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

UNA PLATAFORMA EN LÍNEA PARA AULA INVERTIDA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

SVEN REISENEGGER MEISSNER

PROFESOR GUÍA:
JÉRÉMY BARBAY

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
JUAN ÁLVAREZ RUBIO

SANTIAGO DE CHILE
2021

Resumen

El “aula invertida” es un método de enseñanza que se basa en el aprendizaje activo por parte de los estudiantes que permite a un instructor tratar de una buena forma con estilos de aprendizaje diferentes durante una clase. En una clase invertida los alumnos estudian las materias en su casa (de forma asíncrona), y durante la clase presencial se dedican a aplicar los conceptos y discutirlos con la ayuda del profesor.

Para facilitar la adaptación de clases invertidas del contexto presencial a un contexto digital y en línea, se plantean dos conceptos. Uno consiste en grupos de discusión inteligentes, basados en las respuestas de la audiencia a una pregunta. El otro es la reutilización de preguntas que se hacen en clases. Un sistema que implemente esto ayudaría a los instructores a disminuir sus tiempos de preparación y mejoraría la calidad de las preguntas.

Para implementar lo anterior se desarrollaron dos plataformas. La primera, Easy-Poll, permite hacer preguntas de alternativas y compartirlas fácilmente con una API a la que otras aplicaciones pueden conectarse para generar, guardar y compartir preguntas de alternativas. La segunda aplicación, y el foco principal de este trabajo, Easy-Flip, permite a profesores y alumnos conectarse por videoconferencia usando Jitsi-Meet, una aplicación web de código abierto para realizar conferencias. Easy-Flip funciona como un *wrapper* que agrega a Jitsi-Meet la capacidad para hacer preguntas a la audiencia (usando la API de Easy-Poll) y además generar grupos de discusión inteligentes con una distribución basada en las respuestas de la audiencia a la pregunta hecha.

Easy-Flip fue probada en tres instancias: una clase de 50 alumnos, una prueba informal, y un conjunto pequeño de alumnos.La primera prueba falló, debido a que el servidor Jitsi usado no aguantó la carga y se perdía el audio de parte del profesor. En la segunda prueba se encontraron un par de errores que fueron arreglados. En la última prueba se confirmó que las características implementadas funcionaban, pero la cantidad de alumnos que asistió fue muy baja como para sacar conclusiones relevantes respecto a la utilidad del software.

Se concluye que se cumple con los objetivos del trabajo con respecto a la creación de grupos inteligentes, pero no con los que refieren a la reutilización de preguntas. Además se considera que es necesario hacer pruebas con más usuarios para evaluar la real usabilidad de la aplicación. Finalmente, se mencionan posibles mejoras para trabajar a futuro.

Agradecimientos

A mi profesor guía, Jérémy Barbay, que siempre estuvo ahí para resolver dudas, y fue comprensivo con todos mis atrasos.

A todos los frens del DCC, que me han acompañado de manera remota durante todo este trabajo y me han dado ánimo y energías cuando los he necesitado. Especialmente a Pablo, que ha sido un excelente amigo durante toda mi estadía en la universidad.

A mi hermana y mis hermanos, Stefi, Alex y Lukas, los quiero mucho y verlos siempre me trae alegría.

A mi mamá y mi papá, también los quiero mucho, siempre han estado aquí conmigo, y siempre me han dado todo lo que necesito para seguir adelante.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto

Se le llama "Aula invertida" a una metodología de enseñanza enfocada en el aprendizaje activo por parte de los estudiantes, que permite al instructor tratar de mejor manera con niveles mixtos de conocimiento o habilidades, dificultades de los alumnos y estilos de aprendizaje diferentes durante la clase. La característica principal de este método consiste en mover la transmisión de información (por ejemplo, una explicación de un nuevo concepto por parte del profesor) desde una instancia síncrona, con un ritmo único (el del profesor), al espacio asíncrono y personalizado de la casa, donde cada alumno aprende a su propio ritmo. Esto da lugar para trasladar las actividades interactivas (donde los alumnos le dan sentido a la materia, como trabajos que normalmente se deberían realizar en la casa) a actividades grupales durante la clase presencial. En una clase invertida, los alumnos estudian, colaboran, o investigan en su casa, de manera asíncrona, mientras que durante la clase presencial aplican juntos los conceptos con la ayuda de un tutor [21].

Comenzando el año 2020, con una pandemia global que restringe las reuniones masivas, instructores ya versados en la metodología de aula invertida la adaptaron a un sistema online. Para esto hicieron uso de variadas plataformas para hacer lo que previamente se hacía de forma presencial: videoconferencias para la comunicación del profesor hacia los alumnos, sistemas de votación en línea para evaluar a los alumnos y mostrar los resultados de forma gráfica, salas de discusión para permitir a los estudiantes discutir sus preguntas entre ellos y botones para llamarlos de vuelta a la sesión principal. Uno de los pioneros de la metodología de aula invertida, Eric Mazur, menciona que esta adaptación apurada hacia un "Aula invertida online" introdujo varias innovaciones que incluso podrían (y deberían) ser usadas en la ausencia de cuarentena [14].

Las clases expositivas presenciales en general sí incluyen interacciones entre los instructores y la audiencia, aunque sólo sea para chequear si están entendiendo la materia expuesta o mantener la atención de la audiencia. Estas interacciones se ven enormemente reducidas cuando la exposición se hace en línea (por ejemplo por videoconferencia). Es de esperar que la reducción de dichas interacciones, combinada con un aumento de perturbaciones externas, afecte negativamente la experiencia de aprendizaje de los estudiantes. Como resultado apa-

rente de dicho razonamiento, un gran número de instructores, de diversas culturas e idiomas, están mostrando interés en adoptar nuevas metodologías. Eric Mazur menciona en su cuenta de Twitter, que un total de 3000 personas participaron en los 15 "webinars" (incluyendo uno en español) que hizo al respecto durante marzo, abril y mayo [12] de 2020.

1.2. Problema

Si bien la metodología de "Aula invertida en línea" tiene varias ventajas, tanto frente a clases expositivas tradicionales (presenciales o en línea), como frente al aula invertida presencial, sus implementaciones actuales presentan diversos inconvenientes y posibilidades de mejora:

- **Grupos de discusión selectivos:** en un aula invertida, los grupos de discusión (de 2 a 5 personas) usualmente se generan en base a proximidad física (en una sala de clases), o de manera aleatoria (en línea), lo que eventualmente resulta en que todos los miembros del grupo ya están de acuerdo en la solución y no tienen mucho que discutir [11].
- **Dificultad para crear "buenas" preguntas conceptuales:** Las preguntas que se hacen a los alumnos deberían ser lo suficientemente fáciles para que algunos alumnos las puedan responder, pero al mismo tiempo deben tener una dificultad que promueva la discusión de conceptos importantes del curso [1]. La dificultad de diseñar preguntas que cumplan tales características es similar a la de diseñar preguntas para un examen, y es una habilidad que los instructores desarrollan sólo con la práctica. Esta dificultad puede ser uno de los principales bloqueos para una adopción rápida y masiva de la metodología de aula invertida.

1.3. Solución

En lo que viene se describirán brevemente dos conceptos ideados para resolver el problema (1.3.1) y luego se presentará una formalización de la solución a implementar (1.3.2).

1.3.1. Idea general

Para resolver los problemas recién descritos, es que se introducen dos conceptos. El primero consiste en la generación de grupos de discusión de forma inteligente y el segundo en la reutilización de preguntas.

En teoría, un sistema que utiliza las respuestas de los alumnos para maximizar la diversidad de opiniones en cada grupo asegura un mejor contexto de discusión. Incluso el sistema podría decidir el tamaño de los grupos basado en la cantidad de alumnos que respondieron de forma correcta. Es por esto que se define el concepto de **grupos de discusión inteligentes**, que consisten en separar a los alumnos en subgrupos, pero usando algún algoritmo para generar la distribución de una forma que haga sentido para para la tarea asignada. Por ejemplo, si se quiere que haya diferencia de opiniones para una discusión más fructífera, se utiliza

[1] Eric Mazur menciona [13] que idealmente, 35 a 70 por ciento de los alumnos deberían obtener la respuesta correcta al primer intento, y un 90 a 100 porciento deberían obtenerla luego de conversarlo con sus compañeros.

un algoritmo que deje juntas a las personas que opinaron distinto en una pregunta. O si se quieren armar grupos de trabajo con gente que piensa parecido, se deja juntas a las personas que opinaron igual.

También se cree que si existe alguna forma en que los instructores puedan buscar y compartir preguntas para hacer durante sus clases, se facilita la interacción dentro de éstas, ahorrando tiempo y esfuerzo al momento de buscar dichas preguntas.

1.3.2. Solución Formal

Aplicando los conceptos recién descritos, se desarrolló un sistema para facilitar la adopción e implementación de la metodología de aula invertida en línea, que se enfoca particularmente en:

- Integrar videoconferencia (usando Jitsi [8]) con un sistema de votación (replicando una selección de características contenidas en servicios como PollEverywhere [17]) para así mejorar el proceso de separar los alumnos en grupos de discusión.
- Promover la reutilización y difusión de preguntas y de sus respuestas entre instructores, con el potencial de también compartir estadísticas de las respuestas por parte de los alumnos como un indicador parcial de la calidad de aquellas preguntas.

Tomando en cuenta lo anterior, se desarrollaron dos aplicaciones. Easy Poll, que es un sistema sencillo para crear y compartir preguntas de alternativas, y que pretende implementar a futuro un sistema para reutilizar preguntas. Y Easy Flip, que implementa grupos de discusión inteligentes usando Jitsi Meet como mecanismo de videoconferencias.

Si bien aquí se definen dos aspectos importantes de la solución, sólo el primero fue implementado y validado. Para el segundo, por limitaciones de tiempo, sólo se implementó una base, que es necesario expandir a futuro.

1.4. Objetivos

Basado en la solución descrita, se identifica la necesidad de crear un software que integre aulas virtuales y un sistema de votaciones. A continuación se describirán los objetivos planteados para el desarrollo de esta memoria.

1.4.1. Objetivo General

El objetivo del trabajo consiste en desarrollar y validar una plataforma de enseñanza remota para facilitar la adopción y el uso de la metodología de aula invertida en línea. La plataforma debe permitir que alumnos y profesores se comuniquen mediante videoconferencia y proporcionar al profesor herramientas esenciales para gestionar una clase invertida efectiva.

1.4.2. Objetivos Específicos

1. Permitir que los alumnos y el profesor se puedan comunicar por videoconferencia, audio conferencia y mensajería de texto instantánea a través de una plataforma web que use

Jitsi-Meet.

2. Permitir al profesor hacer preguntas de alternativas a los alumnos, y que los alumnos puedan entregar sus respuestas a través de la plataforma. El profesor debe poder ver las estadísticas de las respuestas y compartirlas.
3. Permitir al profesor dividir al curso en grupos de discusión, basados en sus respuestas. Los alumnos de dichos grupos deben ser capaces de ver la pregunta, ver su respuesta a la pregunta, interactuar (por video/audio-conferencia o chat de texto) y poder pedir ayuda al profesor.
4. Facilitar la colaboración entre profesores, permitiendo buscar y reusar preguntas sobre temas similares a través de la plataforma.
5. Validar el software con usuarios (profesores y alumnos), vía formulario que deberán contestar antes y después de probar el software, para así identificar las fortalezas y debilidades de la solución implementada. De esta manera se podrá guiar el desarrollo futuro de la aplicación.

En los capítulos que siguen, se presentarán en detalle las aplicaciones desarrolladas (capítulos 3 y 4), luego de eso se describirán los procesos que se hicieron para validar dichas aplicaciones (capítulo 5. Luego, en la conclusión (capítulo 6) se presentará una evaluación de los objetivos recién listados. Al final se entregará una lista con mejoras a futuro para la solución.

Capítulo 2

Marco Teórico

En este capítulo se describen las tecnologías utilizadas para desarrollar este trabajo (2.1), se introduce brevemente el concepto de aula invertida (2.2) y también se presenta un estudio de soluciones ya existentes para el problema identificado anteriormente (2.3).

2.1. Tecnologías

La solución implementada utiliza múltiples tecnologías de desarrollo web. En esta sección se da una breve descripción de ellas y se explica por qué son relevantes para este trabajo.

2.1.1. JSON

JSON (o *JavaScript Object Notation*) es un formato de intercambio de datos liviano. Es de fácil lectura y escritura para el ser humano y también es fácil de analizar y generar para las máquinas. [3]

JSON está compuesto por dos estructuras:

- Una colección de pares nombre/valor. En JavaScript a esto se le llama objeto, pero en otros lenguajes puede ser conocido como diccionario, estructura, tabla de hash, etc.
- Un *array*, que consiste en una lista ordenada de valores.

En JSON un valor puede ser un string, que se escribe entre comillas, un número, true, false, null, un objeto o un array. Esto implica que se pueden tener arrays u objetos anidados.

La mayoría de la información intercambiada entre clientes y servidores en este trabajo se encuentra en formato JSON.

2.1.2. HTTP

HTTP, llamado así por sus siglas en inglés *Hypertext Transfer Protocol*, es un protocolo de capa de aplicación para sistemas de información distribuidos, colaborativos y de hypermedia [6]. HTTP es una de las bases de la comunicación de datos en internet.

HTTP funciona como un protocolo de petición-respuesta en los modelos de cliente servidor. Por ejemplo, un navegador puede ser el cliente y una aplicación corriendo en una máquina que aloja un sitio web podría ser el servidor. El cliente manda una petición HTTP al servidor y luego el servidor procesa dicha petición y entrega una respuesta, que puede ser un archivo HTML u otro contenido.

