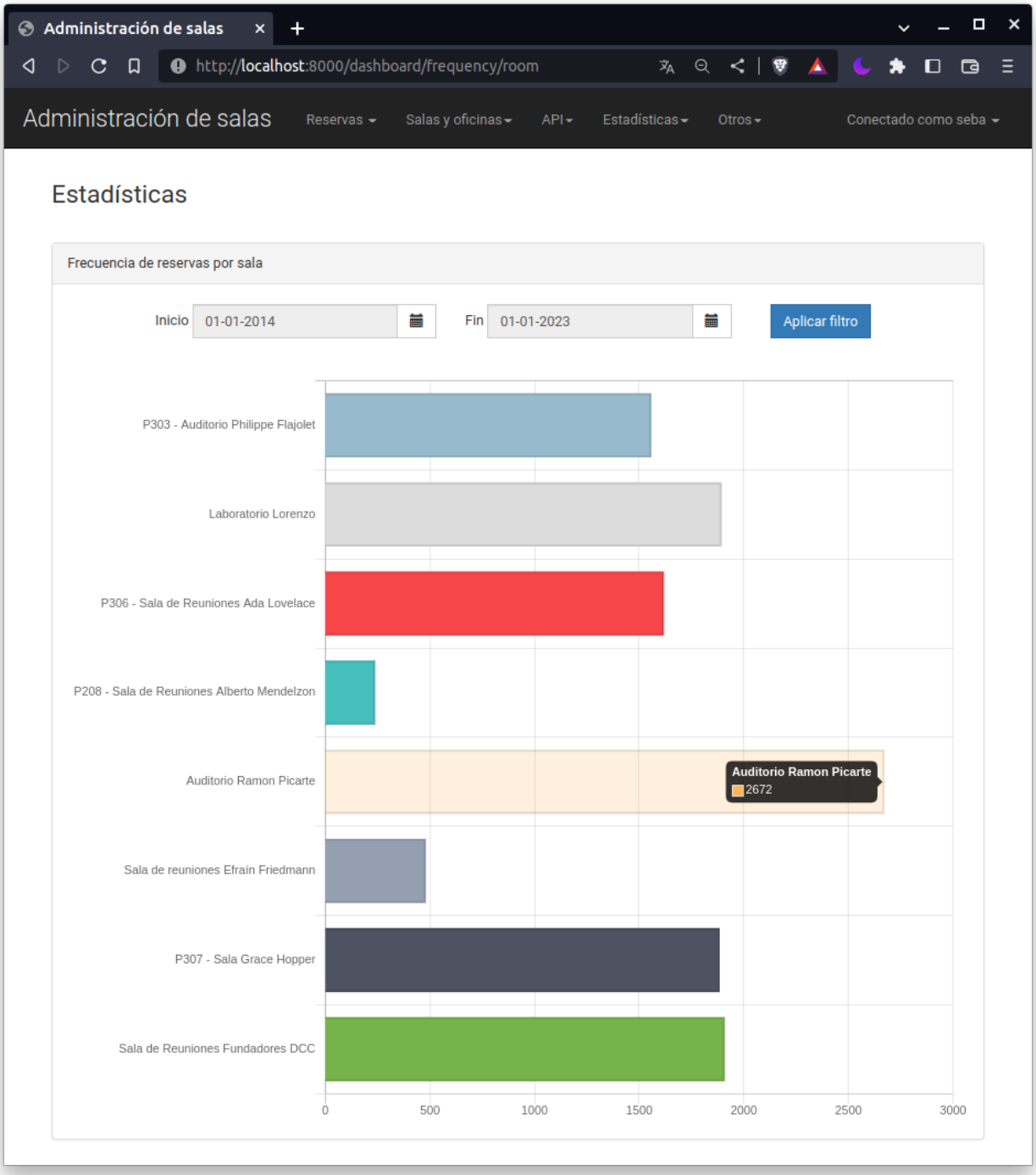

Figura 4.2: Frecuencia de reservas por sala. El cursor está sobre la barra del Auditorio Ramón Picarte.

Se consulta la API para obtener la información de todas las salas del departamento, cabe mencionar que se filtra por los espacios que son efectivamente salas, las oficinas no son consideradas para esta extensión. Luego se itera sobre cada una de las salas, para obtener la cantidad de reservas realizadas en ellas en el intervalo de tiempo especificado. Luego, nuevamente gracias a la API, se filtra por las reservas que comienzan dentro del rango de fechas para obtener los datos que expone el gráfico.

4.2.2. Frecuencia de Reservas por Horario

Esta vista muestra la cantidad de reservas realizadas por intervalo de tiempo de una sala seleccionada, puede seleccionarse un horario diario, mensual o anual, y el calendario se ajusta automáticamente para elegir el intervalo de tiempo correcto. Al pasar el cursor sobre una de las barras, se muestra la cantidad de reservas realizadas en ese intervalo de tiempo.

En la Figura 11 se aprecia poco uso del Laboratorio Lorenzo en el año 2022. En el Anexo IV se muestran los resultados que coinciden con lo encontrado en esta vista, en cuanto a la existencia de la reservación de agosto.

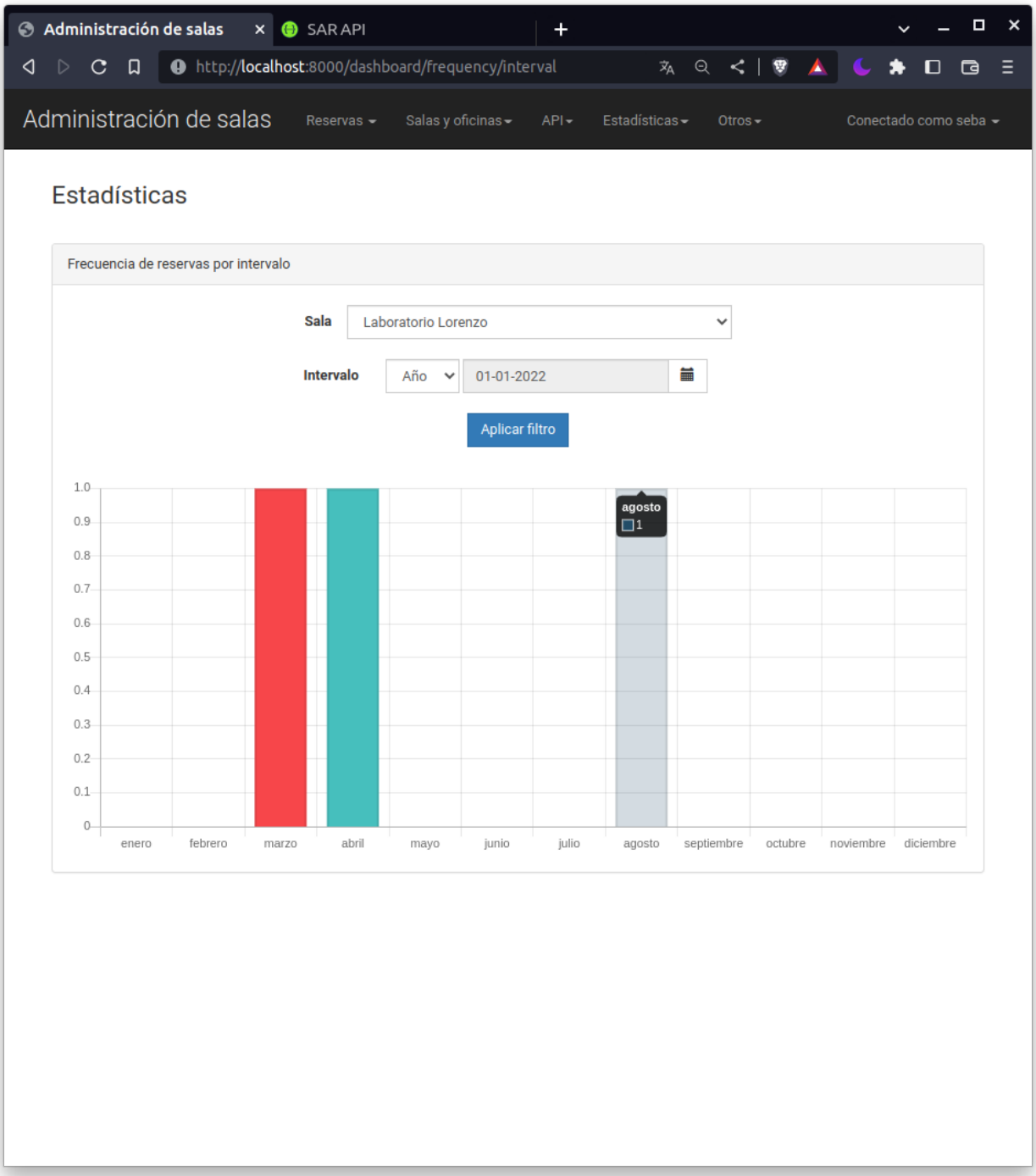

Figura 4.3: Frecuencia de reservas por horario. El cursor está sobre la barra del mes de agosto.

4.2.3. Respondiendo las preguntas de utilidad

Particularmente, respecto a *¿cuáles son las salas más y menos reservadas?*, esta pregunta se responde con la vista Frecuencia de reservas por sala. En el caso global desde el año 2014, gracias a la Figura 10 se puede ver que la sala más reservada es el Auditorio Ramón Picarte, mientras que la menos reservada es la Sala de Reuniones Alberto Mendelzon. Estos resultados son históricos del SAR, pero también se puede responder la misma pregunta para

un intervalo de tiempo en específico.

Por otra parte, para una sala, *¿en qué hora, día o mes es más utilizada?*, esta pregunta se responde con la vista Frecuencia de reservas por horario o intervalo. Se escoge una sala y se selecciona el intervalo de tiempo que se desea consultar, además de la periodicidad. En la Figura 11 se puede ver que el Laboratorio Lorenzo ha sido poco reservado a lo largo del año 2022.

4.3. Tablero de Producto

La Figura 12 que viene a continuación muestra el tablero de la nueva versión del producto (objetivo y alcance), es decir, el sistema luego de haber sido intervenido y expendido. Posteriormente, se explican en detalle las secciones de este tablero.

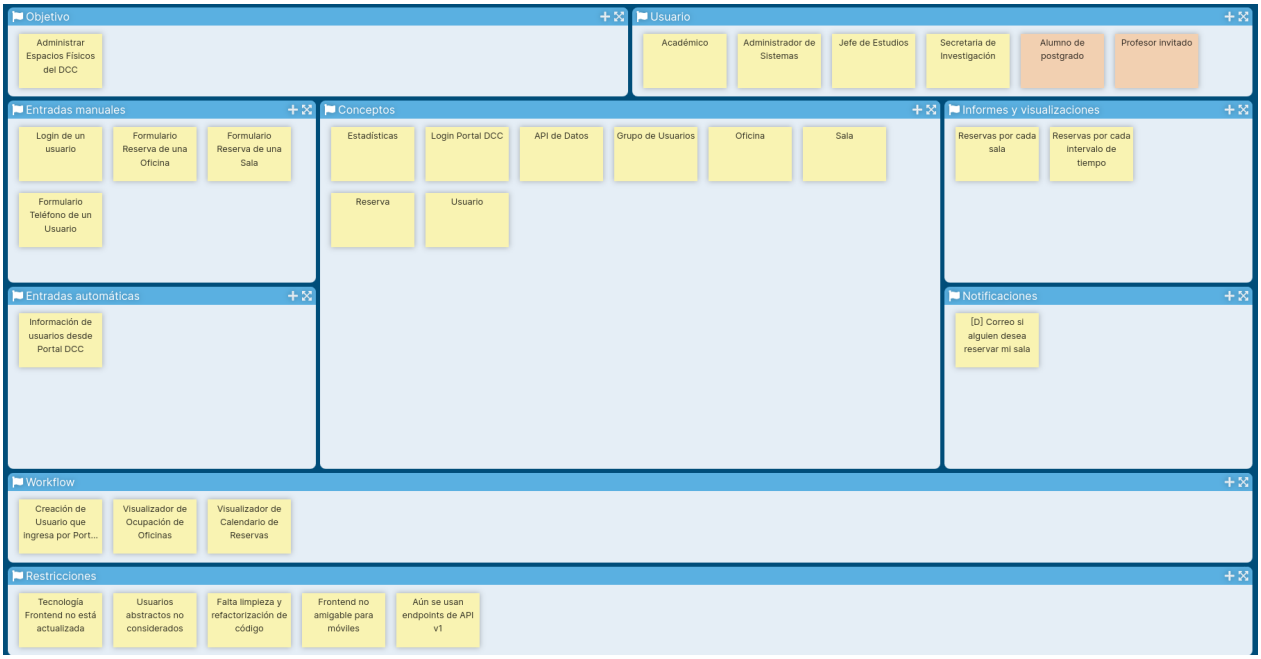

Figura 4.4: Tablero de producto del nuevo sistema.