En este trabajo, Easy Poll hace uso de peticiones HTTP para intercambiar archivos JSON entre el cliente y el servidor. Además de eso, como el trabajo consiste de aplicaciones web, los clientes web reciben los datos de la aplicación a través de HTTP.

2.1.3. Node.js

Node es un entorno de ejecución de back-end multiplataforma y de código abierto, que permite ejecutar código JavaScript fuera de un navegador. Node.js permite a los desarrolladores utilizar JavaScript en un servidor, para generar páginas web dinámicas. El servidor del trabajo desarrollado corre sobre Node.js.

2.1.4. Docker

Docker es una plataforma para desarrollar y correr aplicaciones, que ayuda a los desarrolladores a separar las aplicaciones de la infraestructura, de manera que pasar de un entorno de desarrollo a uno de producción se vuelve mucho más fácil. Para este trabajo se utilizaron imágenes y *containers* de docker, además de una herramienta llamada docker-compose.

Una aplicación que usa docker, es empaquetada y corre en un ambiente aislado llamado *container*. El aislamiento y la seguridad que este servicio provee, permite correr varias aplicaciones de manera simultánea en una misma máquina. Los containers son livianos y tienen todo lo necesario para correr la aplicación, de manera que no es necesario depender de lo que está actualmente instalado en la máquina que hospeda el servicio. [2]

Una imagen de docker es una plantilla de solo lectura que contiene las instrucciones para crear un *container*. Frecuentemente las imágenes están basadas en otras imágenes, por ejemplo, se puede crear una imagen basada en una imagen que contiene ubuntu, y que instala un servidor web y una aplicación sobre ella. En este trabajo se usó una imagen con una distribución de Linux Alpine (que es una distribución liviana) para correr y configurar la solución desarrollada.

La figura 2.1 muestra la arquitectura de Docker, que es del tipo cliente-servidor y ambos pueden correr en la misma máquina. El cliente envía comandos al proceso *Docker daemon* que se encuentra en el servidor, y este hace el trabajo de construir y levantar los contenedores en base a las imágenes que se le entregan.

En este trabajo se usó Docker y docker-compose para empaquetar todas las partes de la aplicación. De esta forma, si a futuro se quiere distribuir la aplicación, o instalar en otra máquina, el trabajo necesario es mínimo.

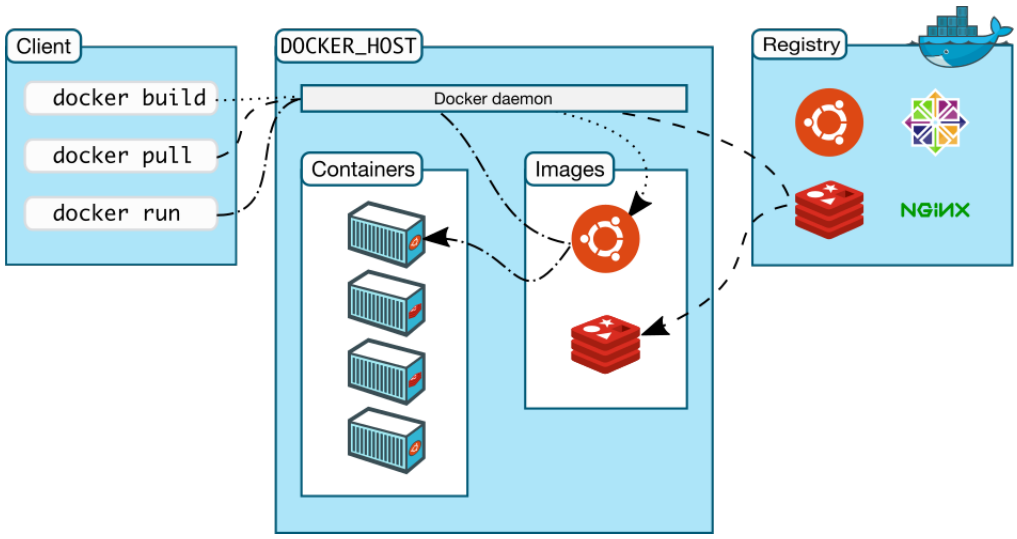


Figura 2.1: Arquitectura de Docker

2.1.5. Socket.io

Socket.io [19] es una biblioteca que permite comunicación bidireccional basada en eventos entre clientes web y un servidor. Para su funcionamiento utiliza principalmente websockets [5], pero si estos fallan se utiliza *HTTP long polling* [10].

Entre las características de socket.io que lo hacen una mejor opción que usar simplemente websockets, se encuentra su fiabilidad (dado que puede usar *HTTP long polling* si los websockets fallan), su capacidad de reconectarse de forma automática, y su facultad para enviar mensajes a todos los usuarios (o un subconjunto de estos) al mismo tiempo.

Para usar socket.io, se debe instalar tanto en el cliente como en el servidor. Tanto en el cliente como en el servidor se utiliza una instancia llamada socket, que se encarga de emitir y recibir eventos, además de guardar información respecto a la conexión. Un socket tiene un identificador propio y además se le pueden agregar atributos adicionales dependiendo de los requerimientos de la aplicación.

La instancia del servidor tiene la facultad de entrar o salir de "salas", que son una forma de agrupar sockets para que el servidor pueda enviar un mensaje a todos los miembros de una sala simultáneamente (ver figura 2.2). Esta característica fue particularmente importante para el desarrollo de este trabajo.

2.1.6. IFrame

Se define IFrame como un contexto de navegación anidado. Un Iframe es un elemento HTML que permite mostrar contenido de una página web dentro de otra. En esta aplicación se utiliza para mostrar el contenido de una reunión de Jitsi Meet a los usuarios de Easy Flip. [22]

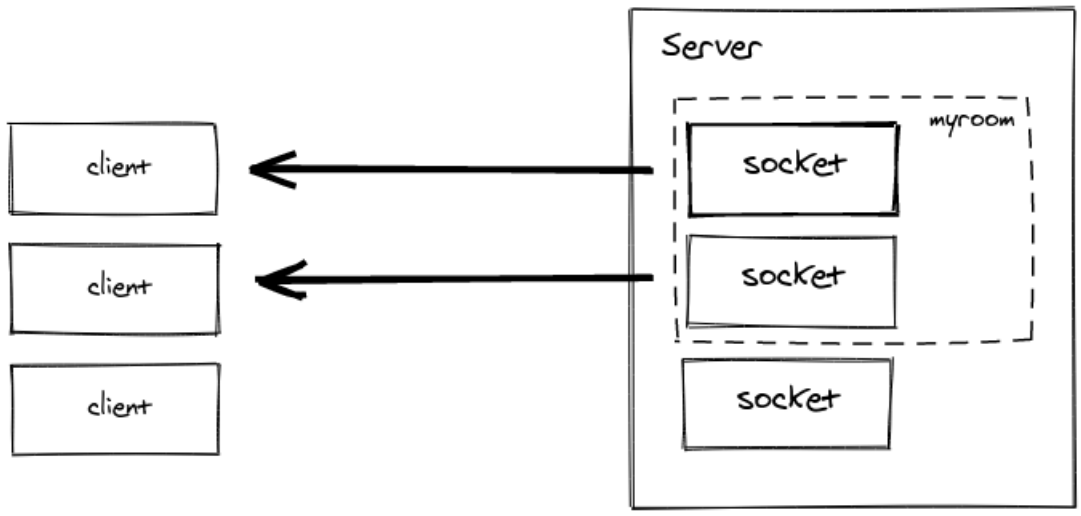


Figura 2.2: Salas de socket.io

2.1.7. React.js

React.js es un framework front-end de JavaScript que sirve para construir interfaces de usuario interactivas y modulares. Las entidades de código que componen una aplicación en React se llaman componentes. Las componentes pueden ser anidadas o compuestas entre si para generar componentes más complejas, además cada componente puede almacenar estado. [4]

Para programar las componentes se utilza JSX (o JavaScript XML), que es una extensión de JavaScript. JSX es muy similar a HTML y otorga una forma de estructurar las componentes que es amigable para los desarrolladores.

La solución implementada utiliza React.js para crear interfaces interactivas, que reaccionen a las acciones del usuario, sin necesidad de refrescar la página luego de cada acción.

Express.js

Express.js es un framework de desarrollo web back-end para Node.js. Express es minimal, flexible y tiene un conjunto de características robusto, que lo hacen ideal para construir aplicaciones web con facilidad. En el desarrollo de la aplicación se usa Express para escuchar y procesar las solicitudes HTTP enviadas por los clientes web al servidor. [16]

MongoDB

MongoDB es un programa para bases de datos escalable y flexible que guarda los datos en documentos similares a archivos JSON, lo que hace que los campos puedan variar entre distintos documentos y la estructura de datos puede ser cambiada. [15]

Se optó por usar MongoDB porque es fácil de configurar para el uso con aplicaciones JavaScript, dado que los datos se guardan en un formato similar a JSON. En la aplicación se usa una biblioteca llamada Mongoose para definir la estructura de los datos y para la

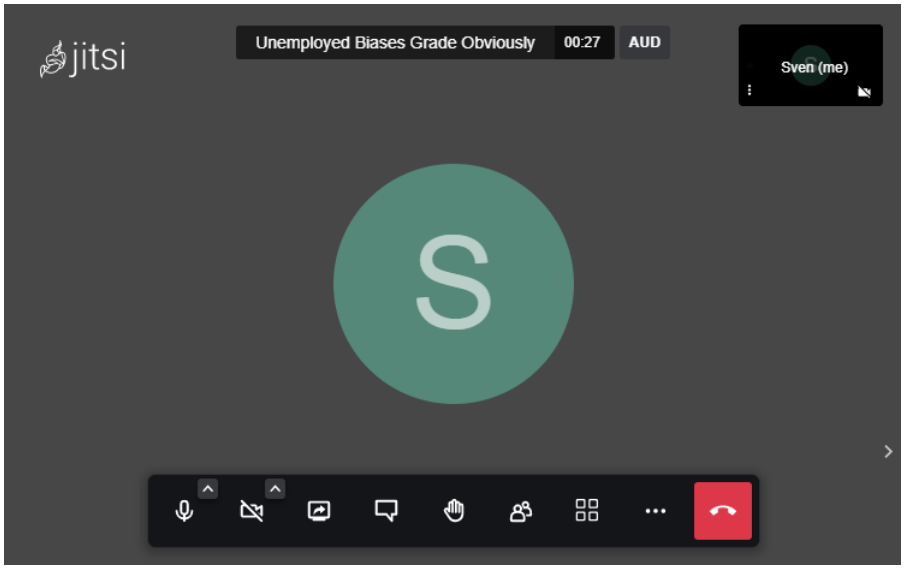


Figura 2.3: Interfaz de Jitsi Meet

integración con MongoDB. Mongoose provee un mecanismo llamado *schema*, que permite al desarrollador modelar los datos, validar los campos y construir consultas.

2.1.8. Jitsi Meet

Jitsi es una colección de aplicaciones de código abierto para videoconferencias y mensajería instantánea. Para este trabajo se utilizó su aplicación web de videoconferencias: Jitsi Meet. Esta aplicación es de uso gratuito y código abierto. Su distribución oficial (`meet.jit.si`) ofrece reuniones de hasta 75 personas simultáneas. [9]

Algunas de sus características importantes son:

1. Chat de texto.
2. Compartir pantalla.
3. Levantar/bajar la mano para pedir hablar.
4. Mostrar videos de Youtube a la audiencia.
5. Se puede integrar a otras apps o sitios web usando un IFrame.

La última característica es especialmente importante, dado que se usó en este trabajo para implementar la comunicación entre profesores y alumnos. En la figura 2.3 se puede observar la interfaz de usuario de Jitsi Meet, que incluye botones para desplegar el chat, activar o desactivar cámara y micrófono, compartir pantalla y levantar la mano, entre otros.

2.2. Aula invertida

El "Aula invertida" es un método de enseñanza en que el aprendizaje activo, o aplicación de los conceptos se hace durante la clase, con un tutor presente para resolver dudas y dirigir, mientras que la adquisición de nuevos conceptos (parte pasiva del aprendizaje) se hace fuera de la clase y de manera personal. [21]

El método de aula invertida cambia el modelo de enseñanza tradicional, donde el instructor está al centro de una clase a uno en que los alumnos están en el foco. Esto otorga oportunidades de aprendizaje significativos a los alumnos y les permite explorar los temas enseñados de una forma más profunda. La forma de entregar contenidos a los alumnos en una clase invertida puede variar, frecuentemente se entregan videos, textos o tareas de investigación, que posteriormente son discutidos en la clase.

Durante la clase misma, la actividad principal deja de ser una monólogo por parte del profesor, sino que se realizan diversas actividades. Estas actividades pueden ser diversas: un experimento práctico, resolución de problemas, tareas de escritura, discusión entre los pares, presentaciones, desarrollo de proyectos, etc. En este trabajo se busca facilitar y mejorar la eficiencia de la discusión de problemas o preguntas conceptuales entre pares y ayudar a profesores a crear mejores preguntas conceptuales.

2.3. Soluciones existentes

Actualmente existe un sinnúmero de aplicaciones que permiten hacer videoconferencias y realizar clases en línea. A continuación se detallarán algunas de las mas populares y también se mencionarán sistemas que se usan junto a estas para hacer preguntas a la audiencia.

2.3.1. Software para videoconferencias

Zoom [23]: Zoom es una de las plataformas mas utilizadas actualmente y viene con features para hacer votaciones y para separar a los asistentes en grupos de discusión. Si bien Zoom tiene una versión gratuita, ésta solo permite conferencias de cuarenta minutos y varias de sus características solo están incluidas en la versión de pago.

Adobe Connect [1]: Esta plataforma es de pago, y trae una gran cantidad de características. Al igual que Zoom, permite hacer breakout rooms y tiene un sistema de encuestas.

Google Meet [7]: Es una plataforma gratuita, pero está pensada para reuniones y tiene una cantidad limitada de usuarios conectados a la vez, y no ofrece sistema para votaciones.

Jitsi Meet [9, 8]: Es un servicio web para videoconferencias, que a pesar de que no tiene tantas features como sus competidores, es gratuito y de código abierto. Además permite a cualquier desarrollador incluir un cuadro para videoconferencias dentro de su página web.

2.3.2. Software para hacer votaciones

En la metodología de aula invertida, los sistemas de videoconferencia se suelen utilizar acompañados de plataformas web para realizar votaciones, dado que los sistemas que traen aún son muy básicos para las necesidades de algunos instructores. A continuación se describen las plataformas de votación mas populares.

PollEverywhere: [17] Es un sitio que permite hacer encuestas de distintos tipos, a las que los participantes ingresan con un simple enlace. El presentador puede luego mostrar las estadísticas con visualizaciones variadas.

SurveyMonkey: [20] Sitio para hacer encuestas, bastante similar a PollEverywhere, pero más orientado a grupos de trabajo y empresas que a audiencias.

Un problema común de estas plataformas es que no están enfocadas en educación (excepto Adobe, que tiene una versión para salas de clase virtuales), y además todas son software propietario, lo que no permite a las instituciones adaptarlas a sus propias necesidades.

2.3.3. Características de interés

Para poder comparar las distintas plataformas, se hará una breve descripción de las características relevantes en el contexto de una clase en línea.

- **Votaciones:** El software permite a quién dirige la clase (o conferencia) plantear una pregunta con múltiples alternativas a la audiencia. Luego de plantear la pregunta, el número de respuestas para cada alternativa debe ser visible para el anfitrión. El anfitrión también es capaz de compartir los resultados de la pregunta con la audiencia, por videoconferencia, o a través de la interfaz de la plataforma.
- **Guardar votaciones:** Cuando el anfitrión hace una votación se le da la opción de guardarla, de manera que pueda ser reutilizada posteriormente.
- **Compartir preguntas:** Las preguntas realizadas por los usuarios se guardan en el sistema y pueden ser publicadas para ser utilizadas por otros usuarios que presenten temas similares.
- **Salas de discusión:** El anfitrión puede separar a su audiencia en grupos más pequeños para que interactúen y discutan alguna temática presentada usando videoconferencia o chat de texto. Una vez finalizado el periodo de discusión el anfitrión puede llamar a la audiencia de vuelta a la sala principal.

2.3.4. Visión global

A continuación se presenta un resumen de las secciones anteriores.

	Videoconferencia	Votaciones	Reuso de votaciones	Compartir preguntas	Salas de discusión
Adobe Connect	✓	✓	✓		✓
Zoom	✓	✓			✓
Google Meet	✓				
Jitsi Meet	✓				
PollEverywhere		✓	✓		
SurveyMonkey		✓	✓		

Tabla 2.1: Comparación entre las distintas aplicaciones usadas para clases online

En la tabla se puede observar que si bien las plataformas Zoom y Adobe Connect presentan gran diversidad de características, ninguna de ellas hace uso de los resultados de votaciones para generar salas de discusión. Tampoco tienen un sistema de sugerencia de preguntas para hacer a la audiencia. Estas dos características serían de gran ayuda al momento de hacer una clase invertida, y más aún para instituciones o instructores que están migrando

a de un sistema de docencia tradicional a uno de aula invertida en línea. La primera, para generar mejores discusiones entre quienes atienden la clase y la segunda para ahorrar tiempo de preparación a quien expone. Además, si se guardan estadísticas sobre cada pregunta, se puede medir la calidad de éstas de forma más objetiva.

En los dos capítulos siguientes se presentarán las dos aplicaciones web que se desarrollaron durante este trabajo: Easy Poll y Easy Flip. Estas soluciones pretenden llenar los vacíos previamente descritos.

Capítulo 3

Easy Poll

Para comenzar con el desarrollo de la solución, se decidió implementar Easy Poll. Ésta es una aplicación web para crear preguntas de selección múltiple y compartirlas a través de un link con otros usuarios, que podrán entregar sus respuestas. La aplicación también permite ver la cantidad de respuestas que obtuvo cada alternativa. El software consta de dos partes, un cliente donde los usuarios pueden crear e interactuar con las preguntas, y un servidor, que almacena cada pregunta junto a las respuestas de los usuarios.

En este capítulo se hace un resumen de las características implementadas (sección 3.1), se describe la arquitectura de la solución (sección 3.2), se muestra la configuración (sección 3.3) y finalmente se expone como fueron construidos tanto el servidor (sección 3.4) como el cliente y sus interfaces (sección 3.5).

3.1. Características

La aplicación desarrollada tiene las siguientes características:

- Creación de preguntas: Cualquier persona que ingresa al sitio debe poder ingresar una pregunta, opciones de respuesta, y enviarla al servidor para ser almacenada.
- Respuesta de preguntas: Al enviar la pregunta, se genera un enlace. Cualquier persona con acceso al enlace puede acceder a una interfaz donde se selecciona una de las alternativas y se envía su respuesta al servidor.
- Visualización de respuestas: Para cada pregunta, existe una vista donde se puede ver el número de personas que contestó cada alternativa.

3.2. Arquitectura

Para conseguir las características deseadas, Easy-Poll usa el stack MERN (MongoDB, Express.js, React.js, Node.js). El cliente es una aplicación web, construida usando `React.js`, y el servidor es una API REST, hecha con `Express.js` y que se conecta con una base de datos `MongoDB`. El cliente envía peticiones a la API, que las procesa y luego las guarda en

la base de datos. En la figura 3.1 se muestra un diagrama ilustrando lo anterior. El servidor Express es utilizado tanto por Easy Poll como por Easy Flip, pero su código se encuentra en el los archivos del proyecto de Easy Poll.

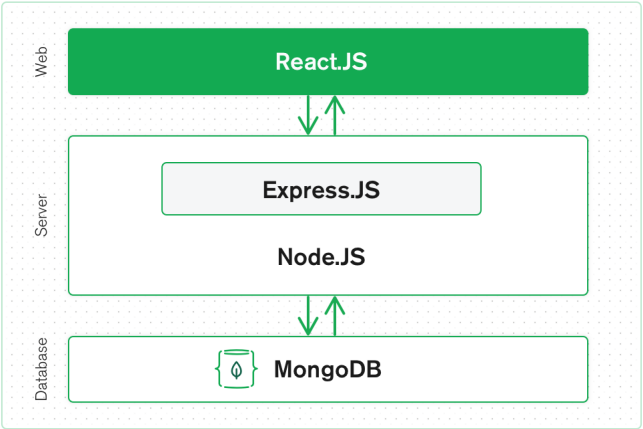


Figura 3.1: MERN

3.3. Configuración de Docker

La aplicación puede correr en cualquier computador que tenga Docker y docker-compose instalados. Cada una de las partes de la aplicación (servidor y cliente) viene con un archivo Dockerfile, que tiene todas las instrucciones para instalar la aplicación dentro de un contenedor y además la carpeta del proyecto tiene un archivo docker-compose, que indica cómo instalar cada uno de los contenedores y cómo éstos interactúan entre si. Este archivo además contiene instrucciones para construir un contenedor con el proceso de la base de datos.

El archivo docker-compose define tres contenedores, uno para la base de datos, que para instalarse usa la imagen oficial de Mongo en Docker Hub. El segundo y tercero usan las imágenes definidas en las Dockerfiles del cliente web y del servidor de la aplicación.

Para configurar los puertos en los que correrá cada servicio se debe crear un archivo llamado ".env" que tenga las siguientes variables:

- API_PORT: el número de puerto en que correrá el servidor de Easy Poll y Easy Flip.
- CLIENT_PORT: número de puerto en que correrá la aplicación web de Easy Poll.
- DB_PORT: número de puerto en que correrá el servicio de base de datos.

3.4. Servidor

Para que la aplicación funcione de manera correcta, es necesario que las peticiones enviadas por los clientes web sean procesadas por una aplicación que corre en el servidor. La implementación actual de esta aplicación contiene funcionalidades para el funcionamiento de Easy Poll y Easy Flip. En esta sección se ignora la lógica que involucra a Easy Flip.

3.4.1. Funcionamiento general

El servidor fue desarrollado en JavaScript, utilizando el framework Express.js. Lo primero que hace el servidor al empezar a correr es conectarse con la base de datos MongoDB, luego de eso comienza a escuchar peticiones en algún puerto, que está definido en las configuraciones del programa (3.4.2). Como los clientes web se conectarán a un servicio que se encuentra en un dominio distinto al del servidor, es necesario usar la librería *CORS* (Cross-Origin Resource Sharing) para procesar las peticiones.

3.4.2. Configuración

Para configurar algunas variables del servidor es necesario tener un archivo ".env" en la carpeta que lo contiene. Las variables definidas en este archivo son las siguientes.

1. API_PORT: Define el puerto en el que escuchará el servidor, por defecto es 9000 y si se usa la configuración de docker no debe cambiarse.
2. DATABASE_URL: Este valor es usado por la parte de Easy Poll, indica la url para conectar la aplicación a la base de datos.
3. SOCKET_IO_CORS_ORIGIN: Debe tener el valor de la url donde está el servidor web de Easy Flip, sirve para que el servidor escuche las peticiones pese a venir de un origen distinto.
4. SOCKET_IO_PATH: Por defecto es "/socket/" y no debería cambiarse. Define la ruta donde los clientes podrán conectarse a socket.io.

3.4.3. Peticiones

Todas las peticiones se hacen al dominio donde se encuentra el servidor, con */poll* al final. Por ejemplo, si la aplicación está alojada en api.easypoll.cl, las peticiones se harían a api.easypoll.cl/poll. En lo que sigue se detallan todas las peticiones que se pueden hacer. Todos los datos son enviados y recibidos en formato JSON.

GET /

Esta petición retorna un arreglo con todas las encuestas que hay en la base de datos.

GET /:pollId

Retorna la encuesta con identificador igual a pollId. Si no se encuentra dicha encuesta se lanza un error.

POST /

Su función es agregar una nueva encuesta a la base de datos. El cuerpo de esta petición solo lleva el texto de la pregunta y un arreglo con alternativas, el identificador de la encuesta es asignado por la base de datos al momento de ser agregadala encuesta.

POST /:pollId/vote

Esta petición registra el voto de un usuario para la encuesta con identificador igual a pollId. El cuerpo de esta petición lleva un string con el identificador de un usuario y un entero que indica el índice de la alternativa por la que votó.

POST /:pollId/reset

Al ser recibida esta petición se reinician los votos de la encuesta con el identificador correspondiente.

GET /:pollId/result

Esta petición entrega los resultados de la encuesta con identificador igual a pollId en un arreglo que contiene enteros. El valor en un índice representa la cantidad de votos recibido por la alternativa con dicho índice.

GET /:pollId/answers

Esta petición entrega un mapa con las respuestas de todos los usuarios a la pregunta con el identificador correspondiente. La diferencia con la petición anterior es que esta entrega los datos sin agregar.

3.4.4. Base de Datos

Para guardar las preguntas se utiliza una base de datos MongoDB, el modelo de datos es sencillo, ya que solo es necesario guardar preguntas. El único *schema* definido para la base de datos es el de una pregunta de alternativas, sus atributos son:

- `id:String` Un string aleatorio que identifica a la encuesta.
- `question:String` El texto de la pregunta.
- `alternatives:[String]`: Un arreglo de strings, cada uno de sus elementos contiene el texto de una alternativa.
- `votes:Map<String->Number>` Un diccionario que asocia el string identificador de un usuario con la alternativa que dicho usuario eligió al votar.

3.5. Cliente

El cliente web posee interfaces sencillas que permiten al usuario crear una encuesta y luego acceder a enlaces para ver los resultados y para votar en dicha encuesta. Al enviar los distintos formularios se hace una petición http al servidor, que envía una respuesta. Como ya se mencionó anteriormente, todo el cliente fue desarrollado usando React.js.

Cuando un usuario ingresa a la página, se chequea el *Local Storage* para ver si hay información asociada a Easy Poll en su navegador. Si no la hay, se genera un string aleatorio de 22 caracteres, este string es guardado en *Local Storage* bajo el nombre "userToken". Este string es usado posteriormente para identificar las respuestas del usuario y prevenir que pueda responder una encuesta múltiples veces.

En este trabajo el objetivo principal de Easy Poll es integrarse con Easy Flip, es por esto que las interfaces creadas son simples, pero funcionales. La aplicación tiene tres vistas, que se pueden acceder a través del navegador. Se usará la dirección `easyflip.cl` como ejemplo para la navegación. Todas las figuras de interfaces presentadas en esta sección corresponden a capturas de pantalla de la aplicación hechas usando el navegador Google Chrome.

3.5.1. Creación de encuesta

Para acceder a esta página un usuario debe ingresar a la dirección principal de la aplicación. La interfaz que se muestra tiene un campo de texto en la parte superior, donde el usuario puede ingresar la pregunta que desea hacer. Bajo eso tiene una lista con más cuadros de texto, en cada uno de ellos el usuario puede ingresar una alternativa de respuesta a la pregunta ingresada. Al lado derecho de cada una de las alternativas hay un botón con una "X", si el usuario da click a una de esas "X", la alternativa correspondiente es eliminada de la lista. Bajo la lista hay botones que dicen "Add alternative" y "Submit". El primero agrega una nueva alternativa a la lista y el segundo sirve para enviar el formulario y generar enlaces de votación y resultados (ver 3.2).