- **Objetivo**: El SAR fue creado con el objetivo de funcionar como un sistema de asignación de recursos para el DCC. Actualmente, administra los espacios físicos del DCC mediante un sistema de reservas que realiza un usuario.
- **Usuario**: Esta sección define a los usuarios del sistema, directos e indirectos. El SAR define 4 usuarios directos que son:
  - **Académico**: Puede realizar reservas de recursos.
  - **Administrador de Sistemas**: Puede administrar todo el sistema, incluyendo el agregar o eliminar recursos del SAR.
  - **Jefe de Estudios**: Puede distribuir de forma fácil y eficiente a los alumnos de postgrado del DCC en las oficinas asignadas para este fin. En otras palabras, asignar puestos de trabajo a dichos estudiantes. Además, puede reservar recursos como cualquier otro usuario.

  - **Secretario de Investigación**: Puede asignar profesores visitantes y postdoctorandos a puestos de trabajo en las oficinas destinadas para tal fin.

  Además, se definen 2 usuarios indirectos:

  - **Alumno de Postgrado**: El sistema le permite contar con un espacio de forma fija por la duración de su estadía en el departamento.
  - **Profesor invitado o postdoctorando**: El sistema le permite agilizar el proceso de asignación de un espacio temporal durante su estadía al DCC.

- **Entradas manuales**: El SAR cuenta con 3 entradas manuales que permiten reservar una sala, asignar una oficina, o editar el número de teléfono de contacto propio de un usuario.

- **Entradas automáticas**: El SAR cuenta con 1 entrada automática que se implementó en la nueva versión, y guarda relación con el Portal DCC, el cual provee información personal sobre los usuarios que se autentican mediante el sistema de U-Pasaporte.

- **Informes y visualizaciones**: El SAR ahora permite saber estadísticas de uso de sus salas mediante dos gráficos estadísticos de frecuencia de reservas. Estas son por sala, o por intervalo de tiempo para una sala en particular. Por otro lado, el SAR también permite visualizar un calendario de reservas para una sala y un horario en particular, además de poder desplegar información relacionada a una reserva o a las oficinas asignadas.

- **Notificaciones**: En la versión legada del SAR, existía una funcionalidad que enviaba un correo a un usuario que tenía reservada una sala, esto ocurría cuando otro usuario deseaba reservar la misma sala en el mismo horario. Esto hacía uso de la librería de correo SMTP de Python, y utilizaba un servidor de correos del DCC. La funcionalidad vive en el código, pero no se usa, por lo que se marcó deprecada con [D].

- **Conceptos**: Los conceptos fundamentales asociados al proyecto son las Salas, las Reservas y los Usuarios. A partir de estas tres definiciones se construye un sistema que además puede entregar información de utilidad mediante una API de Datos, o autenticar usuarios mediante una API de Autenticación.

- **Workflow**: Los flujos de trabajo del SAR son los siguientes:

  1. Un usuario edita su información personal.
  2. Un usuario reserva una sala o asigna una oficina.
  3. Un usuario se autentica en el sistema por medio de su Username/Contraseña o RUT/Contraseña, o mediante U-Pasaporte.

- **Restricciones**: Las restricciones actuales del nuevo sistema tienen que ver principalmente con los ajustes y mejoras desde la versión anterior al frontend, pues no se actualizó el framework que este utiliza, además de seguir usando endpoints de la API v1. Por otro lado, la implementación de RUTs de los usuarios hace que el sistema sea más restrictivo, pues no se tienen considerados a los usuarios abstractos como grupos o comunidades del DCC. Finalmente, el frontend no es amigable para dispositivos móviles y es necesaria una refactorización del código.

4.4. Inclusión de Autenticación U-Pasaporte en el SAR

Como se mencionó en los Ajustes al Sistema SAR, se incluyó la autenticación U-Pasaporte en el SAR, gestionado por el Portal DCC. Se extendió el sistema al generar dos nuevas formas de ingreso al sistema (además del clásico username y contraseña):

- *RUT y contraseña*: Esto fue pensado para personas que desconocen su username pero pueden tener RUT registrado; también para personas que no tienen cuenta en U-Campus y les crearon una cuenta en el SAR. Además, se espera reemplazar el username por el RUT en el futuro, por lo que esta forma de ingreso será útil.
- *U-Pasaporte*: Esta forma de ingreso debe ser la más común, ya que se pretende utilizar en todos los sistemas del DCC. Las vistas y el flujo para un usuario que ingresa por esta vía puede verse en el Anexo V.

Para conectar la aplicación del Portal DCC con el SAR, se siguieron las instrucciones del repositorio django_sso [11] mencionado anteriormente. Allí se especifica la necesidad de contar con un endpoint en el SAR que redirija a la vista de login del portal.

El Portal DCC recibirá algunos datos del SAR como el host de donde proviene la petición o el nombre de la aplicación que se está intentando autenticar. Por otro lado, deben agregarse nuevas rutas pertenecientes al SSO del DCC, estas serán las encargadas de comunicar la información del usuario que desea entrar al sistema. Estas comienzan con el path /sso.[ei]ncluyen cuatro endpoints: /, /login, /logout y /unauthorized.

La Figura 13 presenta un cambio menor al panel de login del SAR para permitir el ingreso vía U-Pasaporte. Para acceder a esta vista, se debe ingresar al sistema mediante el path /login. Cabe mencionar que si se ingresa al index principal del SAR, sin estar autenticado, el sistema redireccionará de inmediato al Portal DCC.

Se acordó que si un usuario no existe en el sistema (porque su RUT no fue encontrado), entonces al ingresar mediante la cuenta de U-Campus se le creará una cuenta en el SAR. Este usuario no posee permisos para interactuar con el sistema, a menos que un administrador lo asigne a un grupo correspondiente, con permisos asociados a este grupo.

El endpoint /sso/login es el encargado de recibir la información desde el Portal DCC. Esta información incluye datos sensibles como el RUT del usuario y su nombre completo, por lo que se debe asegurar que la comunicación entre el Portal DCC y el SAR sea segura. El sistema en producción usará HTTPS para esta comunicación, pero para el desarrollo se puede usar HTTP.

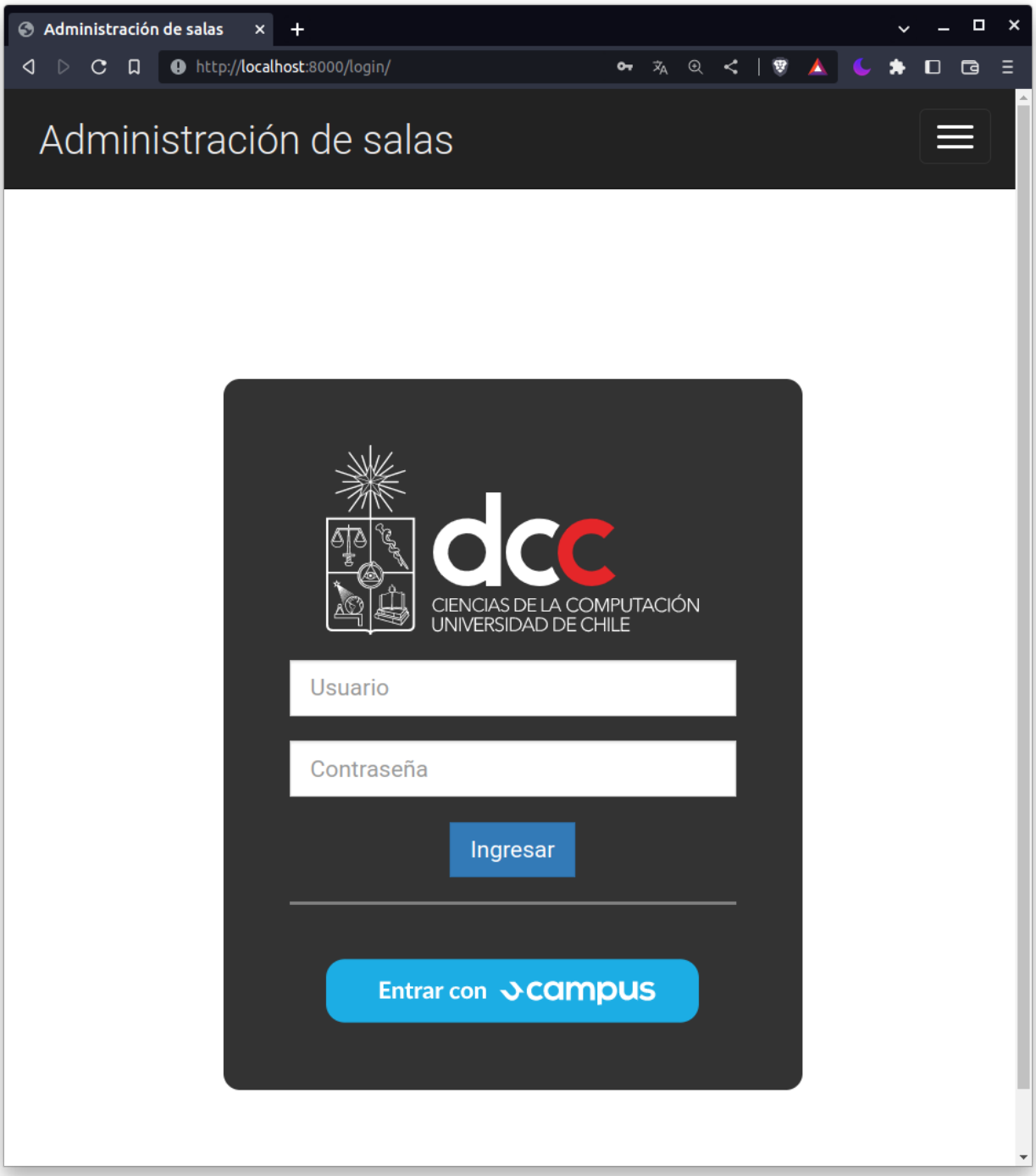

Figura 4.5: Login estilizado. Se le agregó recuadro oscuro, logo DCC y botón para entrar por U-Pasaporte.

Cabe recordar que los ajustes al modelo de datos provienen de una directa implicancia de la adición de este sistema de autenticación que buscará ser el principal dentro del sistema. Como se obtiene la información personal de un usuario desde el Portal DCC, se creará uno nuevo en caso de no estar registrado. Debiendo almacenar el *RUT*, y eliminando el atributo *first_login* que velaba por que la información estuviera completa.