Al enviarse el formulario, se envía una petición HTTP "POST /" (ver 3.4.3) al servidor de Easy Poll, que registra la encuesta y la envía al cliente con toda su información. La información es desplegada bajo el formulario, con los enlaces bajo ellos. El enlace que dice "Vote here" lleva a la página de votación y el que dice "See results here" lleva a la página de resultados de la encuesta.

3.5.2. Votación

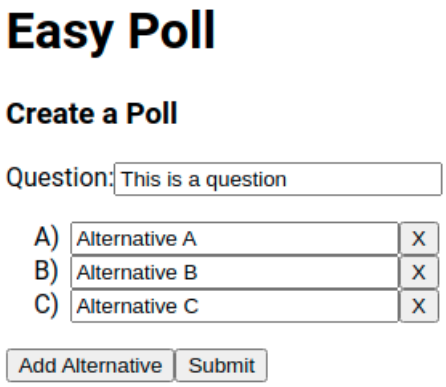
Para llegar a esta página se debe acceder a la url `easypoll.cl/:pollId/vote`, donde "pollId" representa el identificador de una encuesta. Normalmente esta url será compartida por quien genera la encuesta. Al ingresar a la página se envía la petición "GET /:pollId" al servidor para obtener los datos de la encuesta.

La interfaz muestra la pregunta, una lista con las alternativas y un botón para enviar la respuesta. A la izquierda de cada alternativa hay una caja que el usuario puede marcar para seleccionar la alternativa, sólo se puede marcar una alternativa a la vez.

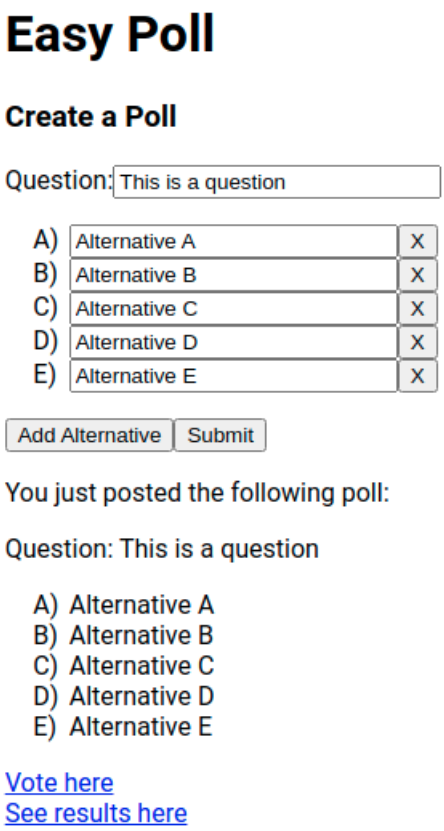
Luego de marcar una alternativa, el usuario debe presionar el botón "Submit" para enviar su respuesta, con lo que se envía al servidor la petición "POST /:pollId/vote" con el identificador del usuario y la alternativa seleccionada. Además de eso se muestra al usuario la alternativa que eligió. Si un usuario vuelve a votar, cambia la opción que escogió y no se agrega un voto adicional al sistema.

3.5.3. Resultados

Para acceder a la interfaz que muestra los resultados de la encuesta, se debe ingresar `easypoll.cl/:pollId/results` en el navegador, donde "pollId" es el identificador de la encuesta. Al entrar se piden al servidor los datos de la encuesta, al igual que para la interfaz anterior.

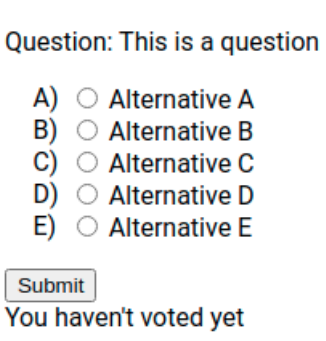


(a) Antes de enviar la encuesta

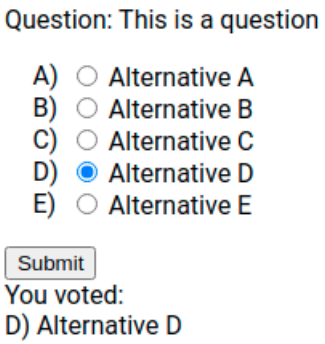


(b) Luego de enviar la encuesta

Figura 3.2: Creación de una encuesta



(a) Antes de enviar la encuesta



(b) Luego de enviar la encuesta

Figura 3.3: Interfaz de votación

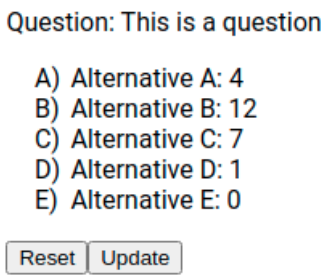


Figura 3.4: Interfaz de resultados

Aquí se muestra la pregunta, una lista con las alternativas seguidas de la cantidad de votos recibidos y dos botones (ver figura 3.4). El botón que dice "Update" pide los resultados al servidor con la petición GET /:pollId/result y actualiza los números con los recibidos. El botón reset envía la petición "POST /:pollId/reset" al servidor y hace que los resultados se reinicien.

Aunque se puede usar de manera independiente, Easy Poll es la base de la aplicación Easy Flip, que se describe en el capítulo siguiente, y se valida en el capítulo 5.

Capítulo 4

Easy Flip

Con el objetivo de crear una plataforma que permita a profesores y alumnos realizar clases invertidas por videoconferencia, se desarrolló Easy Flip. Esta aplicación consiste de un cliente web y un servidor, que se conectan en tiempo real. En este capítulo se describen las características de la aplicación (sección 4.1), su arquitectura (sección 4.2), y la implementación tanto del cliente (sección 4.3) como del servidor (sección 4.4).

4.1. Características

Las características de Easy Flip se detallan a continuación:

- Comunicación por videoconferencia usando Jitsi Meet: La aplicación usa la API IFrame de Jitsi-Meet para dar a sus usuarios la facultad de comunicarse por videoconferencia
- Sistema de preguntas de alternativas a los usuarios usando Easy Poll. Se implementó una interfaz que se conecta con el servidor de Easy Poll para generar preguntas de alternativas y compartirlas con los usuarios de la sala en tiempo real.
- Sistema de separación de la audiencia en subgrupos inteligentes, con opciones para agrupar a los participantes que tuvieron la misma respuesta juntos, o maximizar la diversidad de sus respuestas.

4.2. Arquitectura

La arquitectura de Easy Flip tiene cuatro componentes distintas: el cliente web de Easy Flip, la API Iframe de Jitsi Meet, la API de Easy Poll y finalmente el servidor de Easy Flip. Como ya se mencionó en el capítulo anterior, actualmente la API de Easy Poll y el servidor de Easy Flip están en el mismo programa, pero su lógica está desacoplada y por esto se consideran como partes distintas.

El cliente web de Easy Flip utiliza el IFrame de Jitsi Meet para mostrar la conferencia correspondiente a los participantes de una sala. Además de eso se conecta con Easy Poll a través de peticiones HTTP, para permitir a los clientes web crear, ver y responder preguntas.

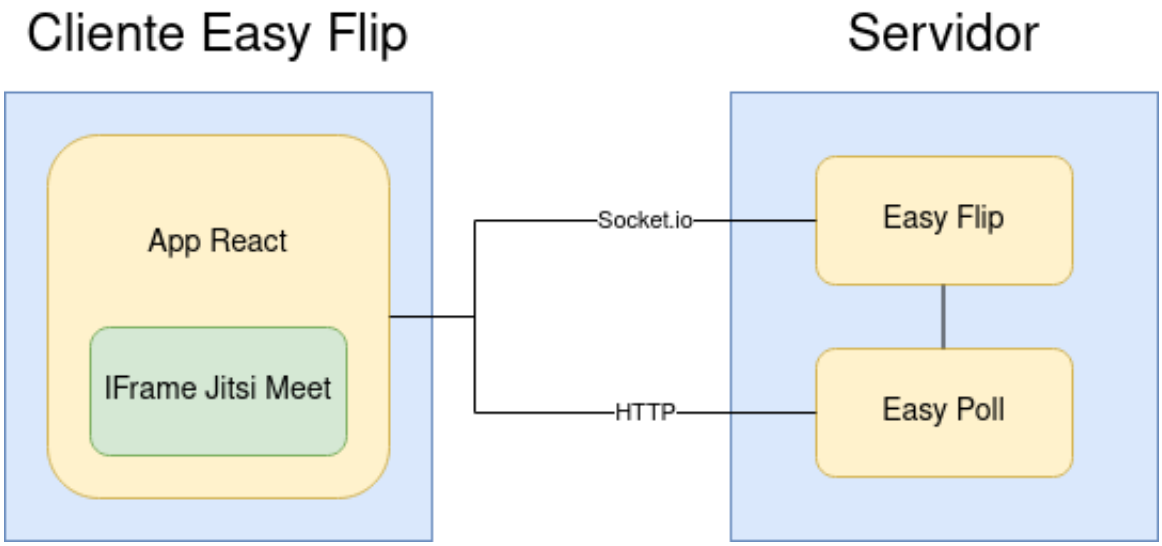


Figura 4.1: Arquitectura de Easy Flip

Para manejar las salas y mantener la sincronización de estas entre usuarios, es necesario que el cliente también esté conectado al servidor de Easy Poll, a través de Socket.io.

La lógica de Easy Flip también se comunica con la de Easy Poll en el servidor, para recibir la información de las encuestas de una sala al momento de crear grupos de discusión (ver 4.4.3). La figura 4.1 ilustra lo descrito.

4.3. Cliente web

El cliente web es una aplicación hecha con React.js donde los usuarios pueden crear y entrar a salas, donde pueden reunirse con otros usuarios usando Jitsi Meet como sistema de videoconferencias, y donde el dueño de una sala tiene el poder de crear preguntas para que el resto responda y para separarlos en grupos de discusión inteligentes. En esta sección se describe la configuración, la navegación por la aplicación y el funcionamiento de cada una de sus partes.

4.3.1. Configuración

Para configurar algunas variables de la aplicación se utilizaron variables de entorno que se encuentran en un archivo llamado ".env" en el directorio de la aplicación. Para el cliente web las variables configurables son:

1. REACT_APP_APIURL: Indica la url donde se encuentra la API de easy-poll.
2. REACT_APP_SOCKETIO_URL: Indica la url donde se encuentra alojado el servidor de Easy Flip
3. REACT_APP_SOCKETIO_PATH: Indica la ruta que debe llevar la url del item anterior para conectarse a socket.io. Debe calzar con el definido en el servidor (3.4.2. Por defecto su valor es "/socket/" y no debería cambiarse.
4. REACT_APP_JITSI_URL: Indica la url que se usará para acceder a las videoconfe-

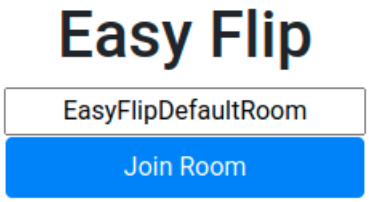


Figura 4.2: Página de inicio

rencias de Jitsi Meet. Si se desea conectar con la distribución oficial de jitsi su valor debe ser `meet.jit.si`.

4.3.2. Navegación

Para usar la aplicación, en primer lugar el usuario debe ingresar desde su navegador de preferencia a la URL que apunta a la aplicación. En este informe se usará `https://easyflip.repositorium.cl/` como ejemplo, ya que es la URL que se usó para hacer las pruebas durante el trabajo. La aplicación tiene dos vistas, la página de inicio y la vista de una sala, que se verán en las secciones siguientes.

Al ingresar a la aplicación a través de la url principal, el usuario es dirigido a la página de inicio (4.3.3). Si un usuario desea entrar directamente a una sala, sin necesidad de pasar por la página de inicio, puede hacerlo ingresando la url de la aplicación seguida por el nombre de la sala en su navegador. Por ejemplo si el usuario quisiera entrar a la sala con nombre "sala1", debería ingresar la url `https://easyflip.repositorium.cl/sala1` en su navegador.

4.3.3. Página de inicio

Esta es la página que se presenta al ingresar a la URL donde está alojada la página desde un navegador, sin agregarle un sufijo. La página de inicio es muy sencilla (ver figura 4.2, contiene un título, un campo de texto donde el usuario puede ingresar el nombre de una sala, y un botón para ingresar a la sala con dicho nombre.

Al pinchar el botón que dice Join Room, la aplicación dirige al usuario a la sala con el nombre ingresado en el campo de texto (en el caso de la figura 4.2 sería EasyFlipDefaultRoom), lo que cambia la URL a `https://easyflip.repositorium.cl/EasyFlipDefaultRoom` en su navegador y lo lleva a la interfaz de la sala.

4.3.4. Sala

La sala, definida como "Room" en el programa, es la componente principal de la aplicación web. Una sala tiene guarda como estado dos variables importantes, la primera es un objeto llamado "room". El estado de este objeto se pide al servidor en cuanto el usuario se conecta, enviando la señal "getRoomData" al servidor y luego recibiendo la señal "roomData" de vuelta. Esta señal trae toda la información de la sala actual, aunque para la aplicación solo

es relevante el padre de la sala y la encuesta actualmente activa en la sala. La segunda variable es el identificador de la encuesta actualmente activa en la sala, que sirve para conseguir y mostrar los datos de dicha encuesta cuando el usuario los solicita. Además de estas dos variables, se guardan un par de variables que indican si alguna componente se debe mostrar o no.