4.5. Especificación de la nueva API del SAR

La API extendida del SAR permite a los desarrolladores de otras aplicaciones acceder a la información del sistema, principalmente las salas y sus reservas. Esta se encuentra en el path /api/v2ƨ se encuentra accesible para cualquier usuario autenticado. Cabe recordar que la API del SAR original seguirá operativa en el path /api/v1". La API v2 se divide en tres partes (autenticación, documentación y acceso a los recursos del sistema) e incluye los siguientes endpoints:

- **/api/v2/auth/login**: Este endpoint permite a un usuario autenticarse en el SAR. Para ello, se modificó ligeramente la vista original que se encarga de autenticar a un usuario por API. Como se mencionó anteriormente, esta versión del SAR también permite ingresar el RUT de la persona junto con su contraseña, esto a modo de transición hacia un sistema de autenticación por RUT o por U-Pasaporte vía Portal DCC. La Figura 14 muestra el diagrama de flujo de la adición del RUT como forma de autenticación.

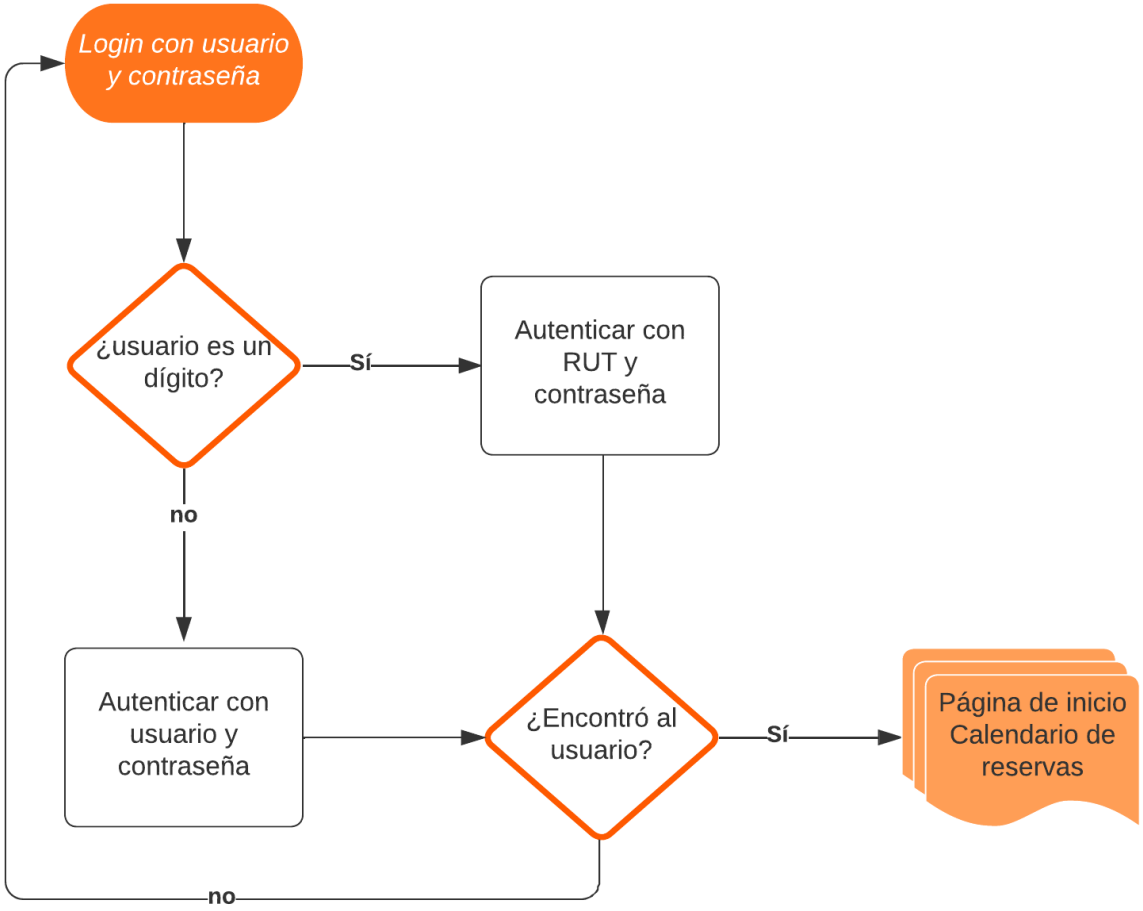

Figura 4.6: Diagrama de flujo del login por API.

- **/api/v2** y **/api/v2/docs**: Estos endpoints permiten acceder a la documentación de la API usando Swagger. Esta documentación incluye información sobre los endpoints

de datos, los parámetros que pueden recibir y los datos que retornan. Además, permite probar los endpoints directamente desde la página web. También puede accederse desde el menú de navegación del SAR. Algunas imágenes de referencia son encontradas en el Anexo VI.

- **/api/v2/[recurso]**: El sistema pone a disposición 9 endpoints relacionados a cada uno de los 9 modelos que pertenecen al sistema legado. Estos endpoints consultan los datos de cada modelo y pueden ser filtrados por id, este debe añadirse al final del path. Se especifican a continuación:

  - **/group**: Retorna los grupos existentes a los que puede pertenecer un usuario, estos son: Visitas, Postgrado, Investigación, Funcionarios, Milenio, Académicos, Estudiantes de Postgrado, Aplicaciones Externas.
  - **/user/information**: Retorna la información extra de un usuario, además de la que guarda Django.
  - **/user**: Retorna los datos de un usuario de Django, excepto su contraseña.
  - **/room**: Retorna los datos de un espacio físico. Esta puede ser oficina o sala.
  - **/room_group**: Retorna los datos de un grupo de salas. Estas son tres: "Sala Compartida (Auditorio, Sala Reuniones)", "Oficina Permanente (Académicos y Funcionarios)" y "Oficina Temporal (Estudiantes, Visitantes, Postdoc)".
  - **/reservation/type**: Retorna los tipos de reservas del sistema. Algunos ejemplos son: Asamblea, Clase, Examen de Título, Reunión de Trabajo, etc...
  - **/reservation/group**: Retorna los grupos de reservas, es decir, si una reserva tiene frecuencia diaria, semanal, mensual o sin frecuencia, se le asigna un grupo.
  - **/reservation/block**: Retorna los bloques de reservaciones, estos son fijos de 15 minutos cada uno, una reserva puede tener muchos bloques reservados.
  - **/reservation**: Retorna las reservas del sistema. Además del filtro por id, pueden filtrar por un usuario específico, por una sala, por reserva externa, o por fecha de inicio de la reserva.

Capítulo 5

Evaluación de la Nueva Solución

El proceso de evaluación busca demostrar que, luego de la intervención realizada al SAR, el esfuerzo para modificar o extender el sistema es menor, ya que se habrá mejorado la gobernabilidad y la capacidad de integración del mismo. Para esto, se evaluó el esfuerzo requerido para extender el sistema con los gráficos estadísticos, y se comparó el esfuerzo de cada intervención sobre el sistema legado y sobre el nuevo sistema. La misma comparación se hizo para determinar las diferencias de complejidad de las intervenciones. De esa manera se podrá ver el impacto que el trabajo realizado tiene sobre la gobernabilidad del sistema.

A partir de este estudio se confeccionará una tabla comparativa de ambas versiones del sistema, y mostrarán los esfuerzos necesarios y la complejidad de la intervención. Los valores de las variables serán inicialmente especificadas en términos conceptuales como: .ªlto, medio, bajo". Una vez que la tabla se encuentre completa, se analizará en detalle, permitiendo generar todas las observaciones y conclusiones finales que demuestren el impacto del trabajo del memorista, así como también, que ayuden a crear y mejorar otros sistemas del DCC.

5.1. Proceso de Evaluación Seguido

Para evaluar el trabajo realizado, se propuso realizar una tabla comparativa de esfuerzo y complejidad entre el sistema legado y el nuevo, al extenderlos con los gráficos estadísticos. Esto es, en conjunto con la lista enumerada de la sección 4.1 del presente informe, donde se exponen las necesidades de la extensión.

Cada ítem de necesidades tiene directa relación con más de un requisito de gobernabilidad e integración, pues algunas acciones deseadas necesitan de un sistema que sea fácil de modificar, extender y mantener, y esto es precisamente lo que buscan tales requisitos. Por lo tanto, se considera que la evaluación de los ítems es una forma experimental, de evaluar el cumplimiento y ajuste a dichos lineamientos, pero dotada de objetividad.

5.2. Resultados Obtenidos

La Tabla 2 se muestra a continuación y refleja el resultado de extender el sistema con alguno de los dos gráficos estadísticos que responden a las preguntas planteadas con anterioridad. Se evalúa entonces la complejidad o esfuerzo de desarrollar dicha extensión al sistema, con los requisitos asociados.

Tabla 5.1: Esfuerzo y complejidad de extender el SAR con un gráfico estadístico que usa datos de su API.

Necesidad a cubrir	**Requisitos de Gobernabilidad e Integración asociados**	**Esfuerzo y Complejidad Sistema legado**	**Esfuerzo y Complejidad Sistema nuevo**
Comprender qué datos se necesitan.	• Modelo de datos • API	**Medio**	**Bajo**
Acceder a los datos, cumpliendo requisitos de acceso.	• Autenticación de usuario • API	**Medio**	**Bajo**
Procesar la información entregada por la API de Datos.	• Stack tecnológico • API	**Alto**	**Medio**
Visualizar la información procesada.	• Stack tecnológico • Arquitectura del software	**Alto**	**Medio**
Desarrollo bajo el mismo sistema, mismo stack tecnológico.	• Stack tecnológico • Objetivos y alcances del sistema	**Alto**	**Bajo**
Desplegar nuevos cambios.	• Arquitectura ecosistema • Indicaciones de deployment	**Medio**	**Bajo**

A continuación, se irá analizando la tabla anterior, fila por fila, centrando la atención en el primer gráfico estadístico que responde la pregunta sobre la **frecuencia de reservas por sala**:

1. *Para dar respuesta a la pregunta, se debe listar todas las salas existentes para luego contar cuántas reservas se han hecho en cada una de ellas. Averiguar información de utilidad sobre los datos es imperante para esta tarea.*

    - **En el sistema legado** existe un modelo de datos documentado, pero no se sabe si se encuentran disponibles en la API, dado que de esta última no hay documentación.
    - **En el sistema nuevo**, el modelo de datos está disponible en la API, y además se puede acceder a la documentación de la misma.

2. *Para acceder a los datos, se debe autenticar al usuario.*

    - **En el sistema legado**, la autenticación se realiza desde el backend interno y es una tarea fácil de realizar, pero nuevamente no se sabe cómo realizar las consultas correctas.
    - **En el sistema nuevo**, la autenticación se puede realizar además por U-Pasaporte, y se puede acceder a la documentación de la API para conocer los parámetros de consulta.

3. *Para procesar la información, se debe tener conocimiento de cómo los datos son retornados una vez que se consultan. Además, este procesamiento debe realizarse en el frontend, por lo que se debe conocer el stack tecnológico que se está utilizando.*

    - **En el sistema legado**, no se sabe cómo los datos son retornados por la API, y además no existe documentación sobre el stack tecnológico.
    - **En el sistema nuevo**, se puede acceder a la documentación de la API para conocer cómo se procesan los datos y se conoce el stack debido a la documentación. Sin embargo, cabe destacar que el stack tecnológico del frontend es antiguo y deprecado, por lo que se debe realizar un esfuerzo extra para aprenderlo sin una comunidad activa.

4. *Para visualizar la información en el sistema, se debe conocer el stack tecnológico que se está utilizando y es necesario conocer la arquitectura del software para comprender cómo debe extenderse el sistema, es decir, dónde puede ir el nuevo gráfico, y quién podría accesarlo.*

    - **En el sistema legado** no se conoce el stack tecnológico y la arquitectura de software no está documentada.
    - **En el sistema nuevo**, se conoce el stack tecnológico y la arquitectura de software está documentada, pero nuevamente para el frontend, se utiliza tecnología antigua y deprecada que complica el desarrollo.

5. *Para desarrollar bajo el mismo sistema, se debe tener conocimiento del stack tecnológico, además de los objetivos y alcances del proyecto. Mientras que no existe un tablero de scoping que defina al proyecto, no se puede saber a ciencia cierta para qué se desarrolla una extensión, o si esta será de utilidad.*

- **En el sistema legado** no existe dicho tablero.
- **En el sistema nuevo** cuenta con un tablero que define el proyecto y sus alcances.

6. *Para desplegar nuevos cambios, se debe conocer la arquitectura del ecosistema del SAR y las indicaciones de deployment.*

   - **En el sistema legado** no existe documentación sobre la arquitectura del ecosistema, pero sí existen instrucciones de deployment.
   - **En el sistema nuevo**, existe documentación sobre la arquitectura del ecosistema y las instrucciones de deployment son diversas dependiendo del caso de uso.

Por lo expuesto anteriormente, se desprende que sí existe facilidad a la hora de extender el SAR, pues este nuevo sistema posee documentación actualizada sobre la API, arquitecturas definidas y stack tecnológico definido. Por lo cual, ya no es necesario revisar el código fuente en todo momento para averiguar cómo funciona el sistema. Sin embargo, se debe tener en cuenta que el stack del frontend está deprecado, lo cual puede complicar el desarrollo de nuevas funcionalidades.

5.3. Nivel de Cumplimiento de Lineamientos de Gobernabilidad

En relación a los lineamientos de gobernabilidad y capacidad de integración definidos para los sistemas del DCC, a continuación se indica el nivel de cumplimiento que tenía el sistema SAR antes de la realización de esta memoria (Tabla 1), y el cumplimiento actual (Tabla 3). Como se puede apreciar, a partir del trabajo realizado se ha mejorado la adherencia a dichos lineamientos.

El trabajo realizado fue siempre enfocado en mejorar la gobernabilidad del sistema y su capacidad de integración. Con respecto a la tabla se cumplieron 8 de los 9 ítems, la alineación del stack tecnológico con las definiciones del DCC se cumplió en parte, pues no se actualizaron las versiones de todos los frameworks del sistema. AngularJS fue el único framework, parte del frontend, que no se actualizó y esto queda como tarea pendiente. El reemplazo de AngularJS por una tecnología actualizada permitirá el cumplimiento del único ítem restante.

Tabla 5.2: Nivel de cumplimiento inicial y actual de SAR.

Ítem Evaluado	**Nivel Inicial**	**Nivel Actual**
La arquitectura del ecosistema debe estar documentada y actualizada.	No cumple	Cumple
La arquitectura de software debe estar documentada y actualizada.	No cumple	Cumple
El modelo de datos debe estar documentado y actualizado.	Cumple	Cumple
El sistema debe contar con una API, la cual debe estar documentada y actualizada para acceder a los datos.	No cumple	Cumple
La autenticación de usuarios debe realizarse usando UPasaporte.	No cumple	Cumple
El código fuente debe estar apropiadamente versionado y respaldado en un repositorio.	Cumple	Cumple
El stack tecnológico utilizado debe estar apropiadamente indicado.	No cumple	Cumple
El stack tecnológico debe estar alineado con las definiciones del DCC.	No cumple	No cumple
El sistema debe contar con las indicaciones pertinentes para que un tercero pueda levantarlo en un ambiente de desarrollo.	Cumple	Cumple

Capítulo 6

Conclusiones y Trabajo a Futuro

El trabajo expuesto en esta memoria permitió mejorar la gobernabilidad del SAR y su capacidad de integración con otros sistemas, esto mediante la intervención directa en el código, y la documentación del sistema basada en la exploración de sus distintos aspectos y componentes. El foco del trabajo siempre estuvo puesto en cumplir con los lineamientos que el DCC empezó a solicitar a cada uno de sus softwares desde marzo del año 2022.

Intervenir un sistema con ese nivel de permiso tiene sus ventajas y desventajas, pues algunos aspectos pueden resultar extensibles de manera natural y generalmente impactan positivamente en el sistema, pero otros aspectos resultan más complejos de extender dada la naturaleza original del proyecto (sus objetivos y alcances) y se debe tener un extremo cuidado para evitar romper funcionalidades que ya operan de manera correcta, pues hay veces que estas resultan ser el núcleo de todo un proyecto.

Debido al punto anterior, documentar un sistema es siempre un buen plan para empezar a extenderlo, pues permite conocer el funcionamiento de cada aspecto del mismo y cómo estos interactúan entre sí. La exploración y el análisis se vuelven una necesidad para tomar las mejores decisiones a la hora de avanzar en un proyecto. Cada decisión debe estar debidamente fundamentada y justificada, pues de no ser así, se corre el riesgo de que el proyecto se desvíe de su objetivo original y se pierda el foco. Los tableros de scoping son muy útiles para dicho sentido, y así tener un panorama general de lo que se quiere lograr y cómo se quiere lograr.

El memorista a cargo de este trabajo es otro más de una lista de personas o equipos que han tomado el SAR, cada uno le agrega un sello personal a la hora de intervenirlo. Esto no es malo por sí mismo, pero es necesario llegar a un punto donde las ideas deben converger, y si esto no es manejado a tiempo, el proyecto puede escaparse de las manos. El SAR no es un sistema complejo debido a los datos que maneja, pero sí tiene un backend enrevesado que se mezcla con un frontend más enredado aún. Esto hace que se necesite un conocimiento más profundo de sus componentes para poder intervenirlo de manera efectiva. Y esto puede tomar más tiempo del que se necesitó para alinearlo con las definiciones del DCC.

Mediante este documento se pretende dejar bien encarrilado al SAR. La documentación que aquí se expone, es un buen punto de partida para que cualquier persona que quiera intervenir el sistema, pueda hacerlo de manera efectiva y sin sentir esa necesidad de buscar dentro del código fuente, ahorrando tiempo y recursos.

La documentación del stack tecnológico (más el modelo de datos, alcances, arquitecturas, etc.), la actualización de casi todas las versiones o frameworks del sistema, la implementación de mecanismos oficiales de autenticación y la correcta estructuración y versionamiento del código fuente, permiten concluir que el SAR es ahora un sistema gobernable y con capacidad de integración. Si bien el de cumplimiento de los ítems no fue el 100 % esperado debido al framework del frontend no actualizado (queda como trabajo a futuro), el desarrollo de la extensión de los gráficos estadísticos permitió mostrar experimentalmente que la gobernabilidad del SAR es mucho más fácil de manejar. El objetivo de la memoria fue cumplido.

En cuanto al trabajo futuro, queda mucho por hacer aún. En el tintero quedaron varias ideas que no formaban parte del alcance de esta memoria, pero que pueden ser implementadas con la ayuda de lo que este trabajo permite, algunas de ellas tienen relación con las restricciones que se encuentran en el tablero de scoping. Dentro de los trabajos a futuro se encuentra:

- Paso a producción limpio, es decir, que se sigan cumpliendo los requisitos de gobernabilidad e integración, una vez que este nuevo sistema se ponga en marcha.
- Mayores filtros para las API de acceso a datos. Además, se puede implementar un CRUD para los datos del modelo de datos, esto fungirá como base para una API más robusta y ayudará a la futura refactorización del backend.
- Refactorizar el backend para que sea más escalable y fácil de mantener. Actualmente no hay mucha abstracción en el código, y los endpoints son demasiado específicos para el sistema. Esto hace que sea difícil de extender y mantener.
- Rediseño de mockup y futura e implementación para el frontend para el SAR. Esto se debe principalmente a que la aplicación no permite visualizar su información correctamente en dispositivos móviles, debido a la antigüedad de las librerías.
- Abstraer los recursos para que no solo se puedan pedir espacios físicos, si no también recursos virtuales, como por ejemplo, un servidor virtual para alojar una aplicación web, o recursos físicos como proyectores, notebooks, etc.
- Abstraer el sistema por completo para que pueda ser utilizado por cualquier departamento, o cualquier universidad, sin necesidad de modificar el código fuente.

Bibliografía

[1] "Sistema de Asignación de Recursos - SAR.", https://sar.dcc.uchile.cl/.

[2] Madrid, F., "Integración y Evolución de Sistemas de Información del DCC," 2017, https://users.dcc.uchile.cl/~sochoa/memoria_fmadrid.pdf.

[3] Ochoa, S. y Arriagada, J. P., "Trabajos de Memoria Ofrecidos - Otoño 2022," 2022, https://onedrive.live.com/view.aspx?resid=C30ADE7E1C4A4678!587&ithint=file\%2cdocx&authkey=!AF648Aa5apZTw1Q.

[4] "The C4 model for visualising software architecture.", https://c4model.com/.

[5] Thompson, M., "Discontinued Long Term Support for AngularJS," 2022, https://blog.angular.io/discontinued-long-term-support-for-angularjs-cc066b82e65a.

[6] endoflife.date, "Django," 2023, https://endoflife.date/django. Lista de versiones Django deprecadas [Tabla].

[7] @bower, 2020, https://twitter.com/bower/status/1235634306564984832. ... Bower is deprecated but maintained [Tweet].

[8] endoflife.date, "Python," 2023, https://endoflife.date/python. Lista de versiones Python deprecadas [Tabla].

[9] "Supported Versions," 2023, https://www.djangoproject.com/download/.

[10] PostgreSQL, "Versioning policy," 2023, https://www.postgresql.org/support/versioning/. Releases [Tabla].

[11] Área Desarrollo de Software, "django_sso," 2022, https://github.com/DCC-FCFM-UCHILE/django_sso. contribuyentes: Arriagada, J. y Herrera, L.

Anexos

Anexo A. Requisitos de Gobernabilidad e Integración de Sistemas del DCC

Para asegurar en gran medida la gobernabilidad de las aplicaciones y la integración entre ellas, cada sistema que se ponga en producción en el DCC debe incluir (al menos) lo siguiente:

- Una descripción del proceso que apoya. Esto incluye el objetivo del proceso abordado, sus actores, las macro-tareas, el workflow del mismo, los dolores actuales del proceso (problemas a resolver u oportunidades a aprovechar) y sus eventuales causas. El proceso debe describirse usando la nomenclatura BPMN (https://www.bpmn.org/). Además, la descripción general del proceso debe especificarse a través de un tablero digital del tipo: "exploración del problema y contexto", como el que se muestra a continuación.