Además de lo anterior, la sala se encarga de mostrar dos componentes: "JitsiWindow" que contiene el IFrame de Jitsi Meet y un menú de usuario en la parte inferior. El menú mostrado depende de si el usuario es dueño de la sala o no. La sala entrega a estas componentes los datos que necesitan para que la información que muestran sea consistente con el estado de la sala.

Socket.io y Eventos

Al entrar a una sala, el cliente web se conecta con el servidor de socket.io y empieza a escuchar los eventos que se describen más abajo. Varios de estos eventos verifican que la sala actual sea efectivamente la sala que debe ser, por el comportamiento al ingresar a salas de discusión o tener la aplicación abierta en varias pestañas.

Además, al conectarse al servidor con Socket.io, se revisa que haya un valor llamado "sessionID" en el *Local Storage*. Si existe, entonces se envía este valor junto a la petición para conectarse con Socket.io. Si no existe, entonces el servidor crea un valor y lo envía al usuario usando el evento "session".

1. `pollChanged(fromRoomName, newPollId)` Actualiza la encuesta actual en la sala con la nueva y carga los datos de la nueva para ser mostrados. Además de eso se quita la vista de resultados para los participantes de la sala.
2. `notifyBreakout(breakoutRoomName, originalRoomName)`: Al recibirse este evento se verifica que el nombre actual de la sala sea igual a originalRoomName y si se cumple dicha condición se cambia la url para ir a la sala con nombre breakoutRoomName.
3. `returnToMainRoom(mainRoomName)` Si el nombre de mainRoomName calza con el nombre del padre de la sala actual, entonces muestra una notificación al usuario para volver a la sala principal.
4. `forceReturnToMainRoom(mainRoomName)` Funciona igual que el evento anterior, pero en vez de avisar al cliente que vuelva, lo fuerza a volver.
5. `showResults(fromRoomName)` Si la sala actual es la correspondiente, entonces hace que se muestren los resultados de la encuesta.

4.3.5. Menú del dueño de una sala

El dueño de una sala puede ver un menú separado en dos secciones bajo la conferencia de Jitsi. La sección superior contiene botones con opciones relacionadas a las encuestas y la sección inferior contiene botones para generar y administrar los grupos de discusión (ver 4.3). En lo que sigue se describirán las funciones de cada botón.

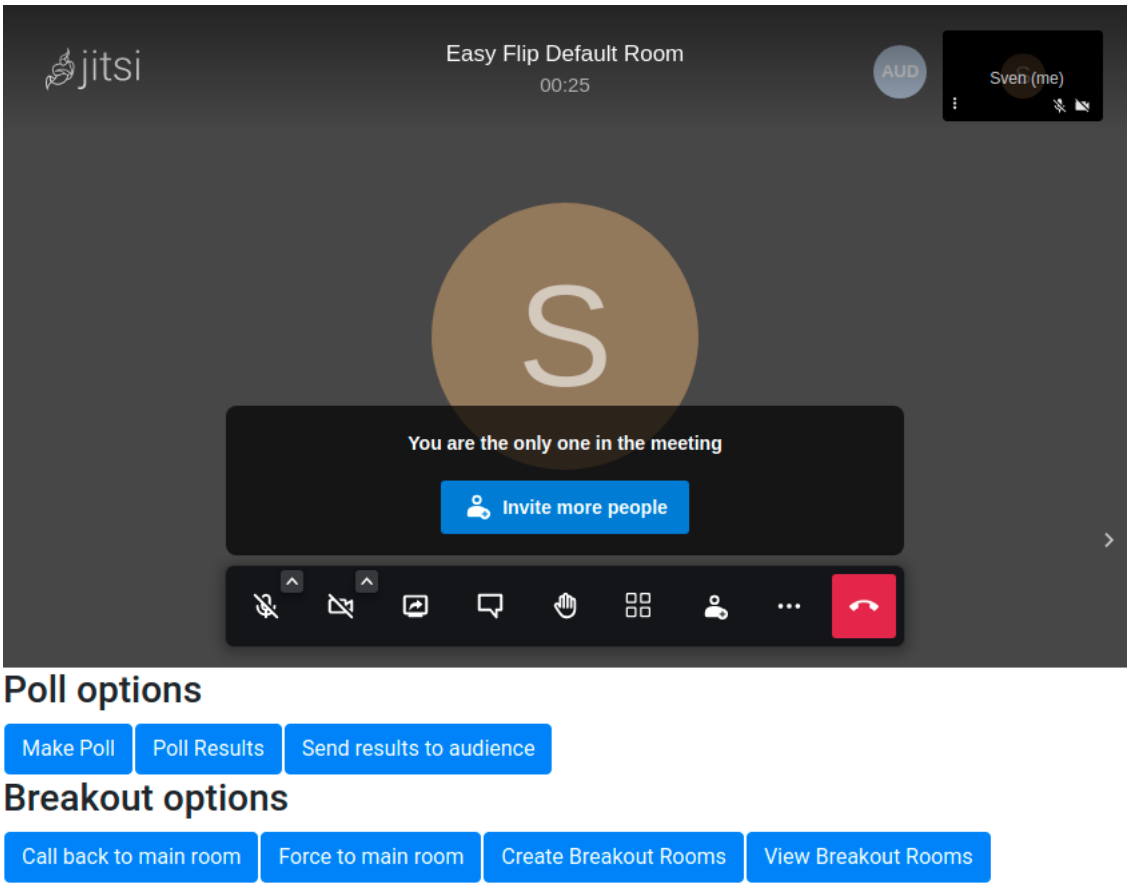


Figura 4.3: Interfaz para el dueño de una sala

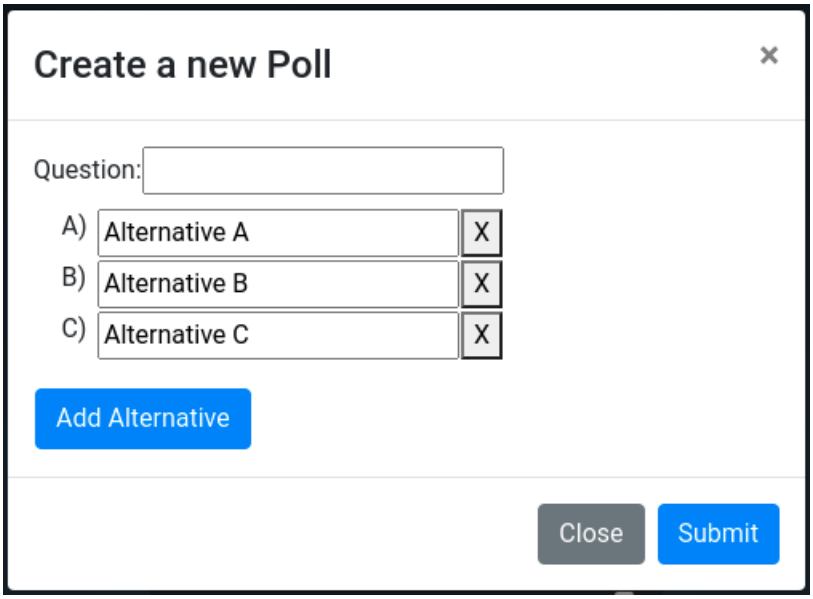


Figura 4.4: Interfaz para crear encuesta

Make Poll

Este botón despliega un diálogo que permite al dueño de la sala crear una nueva encuesta (ver figura 4.4). El botón que dice "Add Alternative" agrega una nueva alternativa a la lista. Al presionar "Submit" se envía una petición POST http al servidor de Easy Poll, donde se registra la encuesta y se reciben sus datos. Luego de eso se emite el evento "setPollId" con el nombre de la sala actual y el id de la encuesta creada (recibido en la respuesta de la petición) al servidor de Easy Flip, que luego avisa a los demás clientes en la sala que la encuesta ha cambiado.

Poll Results

Cuando se presiona este botón, se muestra al usuario la encuesta junto con sus resultados y un botón que dice "Update". (ver figura 4.5). Al lado de cada alternativa se puede ver cuántos usuarios escogieron dicha alternativa. Para conseguir los resultados se envía una petición HTTP GET /:pollId/result (3.4.3 al servidor de Easy Poll. Al presionar el botón "Update" se vuelven a pedir los resultados y se actualiza la lista que los muestra con los nuevos valores.

Send results to audience

Al seleccionar esta opción, se envía la señal "showResults" al servidor, para notificar a los clientes web que pueden ver los resultados de la encuesta.

Create Breakout Rooms

Este botón muestra un diálogo que permite al dueño de la sala configurar y crear salas de discusión (4.6). El diálogo contiene varias opciones para separar a los participantes de la sala en grupos. La primera opción permite decidir si se quiere generar cierta cantidad de grupos o si se quieren generar grupos de cierto tamaño. En la segunda opción se debe

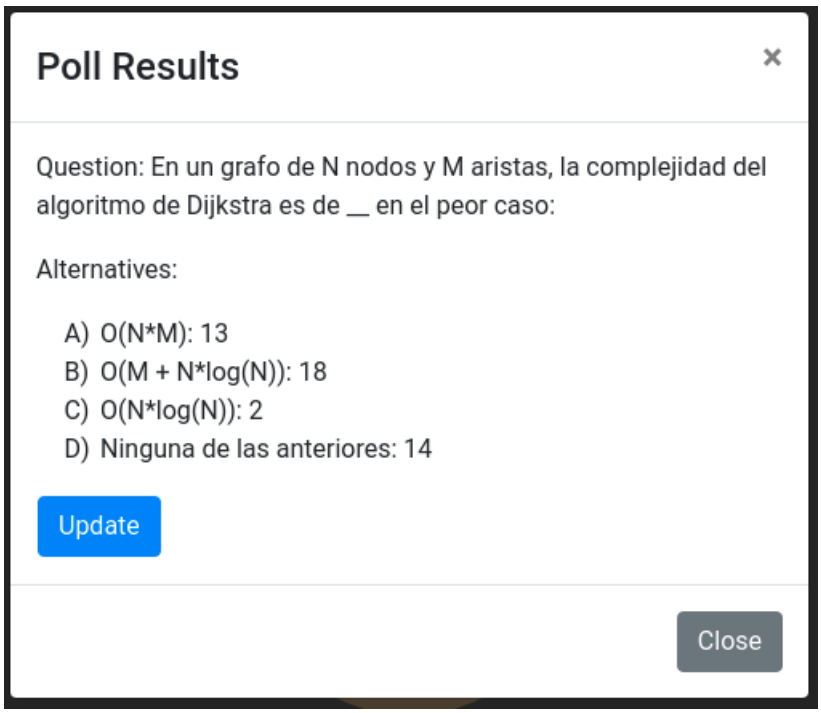


Figura 4.5: Interfaz de resultados

ingresar un número, que sirve para seleccionar el número de grupos o su tamaño. La última opción es para decidir la forma en la que se crearán los grupos, "Random" para generación aleatoria, "Same answers together" para dejar juntos a los participantes que respondieron igual y finalmente "Different answers together" para maximizar la diversidad dentro de cada grupo. Los algoritmos para generar estos grupos se describen en 4.4.3.

Call back to main room

Envía el evento "callToMainRoom" al servidor, para notificar a todos los usuarios que se encuentran en salas de discusión que vuelvan a la sala principal.

Force to main room

Envía el evento "Force to main room" al servidor, que fuerza a los usuarios a volver a la sala principal.

View breakout rooms

Muestra una ventana con una lista de las salas de discusión actualmente activas. Esto se hace enviando la señal "getBreakoutRooms" al servidor para pedirle la información sobre dichas salas. (ver 4.7).

Los datos se muestran como una tabla, donde la primera columna indica el nombre de la sala de discusión, la segunda columna muestra el número de participantes que hay en ella y la tercera tiene un botón que permite ingresar a la sala. Al presionarse el botón la aplicación dirige al usuario a la sala de discusión y el menú de usuario pasa a ser igual al del participante en sala de discusión con un botón para volver a la sala principal (4.3.6.