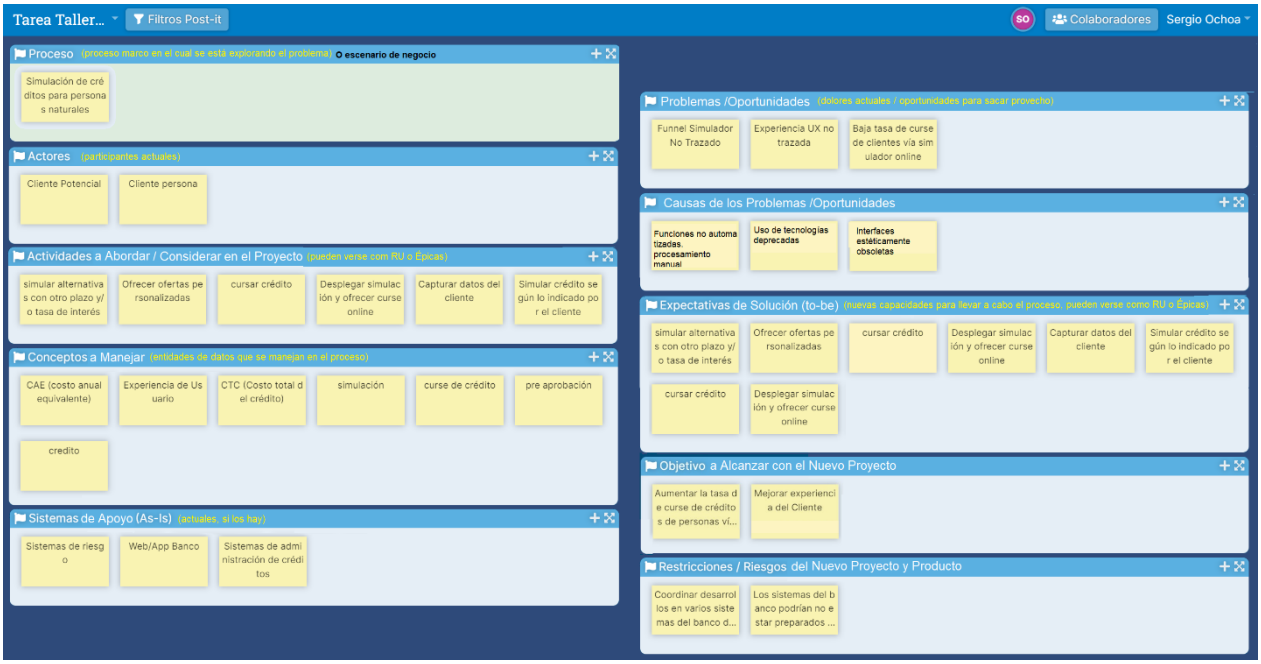

Figura A.1: Ejemplo de un Tablero de Exploración de Problema y Contexto.

- Objetivos y alcance del sistema. Esto debe especificarse utilizando un tablero digital de alcance de productos (tablero de scoping), como el que se muestra a continuación.

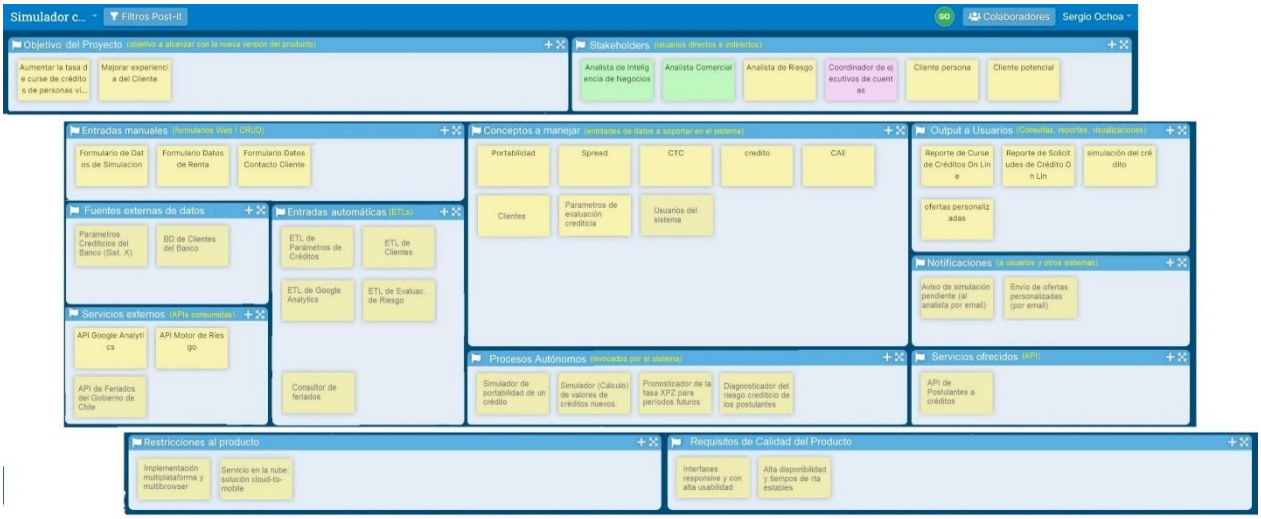

Figura A.2: Ejemplo de un Tablero de Scoping.

- La arquitectura del ecosistema. Esto implica indicar la relación entre el sistema descrito, y otros sistemas con los que éste interactúa. La especificación de la arquitectura del ecosistema debe hacerse utilizando modelos de contexto en C4 (https://c4model.com/). Esta especificación ayuda a entender la integración e interacción entre los distintos sistemas del DCC.

- La arquitectura del software. Esta arquitectura debe estar especificada utilizando modelos de contenedores y componentes de C4 (https://c4model.com/).

- El modelo de datos. En casi todos los casos, el modelo de datos de los sistemas es relacional, por lo tanto, se espera que el modelo cuente con los elementos típicos y mandatorios de este tipo de especificación; es decir, un modelo basado en tablas, con relaciones y cardinalidades entre ellas. Además, para cada tabla debe indicarse la clave primaria, las claves foráneas (si las hay) y los atributos de dicha entidad de datos.

- La API para acceder a los servicios o datos que ofrece el sistema (si la hay). Si los datos del sistema son la fuente de verdad en algún ámbito específico de negocio, entonces éste debe implementar una API a través de la cual otros sistemas puedan acceder a sus datos o servicios. Dicha API debe estar documentada apropiadamente utilizando Swagger (https://swagger.io/).

- Autenticación de Usuarios. Todos los sistemas deben autenticar usuarios utilizando los servicios que provee el DCC. Cada sistema debe implementar su propio esquema de roles y permisos (Autorizaciones).

- El código fuente debe estar apropiadamente versionado. El código fuente de la aplicación deberá estar apropiadamente versionado, estructurado y almacenado en una herramienta de control de versiones; particularmente utilizando Git.

- El stack tecnológico utilizado. Esto debe incluir las tecnologías y sus versiones mínimas requeridas para poder funcionar.

- Indicaciones para su deployment. Esto debe incluir los pasos e indicaciones requeridos para que un tercero pueda poner el software en producción (deployarlo).

Anexo B. Modelo de Datos Completo del Sistema Legado

A continuación se muestra el modelo de datos del sistema legado.

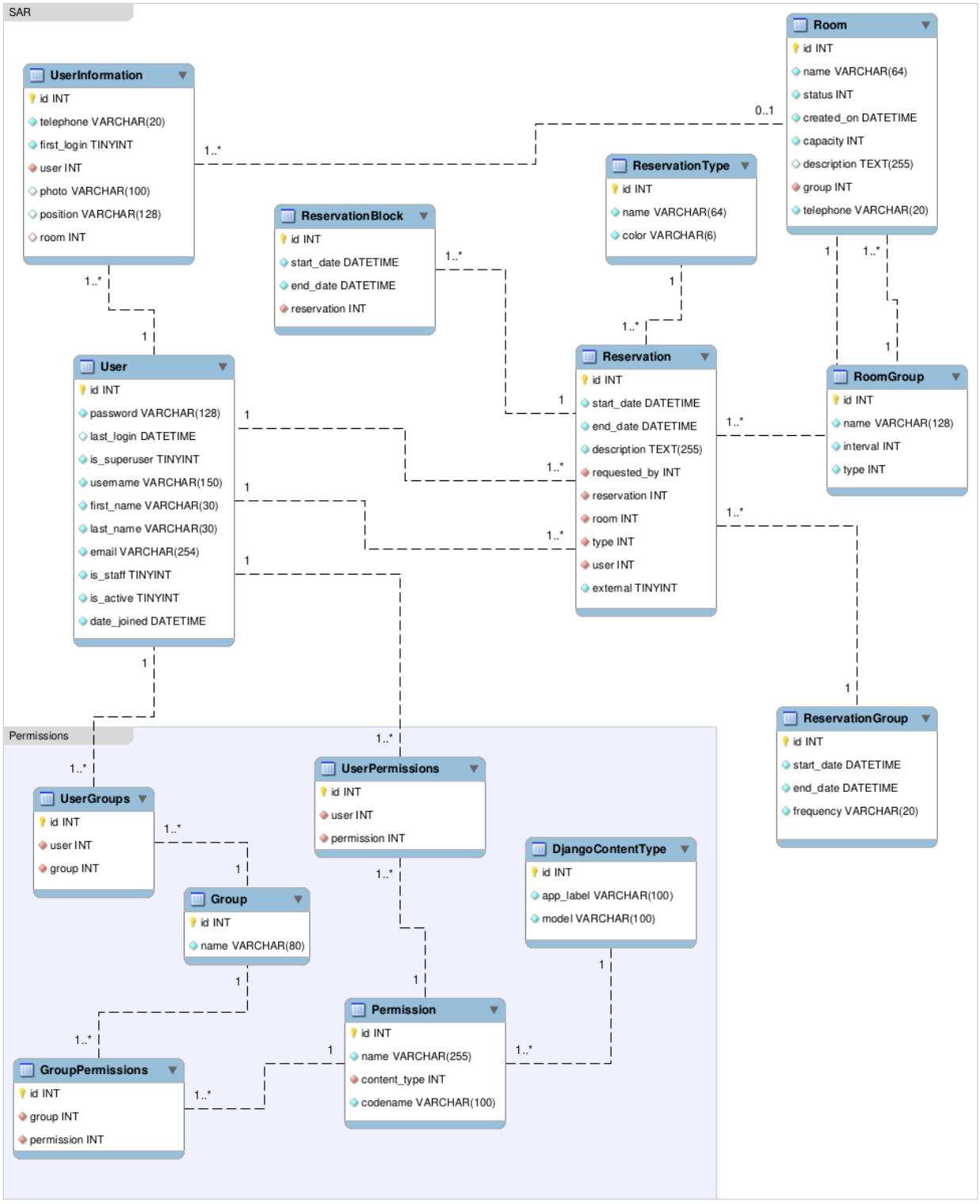

Figura B.1: Modelo de datos del sistema legado.

Anexo C. Modelo de Datos Completo del Sistema Nuevo

A continuación se muestra el modelo de datos del nuevo sistema.