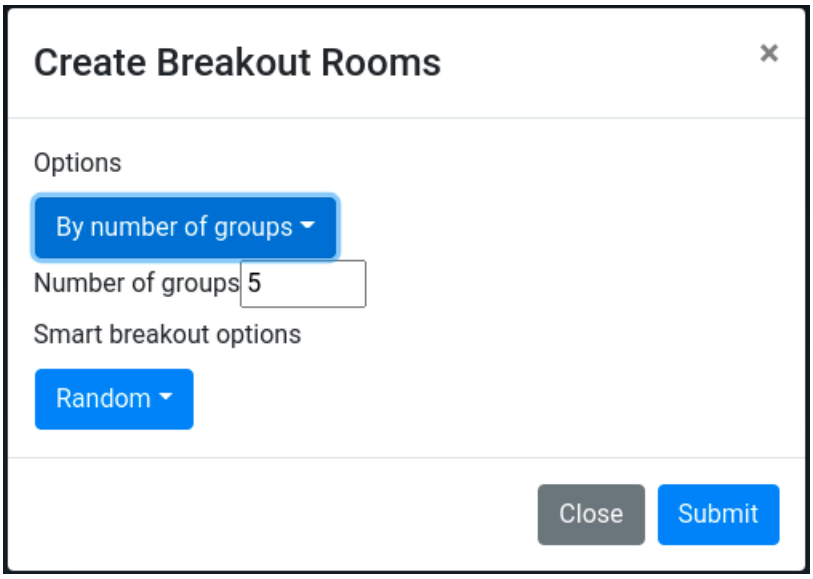


Figura 4.6: Interfaz para crear salas de discusión

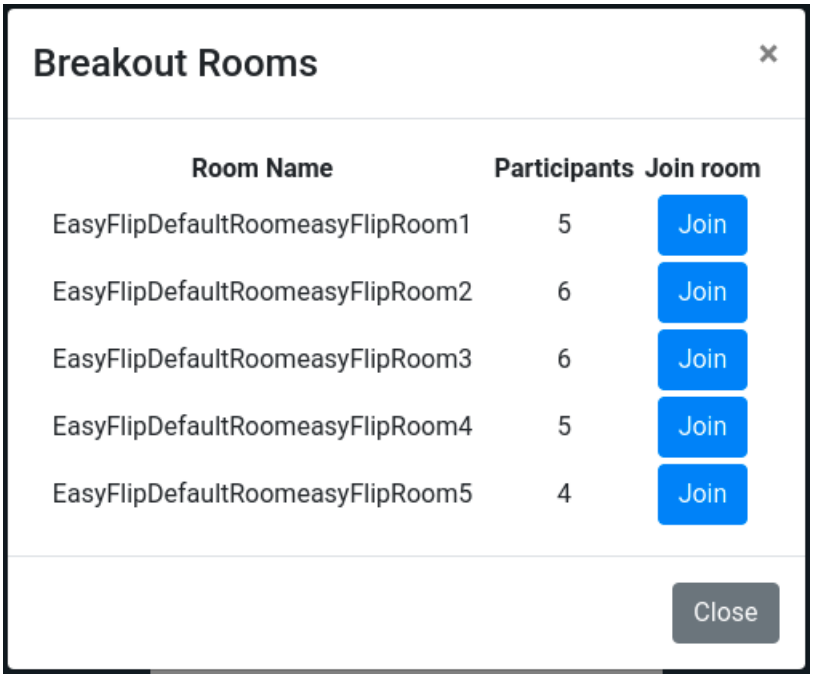


Figura 4.7: Interfaz para ver salas de discusión activas

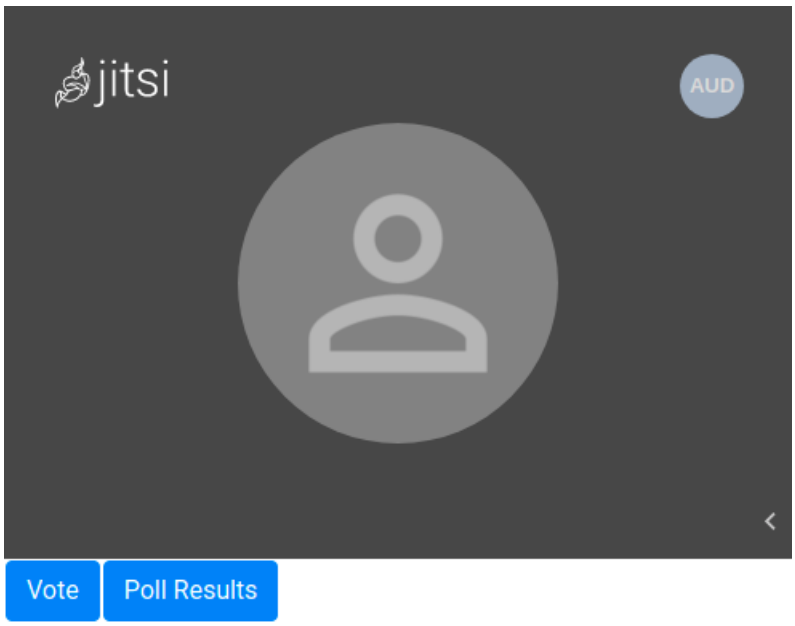


Figura 4.8: Vista de participante

4.3.6. Menú de participante

Un participante en una sala puede ver un cuadro que muestra la conferencia de Jitsi a la que está conectado y un menú con botones en la parte inferior de este. El botón que dice "Poll Results" sólo se muestra si el dueño de la sala lo indica, de manera que los participantes no elijan sus respuestas en base a lo que ven en los resultados (ver 4.8).

Menú de votación

Al presionar el botón que dice "Vote", los usuarios podrán participar en la pregunta que el dueño de la sala ha configurado. Si no se ha configurado ninguna pregunta aún, entonces al presionar el botón aparecerá un mensaje indicando la situación. Para enviar su respuesta al servidor de Easy Poll, el usuario simplemente deberá marcar la casilla desee y luego presionar el botón "Submit" (ver figura 4.9). Al presionar el botón, aparecerá abajo un texto indicando la opción escogida por el usuario.

El botón que dice "Poll Results" muestra la misma interfaz que para el dueño de la sala (ver figura 4.5) y su funcionamiento es idéntico al descrito anteriormente.

Menú de participante en sala de discusión

La única diferencia entre la interfaz de un usuario en una sala de discusión y la sala principal, es que en la sala de discusión aparece un nuevo botón que dice "Return to main room" (ver 4.10). Al presionarse este botón, se envía una señal al servidor para dejar la sala actual, y además se cambia la url actual de la aplicación a la url de la sala principal. Esto hace que la reunión de jitsi también cambie.

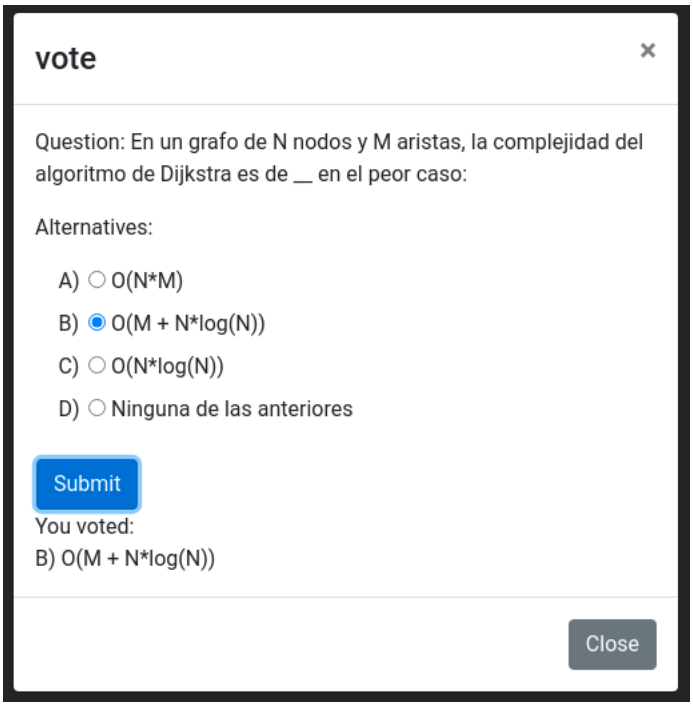


Figura 4.9: Interfaz de votación

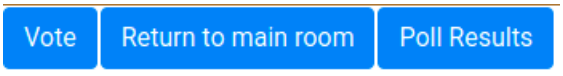


Figura 4.10: Menú de participante en sala de discusión

4.4. Servidor

Como ya se mencionó anteriormente, los clientes web de Easy-Flip deben estar en comunicación constante con el servidor para mantener sincronizado el estado de una sala entre todos los clientes, para mantener esta conexión se utilizó la biblioteca socket.io, que mantiene conexiones entre los clientes y el servidor a través de websockets. En esta sección se describen los eventos que escucha el servidor y los procesos que ejecuta en respuesta a dichos eventos, además de la forma en que el servidor almacena el estado.

4.4.1. Gestión de salas y usuarios

Salas

Para guardar toda la información relevante al estado de una sala se definió definió la clase `Room`. Sus campos se describen en lo que sigue:

1. `parent`: String que indica el nombre de la sala principal (la sala "padre"), es indefinido si la sala es una sala principal.
2. `roomName`: String que indica el nombre de la sala.
3. `owner`: String que indica el id del usuario que creó la sala.
4. `connectedUsers`: Set que guarda los ids de todos los usuarios conectados a la sala.
5. `pollId`: String que indica el id de la encuesta activa en la sala actualmente. Es indefinido por defecto.
6. `breakoutRooms`: Arreglo que contiene los nombres de todas las salas de discusión creadas desde esta sala.

Toda la información de las salas se guarda en una instancia de una clase llamada *InMemoryRoomStore*, que hereda de una clase abstracta llamada *RoomStore*. La ventaja de definir una clase abstracta en este caso es que si se quiere escalar la aplicación en el futuro, entonces se puede implementar una nueva clase que implemente los sus métodos y guarde la información de otra forma, por ejemplo usando una base de datos en memoria como Redis [18].

`RoomStore` define los siguientes métodos, que `InMemoryRoomStore` implementa:

1. `newUser(string: userId)`: Guarda el id de un nuevo usuario en memoria.
2. `removeUser(string: userId)`: Elimina el id de un usuario de la memoria.
3. `newRoom(string: roomName, string: ownerId, string: parentRoomName)`:
   Crea una nueva sala y la almacena en memoria. El nombre de la sala es roomName, el Id de su dueño es ownerId, y parentRoomName indica cual es el padre de la sala de ser necesario. Si la sala no tiene padre, entonces no es necesario especificar ese campo.
4. `getRoom(string: roomName)`: Obtiene la sala que tiene el nombre roomName.
5. `getUsers(string: roomName)`: Obtiene todos los usuarios de la sala indicada.
6. `joinRoom(string: roomName, string: userId)`: Agrega al usuario con id igual a userId a la sala con el nombre indicado. Si la sala no existe, entonces es creada por este mismo método. Retorna la sala a la que entró el usuario.

7. `leaveRoom(string: roomName, string: userId)`: Quita al usuario indicado de la sala con nombre roomName. Si la sala queda vacía una vez el usuario sale, entonces la sala es eliminada, para evitar fugas de memoria.
8. `getAllRooms()`: Retorna un arreglo con todas las salas actualmente almacenadas en memoria.
9. `getRoomsForUser(string: userId)`: Retorna un arreglo con todas las salas donde el usuario con el id indicado está actualmente.
10. `setPoll(string: roomName, string: pollId)`: Actualiza el id de la encuesta de la sala indicada.
11. `getPoll(string: roomName)`: Obtiene el id de la encuesta que está actualmente activa en la sala de nombre roomName.
12. `getBreakoutRoomsForRoom(string: roomName)`: Retorna un arreglo con todas las salas de discusión que están actualmente activas para la sala con nombre roomName.

Almacenamiento de usuarios

Es necesario tener información de los usuarios que usan el sistema para poder asociarlos con sus respuestas a las preguntas, es por esto que cada vez que se establece una conexión al servidor, se guarda un string que identifica al usuario conectado. Este string, llamado *userID*, es compartido entre el cliente y el servidor.

Con este objetivo en mente y con un diseño similar al de gestión de salas, se creó una clase abstracta llamada *InMemorySessionStore*, que hereda de la clase abstracta *RoomStore*. Esta clase almacena información para identificar los sockets conectados. Este sistema es particularmente útil para persistir los datos de un usuario que se desconecta y reconecta, lo que puede hacer que se pierda información sobre el socket que se estaba utilizando, pero los datos de la sesión se mantendrán en el servidor.

InMemorySessionStore almacena sesiones en un diccionario, que asocia *sessionID* con un objeto que tiene solo dos atributos, uno es el *userID* y el otro es un valor booleano llamado *connected*, que indica si la sesión está activa o no.

Esta clase tiene tres métodos, uno retorna la información de una sesión asociada al identificador que se entrega como argumento. Otro que recibe un identificador y un objeto del tipo *session* y guarda su estado en memoria, y finalmente un método que entrega todas las sesiones actualmente almacenadas.

4.4.2. Socket.io

Cuando un cliente intenta conectarse al servidor con Socket.io, lo primero que hace el servidor es revisar si la petición para conectar viene con un valor llamado *sessionID*. Si la petición viene con *sessionID*, entonces se guarda ese valor en el socket y se busca el *userID* asociado a esa sesión para guardarlo en el socket. Si el identificador de sesión no existía, entonces se crea uno nuevo, junto con un nuevo *userID*. Luego de esto, se guardan estos valores en la *sessionStore* y se envía al cliente recién conectado el evento "session", que contiene el *userID* y *sessionID* recién mencionados. Finalmente, el socket ingresa a la sala

de socket.io llamada userID y empieza a escuchar eventos emitidos por el cliente, que se enumeran en lo que sigue:

1. `joinRoom(roomName)`: Al recibir este evento, se registra en memoria al usuario que lo envió como participante de la sala con el nombre "roomName" entregada por el evento.
2. `getRoomData(roomName)`: Sirve para solicitar los datos de una sala al servidor. Cuando se recibe este evento se emite un evento llamado roomData al cliente. Dicho evento contiene los datos de la sala solicitados.
3. `sendToBreakout(roomName, size, breakoutOption, smartBreakoutOption)`: Al recibir esta señal por parte del dueño de una sala, se generan los grupos de discusión y se envía a todos los usuarios participantes de la sala indicada la señal para ir a la sala de discusión que les corresponde (ver 4.4.3).
4. `callToMainRoom(mainRoomName)`: Envía a todos los sockets en la sala de nombre "mainRoomName" la señal para notificarles volver a la sala principal.
5. `forceToMainRoom(mainRoomName)`: Similar a la función anterior, envía a todos los sockets en la sala mainRoomName la señal para forzarlos volver a la sala.
6. `setPollId(roomName, pollId)`: Cuando el dueño de la sala envía este evento, se guarda la encuesta con id "pollId" para la sala.
7. `showResults(roomName)`: Envía la señal a la sala para mostrar los resultados de la encuesta.
8. `leaveRoom(roomName)`: El usuario que envió la señal se elimina de la lista de usuarios conectados a la sala (llamando a la función `leaveRoom` de `roomStore`.
9. `getBreakoutRooms(roomName)`: Al recibirse este evento se envía al usuario que lo envió una lista con todas las salas de discusión creadas desde la sala con nombre roomName.
10. `disconnect()`: Este evento es emitido automáticamente por los sockets del cliente cuando se produce una desconexión, sin importar el motivo de ésta. Cuando llega este evento, se revisa que todos los sockets asociados al usuario estén desconectados. Si se cumple esa condición, se quita al usuario de todas las salas a las que pertenecía, para hacer limpieza de los datos en memoria.