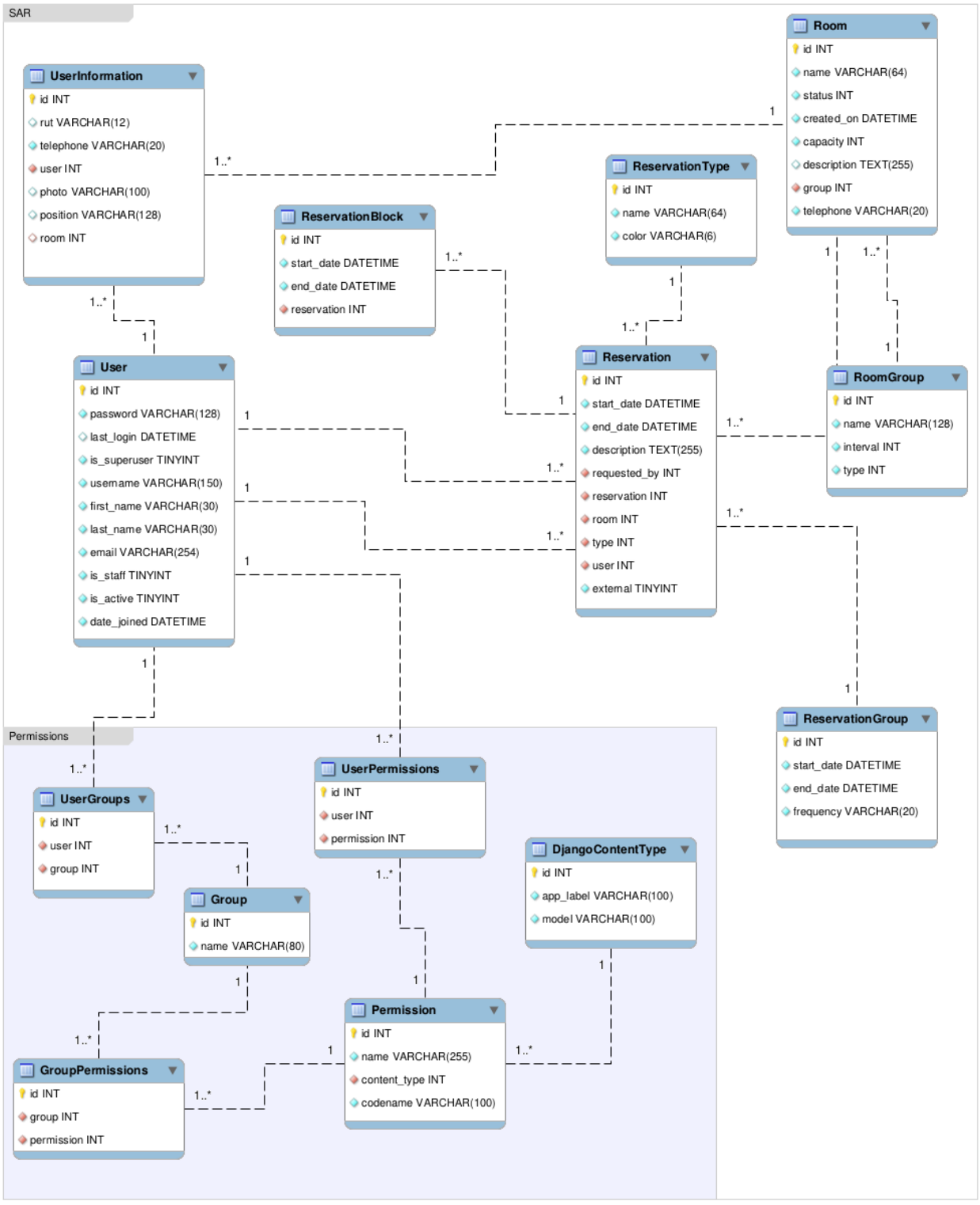

Figura C.1: Modelo de datos del sistema nuevo.

Anexo D. Reserva de salas públicas en agosto de 2022

A continuación se muestra una estadística (cantidad de reservas) de las salas públicas del DCC, para el mes de agosto de 2022. Particularmente se muestra que el Laboratorio Lorenzo tuvo una sola reserva ese mes.

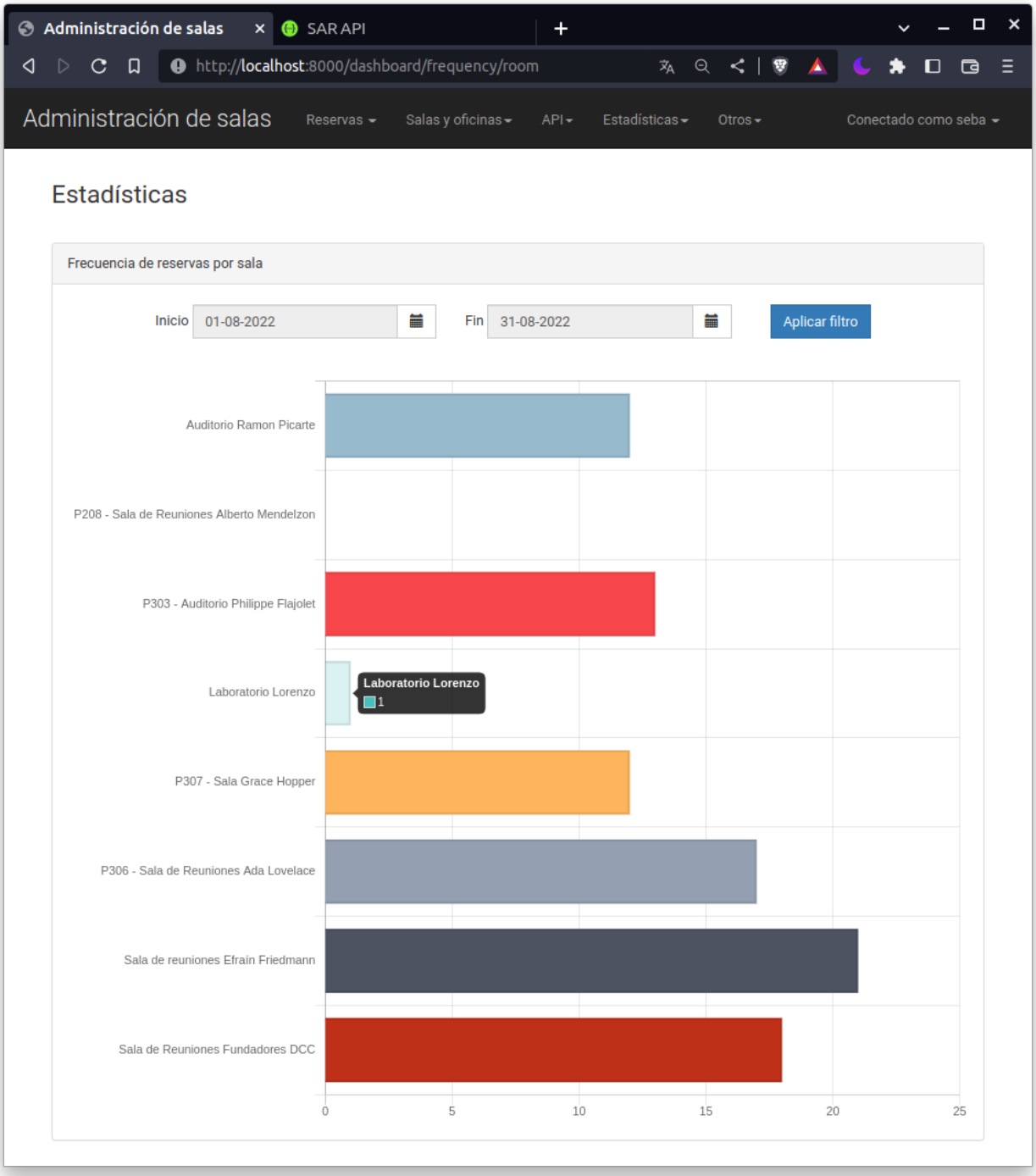

Figura D.1: Estadísticas de uso de espacios compartidos.

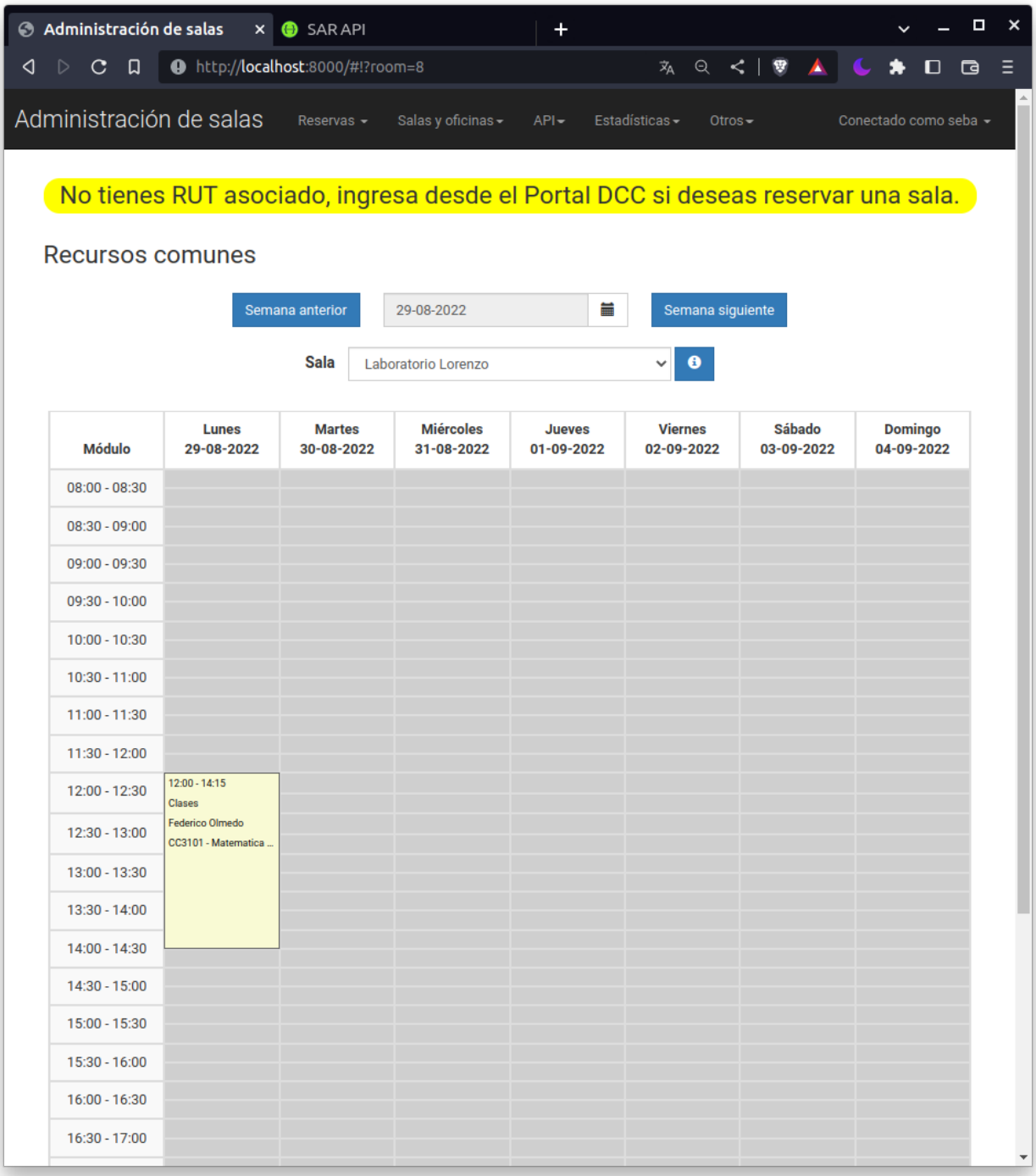

Figura D.2: Aviso que aparece cuando un usuario no tiene información personal asociada.

La figura también muestra una única reserva del Laboratorio Lorenzo en una de las semanas de agosto 2022. No hubo actividad en las semanas anteriores. La reserva de la imagen anterior expone 1 *Reservation* que contiene 9 *ReservationBlocks* (9 bloques de 15 minutos) y con un *ReservationGroup* que indica que no se repite la frecuencia de esta reserva (es única).

Anexo E. Autenticación vía Portal DCC

A continuación se muestra la nueva vista de autenticación de usuarios del SAR. Se pueden apreciar *query parameters* en la URL.