4.4.3. Creación de grupos

Un aspecto importante a destacar del funcionamiento del programa, es la forma en la que se crean los grupos de discusión. La función para separar a los participantes en grupos de discusión tiene los siguientes parámetros: el nombre de la sala, un número que indica el tamaño de los grupos o la cantidad de grupos a crear y dos strings con opciones. El primero indica si los grupos se quieren crear por tamaño o por número de grupos y el segundo indica la opción de *smart breakout rooms*, que puede ser creación aleatoria, maximizando la diversidad de respuestas a una pregunta, o dejando juntos a los que respondieron igual. El código usado por la aplicación para generar los grupos se encuentra en el apéndice 6.2.

Luego de recibirse la señal, se revisa que la encuesta actual de la sala sea válida. Si la encuesta no es válida, entonces se crea una encuesta vacía. Luego de eso, se llama a la función que crea los grupos. Esta función recibe los datos mencionados anteriormente además de la lista de participantes (sin el dueño) y los datos de la encuesta.

Las funciones que generan la distribución de los grupos sólo trabajan con la cantidad de grupos a crear y no con el tamaño de los grupos, es por esto que si la opción es por tamaño, se hace la conversión:

$$N = \lfloor P/T \rfloor \tag{4.1}$$

Donde N es la cantidad de grupos a crear, P es el número de participantes y T es el tamaño de grupo deseado. De esta forma se asegura que los grupos tengan una cantidad de participantes entre T y $T + 1$.

Después de hacer esta conversión, se entregan los argumentos a una de las tres funciones que generan una distribución. Dichas funciones se describen a continuación.

Distribución aleatoria

Esta función recibe un entero que define la cantidad de grupos a crear y un arreglo A con los identificadores de los usuarios a distribuir. Sigue los siguientes pasos:

1. Mezcla A, para que la distribución sea realmente aleatoria.
2. Crea un arreglo D con N arreglos vacíos para guardar la distribución final.
3. Itera por el arreglo de usuarios, agregando al usuario con índice i al arreglo de D con índice i mód N

Distribución por respuestas iguales

Este método, al igual que el anterior, recibe la cantidad de grupos a crear y un arreglo A con los usuarios, pero además de eso recibe los datos de una encuesta. La idea del algoritmo es separar a todos los participantes en grupos según sus respuestas y luego de eso subdividir esos grupos en grupos más pequeños, dependiendo del tamaño de grupo deseado. El algoritmo funciona como sigue:

1. Separa a los usuarios un arreglo B, que contiene $x+1$ arreglos, donde x es la cantidad de alternativas de la encuesta entregada. En cada sub-arreglo de B se guardan los usuarios que respondieron la alternativa correspondiente al índice de dicho sub-arreglo. En el último índice de B se guardan los usuarios que no entregaron respuesta.
2. Se calcula el tamaño esperado de cada grupo, haciendo la operación inversa de 4.1: $T = \lfloor P/N \rfloor$
3. Se itera por cada sub-arreglo B_{i} de B, calculando la cantidad de subgrupos para dicho sub-arreglo y luego llenando estos subgrupos de la misma forma que en la distribución aleatoria.

Distribución maximizando la diversidad

Esta función recibe la cantidad de grupos a crear, un arreglo A con los usuarios y los datos de una encuesta. El algoritmo intenta crear los grupos de discusión de forma que los participantes de cada grupo tengan respuestas distintas entre sí. Para lograr este objetivo el algoritmo sigue los siguientes pasos:

1. Al igual que el primer paso del algoritmo anterior, se separa a los usuarios un arreglo B, que contiene $x + 1$ arreglos, donde x es la cantidad de alternativas de la encuesta

entregada. En cada sub-arreglo de B se guardan los usuarios que respondieron la alternativa correspondiente al índice de dicho sub-arreglo. En el último índice de B se guardan los usuarios que no entregaron respuesta.

2. Se inicializa un contador c en 0 y un arreglo D, que contiene una cantidad de arreglos vacíos igual a la cantidad N de grupos que se quiere generar.
3. Para cada uno de los arreglos B_{i} de B, mientras B_{i} no esté vacío se siguen los siguientes pasos:
   (a) Se quita el primer elemento de B_{i} y se agrega al arreglo D_c
   (b) Se hace la operación $c = (c + 1) \mod N$
4. Se retorna el arreglo D con la distribución

De esta manera se asegura que cada grupo tenga una cantidad similar de participantes con cada respuesta entregada, ya que los participantes de cada respuesta son repartidos de forma equitativa entre los grupos.

En el próximo capítulo se describe el proceso de validación con usuarios de la aplicación Easy Flip, que también sirve de forma indirecta para validar Easy Poll.

Capítulo 5

Validación

Para validar el software, se instalaron todas las partes de la aplicación en un servidor del Departamento de Ciencias de la Computación (DCC) de la Universidad de Chile. En este capítulo se hace una breve descripción de la instalación del software (sección 5.1), luego se presentan las tres pruebas realizadas con usuarios (sección 5.2) y finalmente se discute el resultado de éstas (sección 5.3)

5.1. Instalación

Durante el desarrollo de este trabajo se mantuvo constantemente una versión del software funcionando en un servidor del Departamento de Ciencias de la Computación de la universidad de Chile. La máquina, llamada Búho, tiene las siguientes características de interés. La instalación de la aplicación fue relativamente fácil gracias a Docker y la ayuda del personal de sistemas de la universidad.

- Procesador: Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz
- Memoria RAM: 4GB

En la misma máquina además se mantuvo un servidor de Jitsi Meet, para no estar restringidos por las limitaciones de la distribución oficial. Para las pruebas se utilizó la versión de Jitsi instalada en búho, y la distribución oficial de Jitsi Meet: `meet.jit.si`, dado que la instalación propia no funcionó bien en reuniones grandes.

5.2. Pruebas

Durante el desarrollo del software, este se sometió a pruebas locales para comprobar que sus partes funcionaran de manera adecuada. Además de esto, el software se probó en tres instancias, con distintos grupos de usuarios. En esta sección se describirá el proceso de las tres pruebas hechas, ordenadas por fecha.

5.2.1. Prueba 1: clase con 94 alumnos

Esta prueba fue realizada el 18 de marzo de 2021 y se utilizó la configuración mencionada anteriormente, usando la instalación de Jitsi Meet en Búho.

La primera prueba del software se hizo al comienzo de una clase del curso "Diseño y análisis de algoritmos", con una clase de 94 alumnos. La clase comenzó en una reunión de Zoom, donde el profesor avisó a los alumnos que se haría una prueba de software. Luego de eso publicó el enlace en el chat de Zoom y se intentó hacer la transición a Easy Flip. Una vez conectados la mayoría de los alumnos el profesor intentó hablar, pero las frases se escuchaban cortadas y no se registraba video, lo que hacía imposible la comunicación.

Al revisar el historial de eventos del servidor, se observó que aproximadamente 60 alumnos lograron abrir la aplicación y quedar registrados en una sala. No se logró determinar la cantidad exacta de alumnos que efectivamente abrieron la aplicación, pero se estima que todos o casi todos los que lo intentaron quedaron registrados.

En el momento, se atribuyó el error a haber usado un servidor de Jitsi alojado en una máquina sin el poder de procesamiento o conexión a internet suficiente para manejar tal cantidad de alumnos, y como se trataba de una clase oficial se decidió terminar con la prueba luego de cinco minutos de intentos de comunicación fallidos. La causa del error fue confirmada en la segunda prueba.

5.2.2. Prueba 2: grupo de 8 voluntarios

Para replicar el error de la prueba anterior, se hizo otra prueba el día 20 de marzo de 2021, esta vez con siete usuarios voluntarios, además del autor de este informe. Nuevamente se probó usando la instalación de Jitsi Meet en Búho. El error de audio persistió, por lo que se decidió cambiar la configuración para usar la distribución oficial de Jitsi, lo que solucionó el problema.

Durante esta prueba se probaron las funcionalidades del software. El sistema de encuestas funcionó correctamente, pero al hacer pruebas con el sistema de salas de discusión se descubrió que al enviar a los usuarios a salas de discusión, algunos de estos quedaban en la sala principal.

El error se atribuyó al evento que agregaba usuarios a una sala, que llamaba a una función asíncrona y probablemente generaba una situación de *data race* en que no se mantenía la consistencia en el almacenamiento de usuarios cuando estos entraban y salían de salas de discusión.

Luego de esta prueba se hicieron cambios en el código para solucionar dicho error, y se cambiaron levemente los algoritmos de generación de grupos, ya que también se encontraron errores en estos en las pruebas locales. En la tercera prueba se comprobó que los cambios hechos al código efectivamente solucionaban el problema, ya que este no se repitió.

5.2.3. Prueba 3: alumnos voluntarios

Para esta prueba, que fue hecha el 25 de marzo de 2021, nuevamente se intentó con el curso de "Diseño y análisis de algoritmos", pero con un subconjunto voluntario de alumnos

de ésta, y quince minutos antes del horario oficial de la clase, usando la distribución oficial de Jitsi Meet en la configuración. Desafortunadamente llegaron solamente cinco alumnos a la prueba, por lo que no se pudo hacer una prueba tan realista como se deseaba.

El profesor entregó distintas preguntas de alternativas, relacionadas con la materia del curso, a la audiencia usando la aplicación. Luego de que los participantes hubieran contestado la encuesta, los envió a salas de discusión. El proceso se repitió tres veces, con distintas preguntas. En esta prueba todas las funciones de la aplicación se comportaron de manera correcta.

5.3. Discusión

Dados los resultados de la primera y segunda prueba, se puede intuir que si se quiere tener una configuración de la aplicación que no dependa de una distribución externa de Jitsi Meet, se necesitarían mejores máquinas para servir una distribución propia de Jitsi. Tener una distribución propia es relevante si se quieren tener reuniones con más de 75 personas.

En cuanto a la usabilidad del programa, ninguno de los usuarios con que se probó en la segunda y tercera prueba tuvo problemas para responder las encuestas ni para navegar entre las salas, por lo que se concluye que la aplicación no es difícil de usar. Esto es de esperarse, dado que los menús de los usuarios son bastante básicos.

También cabe mencionar que, considerando la última prueba realizada, el programa funciona correctamente y que no se han encontrado errores graves que hagan que la aplicación deje de funcionar. No se sabe cómo funcionará la aplicación con una cantidad más grande de usuarios concurrentes, ni tampoco se ha probado tener más de una clase activa en la plataforma, por lo que no se tienen resultados concluyentes respecto a cómo funcionaría la aplicación si esta fuera distribuida de forma más masiva.

Capítulo 6

Conclusión

Para concluir este informe, se hace una evaluación del trabajo realizado (sección 6.1) y se enumeran características y formas de mejorar las aplicaciones desarrolladas (sección 6.2).

6.1. Trabajo realizado

El software desarrollado cumple con los objetivos 1, 2 y 3, relacionados con la creación de grupos basados en las respuestas de los alumnos a preguntas de selección múltiple, planteados al comienzo. La plataforma tiene la capacidad para conectar personas a través de videoconferencia, usando Jitsi Meet para ello. Además, el dueño de una sala puede agregar preguntas de alternativas que los participantes luego pueden contestar y se tiene la capacidad para enviar a la audiencia de la sala a subgrupos, donde pueden discutir sus respuestas a las preguntas. La división en subgrupos se puede hacer de forma aleatoria o en base a las respuestas entregadas, tanto juntando a quienes respondieron de la misma forma o intentando maximizar la diversidad de respuestas dentro de cada sala. Sumado a lo anterior, el trabajo realizado sienta una buena base para seguir explorando el concepto de grupos de discusión inteligentes. Basándose en Easy Flip, a futuro se pueden generar nuevos paradigmas para separar una audiencia.

Los objetivos 4 y 5 fueron logrados solamente de forma parcial. El objetivo específico número 4, que consiste en permitir la búsqueda y reutilización de preguntas de alternativas para facilitar la colaboración entre profesores, sólo se formalizó y será el tema de la tesis de magíster de otro alumno (Joaquín Cruz). Pese a haberse desarrollado una plataforma especial para guardar preguntas, no se alcanzó a implementar un buscador u otro sistema que permita reutilizarlas. El objetivo específico 5, sobre validar el software con usuarios, tampoco se cumple, porque aunque se hicieron pruebas, las condiciones de éstas no permitieron preguntar a los asistentes por la calidad del software, en la primera por una falla técnica con el servidor de Jitsi usado y en la segunda por la baja cantidad de asistentes.

La aplicación tiene características que pueden ayudar a los profesores a mejorar sus clases invertidas, y que los programas para videoconferencias que existen hoy en día aún no implementan. El software también puede ser útil para validar una hipótesis de investigación

respecto al impacto en el aprendizaje que puedan tener los grupos de discusión inteligentes (tarea que se encuentra fuera del alcance de este trabajo de título). Además de eso, el software es de código abierto y puede ser modificado y mejorado por quien tenga interés. Cualquier persona que tenga acceso a un servidor puede instalarlo y correr su propia versión de Easy Flip.

Un último beneficio a considerar sobre la aplicación, es que si bien la versión actual usa Jitsi Meet como sistema de videoconferencia, esta es independiente de Jitsi Meet. Esto hace que si a futuro se inventa una nueva plataforma web para videoconferencias, o si se desea implementar Easy Flip con otro software ya existente, el trabajo ya realizado aún es válido.