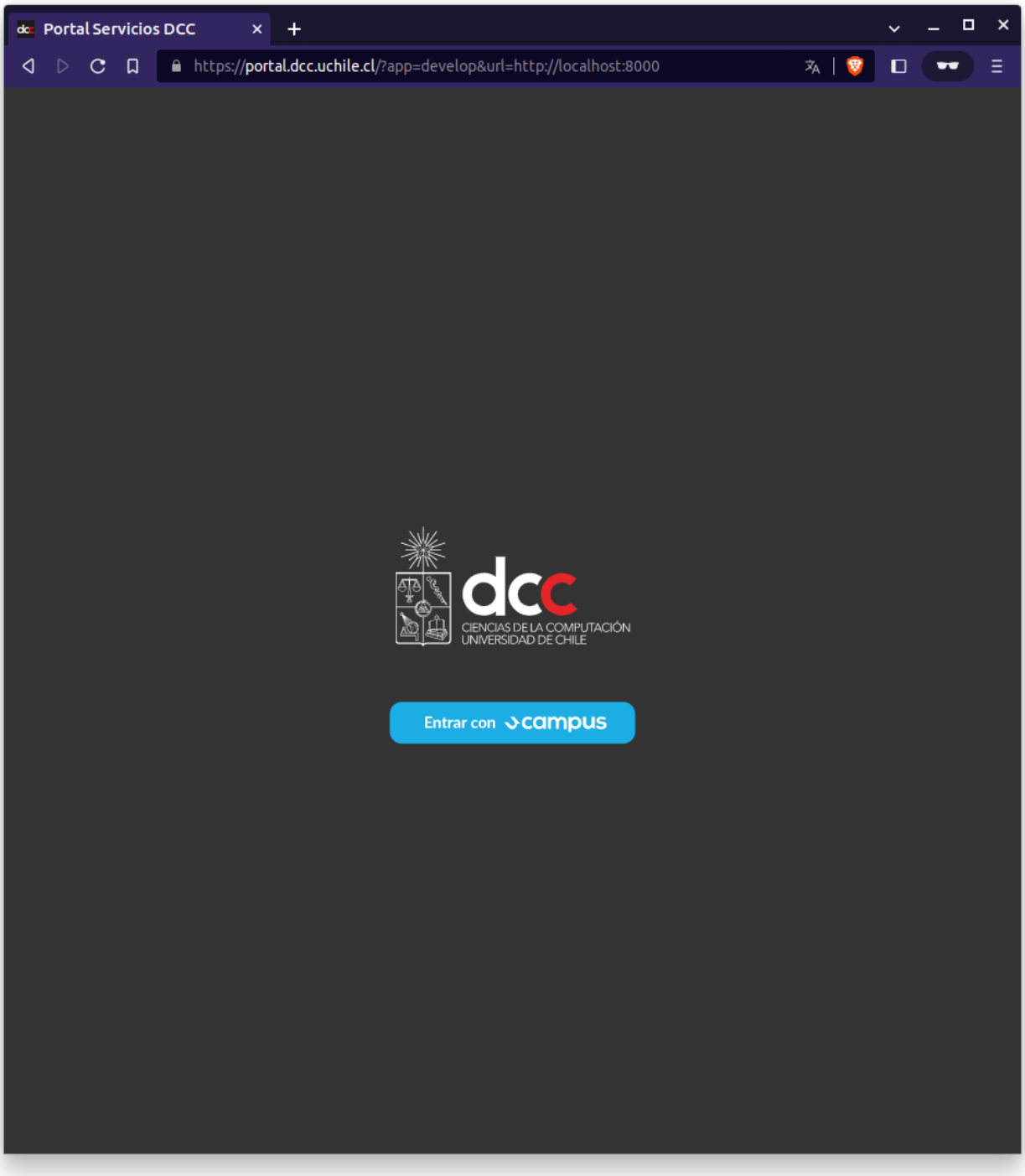

Figura E.1: Portal DCC accedido desde el login del SAR en entorno de desarrollo.

En la Figura V.2 se muestra el login con U-Pasaporte, donde se puede ver también en los *query parameters* de la url que se pretende acceder al servicio del Portal DCC.

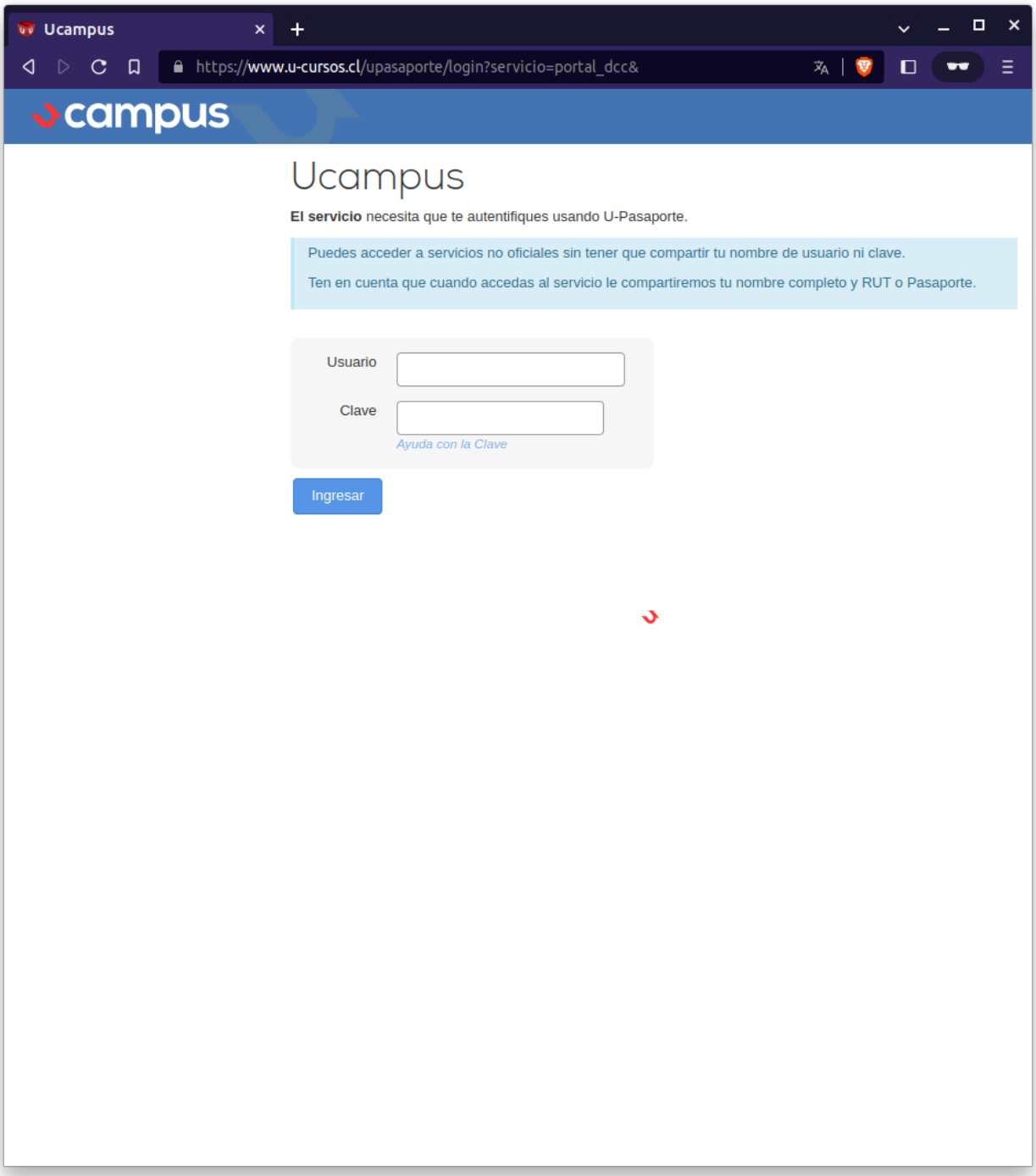

Figura E.2: Login y autenticación con U-Pasaporte.

Una vez autenticado, el usuario es redirigido al Portal DCC el cual realiza verificaciones internas para validar el acceso. Si es exitoso, redirige a la vista principal del SAR (si es que así fue configurado).

Anexo F. Vistas de la documentación de la API v2

A continuación se muestra la documentación de la nueva API.

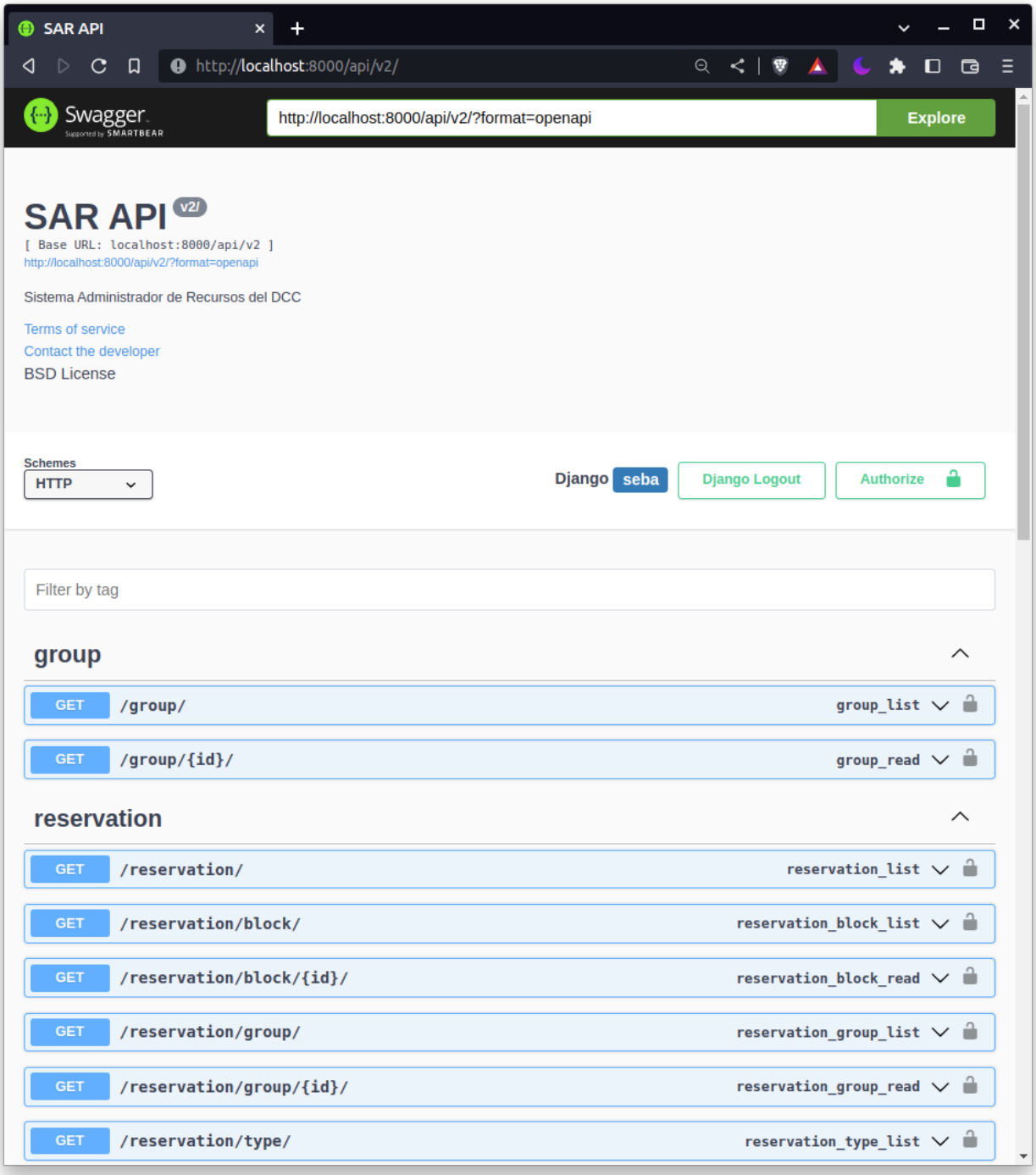

Figura F.1: Vista principal de la documentación de la nueva API.

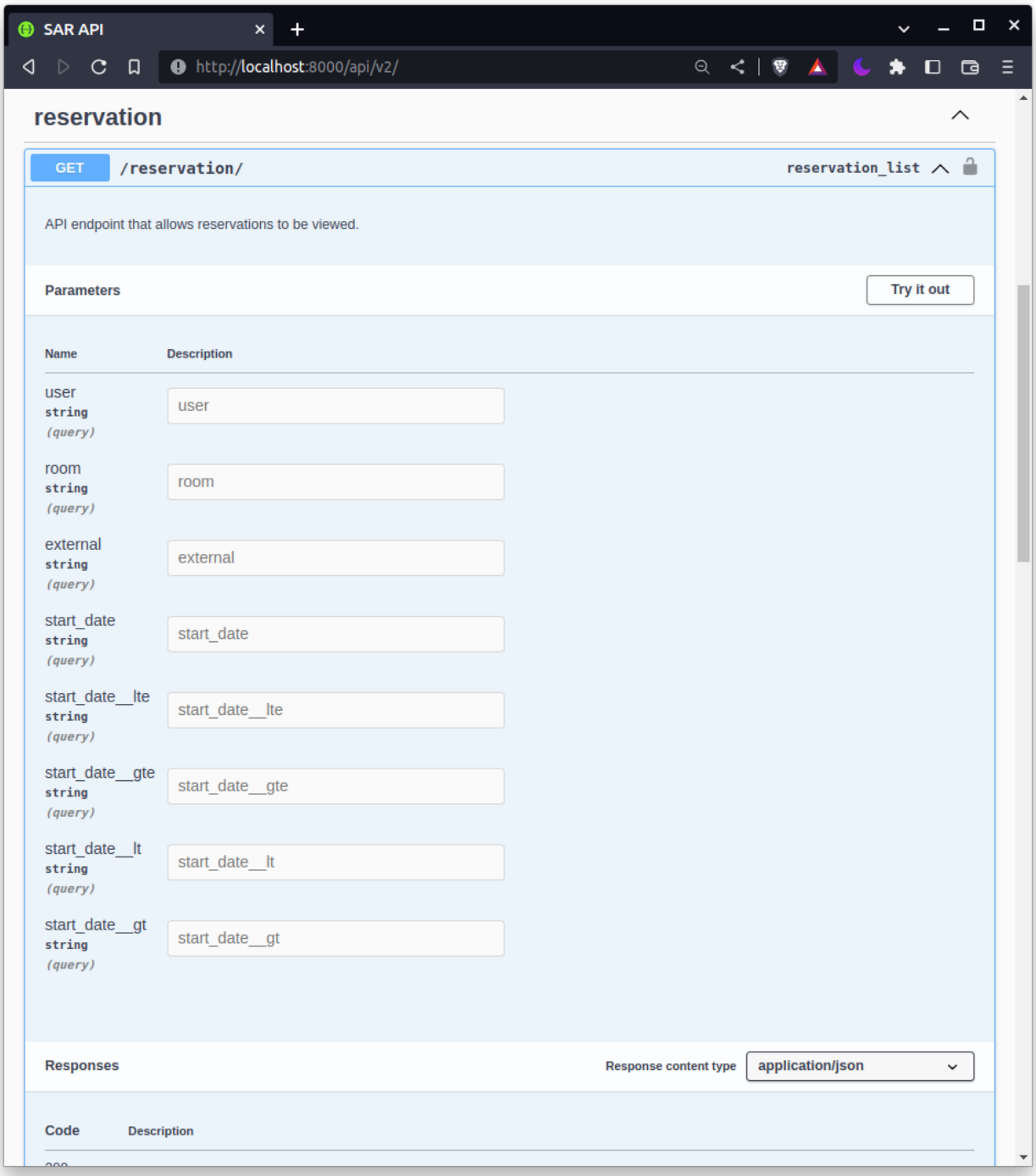

Figura F.2: Detalle de los parámetros que permiten filtrar una búsqueda mediante la API.

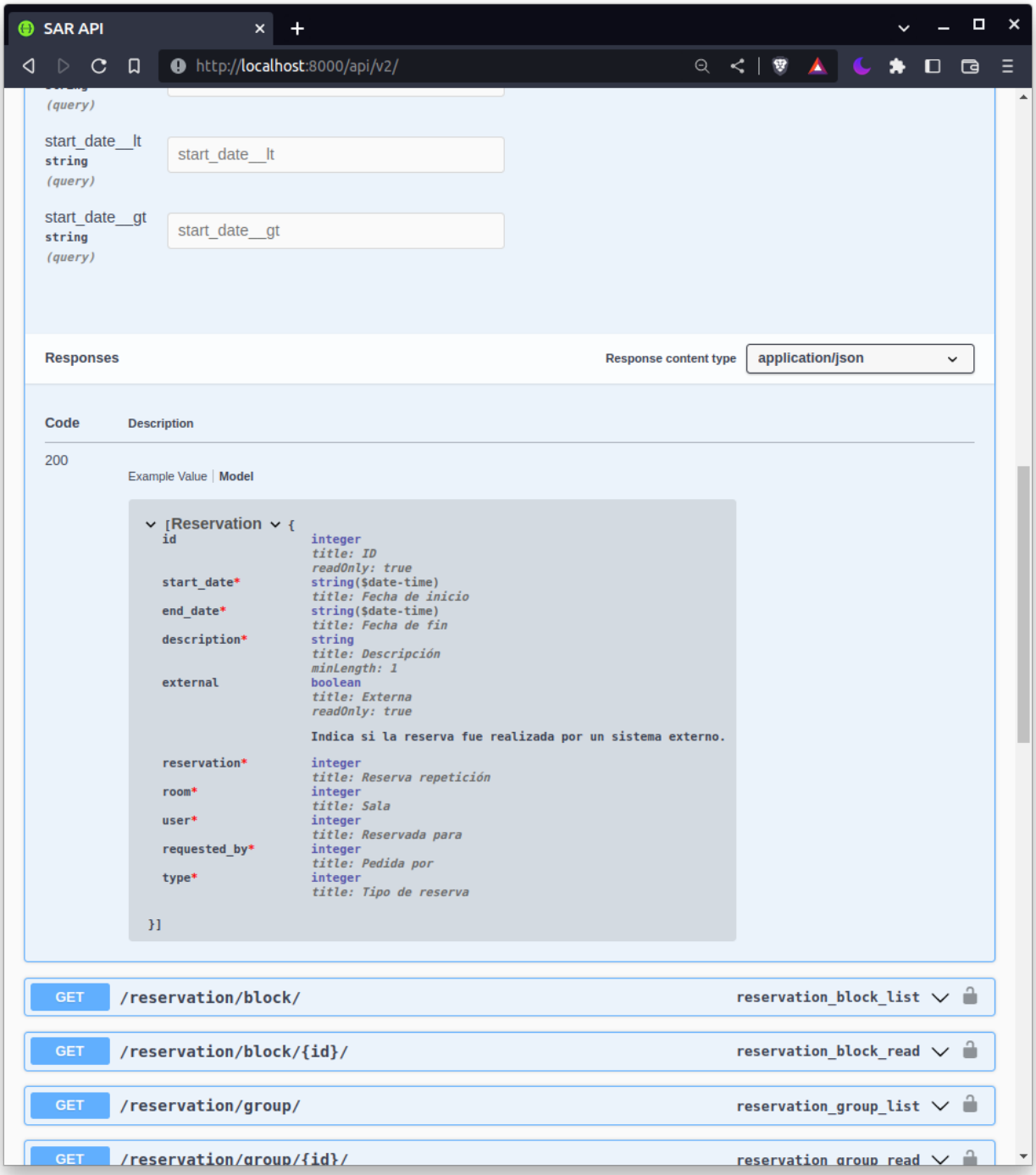

Figura F.3: Esquema de la respuesta esperable por parte de la API para el modelo de reservaciones.

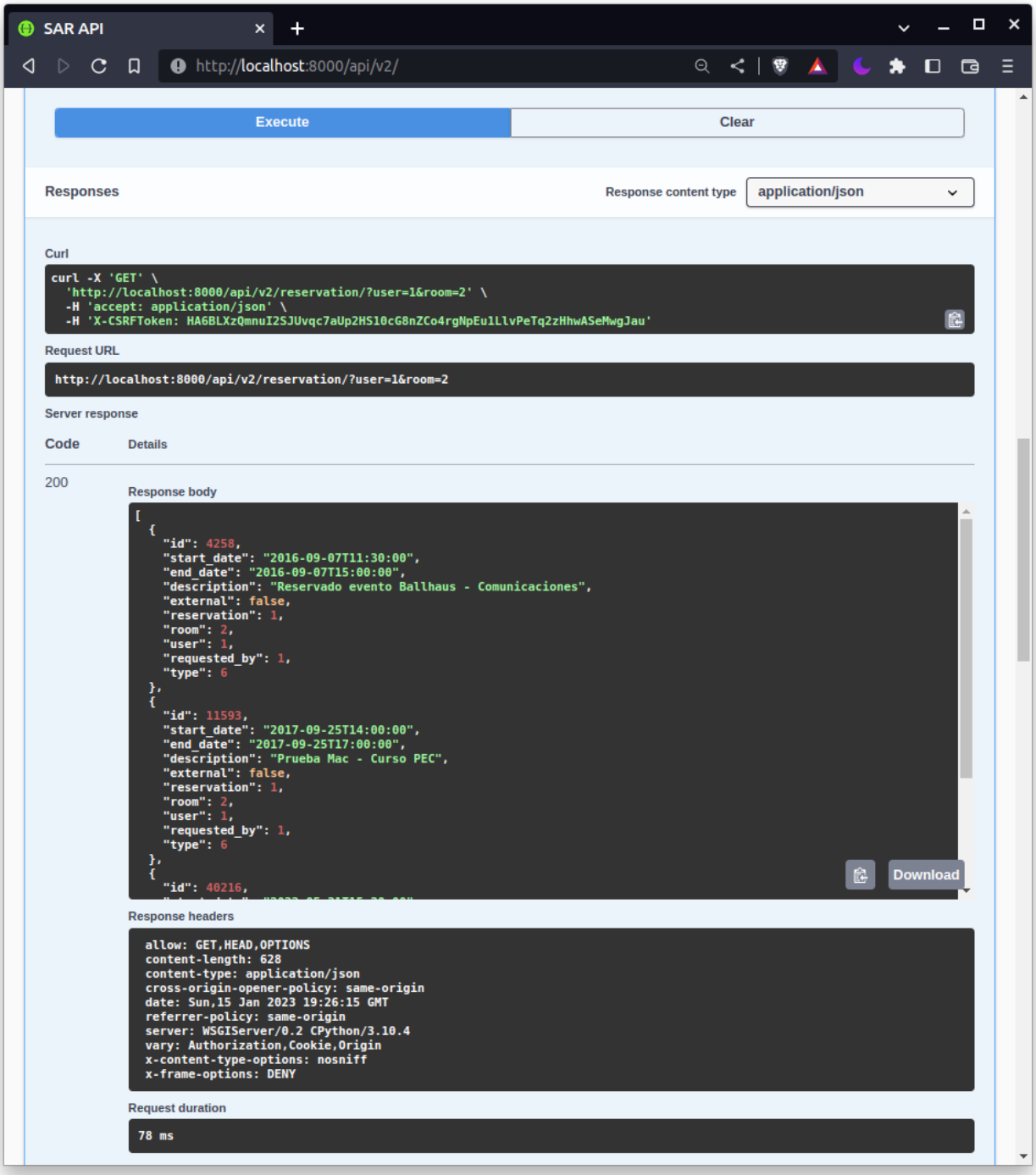

Figura F.4: Ejemplo real de petición y respuesta al endpoint de *Reservation* de la API, aplicando un filtro por el usuario con id 1 y la sala con id 2.