6.2. Trabajo futuro

El trabajo realizado aún tiene falencias y un gran potencial de mejora. Algunas características y aspectos a mejorar se listan a continuación:

1. Mejora general a la interfaz: Para esta memoria no se dio mucho énfasis a la experiencia de usuario ni a la estética de las interfaces. Se sugiere trabajar en mejorar los menús para que sean más agradables a la vista y mejorar el feedback entregado por el programa al interactuar con este. Se estima que una persona con experiencia en React.js y desarrollo de frontend tarde no más de dos semanas en lograr una buena mejora.
2. Mejorar la administración de las salas de discusión: La interfaz de Zoom para los grupos de discusión permite al dueño de una sala armar a mano los grupos, poner un temporizador antes de forzar a los participantes a volver y también enviar mensajes a los usuarios dentro de un grupo. Todas estas son características deseables en una aplicación como esta. Agregar todas estas características requeriría diseñar interfaces o replicar las de zoom, probablemente tomaría un mes de trabajo a alguien que conozca las tecnologías utilizadas.
3. Hacer que la aplicación escale: Las tecnologías usadas en el servidor de Easy Flip le permiten escalar, pero para que esto resulte bien sería bueno en primer lugar separar la lógica de Easy Flip y Easy Poll en aplicaciones separadas. Luego de eso habría que cambiar el sistema de almacenamiento de salas por una sistema de base de datos en memoria como Redis. Esto le daría la facultad al servidor de ser instalado en un cluster, para admitir una mayor cantidad de usuarios concurrentes. Hacer esto debería tomar no más de un mes de trabajo, sin contar con el tiempo necesario para hacer pruebas. Además de hacer los cambios en la aplicación habría que evaluar dónde alojarla.
4. Guardar las preguntas de forma más ordenada: En Easy Poll se pueden agregar tags a las preguntas según tema, idioma, etc y crear un buscador. Además de esto agregar la opción para tomar preguntas de alternativas de este buscador para usarlas en Easy Flip. Con esto implementado se puede ahorrar mucho tiempo de preparación de clases a los profesores, o salvarlos de imprevistos si necesitan material para su clase a última hora. Para esto habría que cambiar levemente el modelo de datos para añadir estos atributos a las preguntas y además diseñar interfaces, debiera tomar aproximadamente tres semanas de trabajo implementar esta característica.
5. Generar grupos de nuevas formas: Antes de empezar a desarrollar la aplicación, sólo se esperaba implementar la opción para maximizar la diversidad de respuestas dentro de

cada grupo. La idea de juntar a quienes respondían igual surgió de forma posterior y se decidió implementar, ya que permite al profesor separar a los alumnos según alguna preferencia de estos mismos. Por ejemplo si hay un grupo de alumnos que prefiere trabajar con la cámara apagada y otro que prefiere hacerlo con la cámara encendida, se les puede juntar según sus preferencias. Esto sería un trabajo de investigación e innovación, por lo que el tiempo necesario para lograrlo es variable, se estima que en un semestre se podrían obtener resultados.

Analizar y explorar otras opciones puede ser beneficioso para el desarrollo de la aplicación y puede ayudar a innovar en la forma de enseñar que existe hoy en día.

Bibliografía

[1] Adobe: *Adobe Buying Guide*. https://www.adobe.com/products/adobeconnect/buying-guide.html, (accedido el 10 de mayo de 2020).

[2] Docker Inc.: *Docker*. https://docs.docker.com/get-started/overview/, (accedido el 17 de abril de 2021).

[3] Ecma International: *JSON*. https://www.json.org/json-en.html, (accedido el 20 de abril de 2021).

[4] Facebook Inc.: *React.js*. https://reactjs.org/, (accedido el 20 de abril de 2021).

[5] Fette, I. y A. Melnikov: *The WebSocket Protocol*. RFC 6455, RFC Editor, December 2011. http://www.rfc-editor.org/rfc/rfc6455.txt, http://www.rfc-editor.org/rfc/rfc6455.txt.

[6] Fielding, Roy T., James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach y Tim Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616, RFC Editor, June 1999. http://www.rfc-editor.org/rfc/rfc2616.txt, http://www.rfc-editor.org/rfc/rfc2616.txt.

[7] Google: *Google Meet*. https://meet.google.com/, (accedido el 10 de mayo de 2020).

[8] Jitsi: *Jitsi Homepage*. https://jitsi.org, (accedido el 10 de mayo de 2020).

[9] Jitsi Meet: *Jitsi Meet Homepage*. https://meet.jit.si/, (accedido el 10 de mayo de 2020).

[10] Loreto, S., P. Saint-Andre, S. Salsano y G. Wilkins: *Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP*. RFC 6202, RFC Editor, April 2011. http://www.rfc-editor.org/rfc/rfc6202.txt, http://www.rfc-editor.org/rfc/rfc6202.txt.

[11] Mazur, E.: *Peer Instruction: A User's Manual*. Series in Educational Innovation. Prentice Hall, 1997.

[12] Mazur, Eric: *Mazur's tweet on his webinar*. https://twitter.com/eric_mazur/status/1257828711782658048, (accedido el 10 de mayo de 2020).

[13] Mazur, Eric: *Farewell, Lecture?*, 2009. https://science.sciencemag.org/content/323/5910/50.full.

[14] Mazur, Eric: *Webinar, How to flip your class online when the world is flipping out*, abril 2020. https://www.youtube.com/watch?v=JK-vv2NuWvA, (accedido el 26 de abril de 2021).

[15] MongoDB, Inc.: *What is MongoDB?* https://www.mongodb.com/what-is-mongodb, (accedido el 19 de abril de 2021).

[16] OpenJS Foundation: *JSON*. https://expressjs.com/, (accedido el 20 de abril de 2021).

[17] PollEverywhere: *PollEverywhere Homepage*. https://www.polleverywhere.com/, (accedido el 10 de mayo de 2020).

[18] Redislabs: *Redis*. https://redis.io/, (accedido el 9 de abril de 2021).

[19] socket.io: *Socket.io*. https://socket.io/docs/v4/, (accedido el 11 de abril de 2021).

[20] SurveyMonkey: *SurveyMonkey*. https://www.surveymonkey.com/mp/audience/features/, (accedido el 10 de mayo de 2020).

[21] Wikipedia: *Flipped classroom article*. https://en.wikipedia.org/wiki/Flipped_classroom, (accedido el 10 de mayo de 2020).

[22] World Wide Web Consortium: *Iframe*. https://www.w3.org/TR/2010/WD-html5-20100624/the-iframe-element.html, (accedido el 12 de abril de 2021).

[23] Zoom: *Zoom Homepage*. https://zoom.us/, (accedido el 10 de mayo de 2020).

Apéndice A. Código de algoritmos para generación de grupos

A continuación se muestra todo el código de los algoritmos utilizados para generar grupos de discusión.

```
1  function createRooms(size, users, breakoutOption, smartBreakoutOption, poll) {
2      if (breakoutOption === 'byNumber') {
3          return createRoomsByQuantity(size, users, smartBreakoutOption, poll);
4      }
5      return createRoomsBySize(size, users, smartBreakoutOption, poll);
6  }
7  /**
8   * Creates breakout rooms with the selected size, and sends people to them.
9   * @param {int} groupSize the size of the groups to be created
10  * @param {Set<String>} userIDs the set containing all the jitsi participants
11  *
12  */
13 function createRoomsBySize(groupSize, userIDs, smartBreakoutOption, poll) {
14     return createRoomsByQuantity(Math.floor(userIDs.length / groupSize),
15     userIDs, smartBreakoutOption, poll);

17     //(if fillup false)
18     //return createRoomsByQuantity(Math.ceil(users.length / groupSize), users);
19 }

21 /**
22  * Creates a number of groups depending on the numberofgroups property
23  * @param {int} numberOfGroups quantity of groups to be created
24  * @param {Array<String>} userIDs array containing all user ids
25  * @returns {Array}
26  */
27 function createRoomsByQuantity(numberOfGroups, userIDs, smartBreakoutOption, poll) {
28     if (smartBreakoutOption === 'sameAnswers') {
29         return sameAnswerDistribution(numberOfGroups, userIDs, poll);
30     }
31     else if (smartBreakoutOption === 'differentAnswers') {
32         return differentAnswersDistribution(numberOfGroups, userIDs, poll);
33     }
34     else if (smartBreakoutOption === 'random') {
35         return randomDistribution(numberOfGroups, userIDs, poll);
36     }
37 }

39 /**
40  * This function distributes the users in a random way
41  * @param {int} numberOfGroups approximately how many groups will be created
42  * @param {Array<String>} userIDs array containing all user ids
43  * @param {Poll} poll not used for this function
44  * @returns {Array<Array<String>>} An array that shows how the groups will be distributed
45  */
46 function randomDistribution(numberOfGroups, userIDs, poll) {
47     if (numberOfGroups === 0) numberOfGroups = 1;
48     let groupCounter = 0;
49     shuffle(userIDs);
50     let distribution = Array.from(Array(numberOfGroups), () => []);
51     for (let user of userIDs) {
52         distribution[groupCounter].push(user);
53         console.log('pushing user ' + user + 'to group ' + groupCounter);
54         groupCounter++;
55         groupCounter = groupCounter %numberOfGroups;
56     }

58     showDistribution(distribution, poll);
59     return distribution;
60 }

62 /**
63  * This function puts users who had the same answer on the poll together
64  * @param {int} numberOfGroups approximately how many groups will be created
65  * @param {Array<String>} userIDs array containing all user ids
66  * @param {Poll} poll the poll used for distributing the users
```

```
67   * @returns {Array<Array<String>>} An array that shows how the groups will be distributed
68   */
69  function sameAnswerDistribution(numberOfGroups, userIDs, poll) {
70      if (numberOfGroups === 0) numberOfGroups = 1;
71      const groupSize = Math.max(Math.floor(userIDs.length / numberOfGroups),1);
72      const answers = poll.votes;
73      let distribution = [];
74      let buckets = separateByAnswers(userIDs, poll);
75      let previousNumberOfGroups = 0;
76      for (let bucket of buckets) {
77          let groupCounter = previousNumberOfGroups;
78          const numberOfGroupsForThisBucket = Math.max(Math.floor(bucket.length / groupSize),1);
79          for (let i = previousNumberOfGroups; i < (previousNumberOfGroups + numberOfGroupsForThisBucket); i++) {
80              distribution[i] = [];
81          }
82          while (bucket.length > 0) {
83              distribution[groupCounter].push(bucket.pop());
84              groupCounter++;
85              groupCounter = ((groupCounter - previousNumberOfGroups) %numberOfGroupsForThisBucket) +
86              previousNumberOfGroups;
87          }
88          previousNumberOfGroups += numberOfGroupsForThisBucket;

90      }
91      showDistribution(distribution, poll);
92      return distribution;
93  }

95  /**
96   * This function tries to put users who had different answers together
97   * @param {int} numberOfGroups approximately how many groups will be created
98   * @param {Array<String>} userIDs array containing all user ids
99   * @param {Poll} poll the poll used for distributing the users
100  * @returns {Array<Array<String>>} An array that shows how the groups will be distributed
101  */
102 function differentAnswersDistribution(numberOfGroups, userIDs, poll) {
103     if (numberOfGroups === 0) numberOfGroups = 1;

105     let distribution = Array.from(Array(numberOfGroups), () => []);
106     let buckets = separateByAnswers(userIDs, poll);
107     let groupCounter = 0;

109     for (let bucket of buckets) {
110         while (bucket.length > 0) {
111             distribution[groupCounter].push(bucket.pop())
112             groupCounter++;
113             groupCounter = groupCounter%numberOfGroups;
114         }
115     }
116     showDistribution(distribution, poll);
117     return distribution;

119 }

121 /**
122  * Utility function that puts users who had the same answer in the poll in the same array
123  * @param {*} userIDs the users to separate
124  * @param {*} poll the poll used for separating users
125  * @returns {Array<Array<String>>} The users grouped together based on their answers,
126  * users who didn't answer are put in the last array
127  */
128 function separateByAnswers(userIDs, poll) {
129     const answers = poll.votes;
130     const numberOfBuckets = poll.alternatives.length + 1;
131     let buckets = Array.from(Array(numberOfBuckets), () => []);
132     //we separate participants in buckets, according to their answers
133     for (const userID of userIDs) {
134         if (answers.get(userID)!==undefined) {
135             buckets[answers.get(userID)].push(userID);
136         }
137         else {
138             buckets[numberOfBuckets - 1].push(userID);
139         }
140     }
141     console.log("the buckets are :", buckets);
142     return buckets;
143 }


146 /**
147  * Sends a message to each user, telling them which room to go to
148  * @param {socket} socket the socket sending the message, normally it's the owner of the room
149  *(maybe verify this in the future for security)
150  * @param {string} roomName the name of the room that's being split
151  * @param {Array<Array<string>>} distribution Each sub-array contains all the user IDs of a breakout room
152  * @param {RoomStore} roomStore the room store, used to save the breakout rooms in memory
153  * @param {string} pollId the pollId, used to persist the poll to the breakout rooms
154  */
155 function sendToBreakout(socket, roomName, distribution, roomStore, pollId) {
156     let roomCounter = 0;
157     const breakoutRoomNames = [];
158     for (let group of distribution) {
159         if(group.length>0){
160             roomCounter++;
161             const breakoutRoomName = roomName + 'easyFlipRoom' + roomCounter;
162             breakoutRoomNames.push(breakoutRoomName);
```

```
163                 const newRoom = roomStore.newRoom(breakoutRoomName, undefined, roomName, pollId);
164                 for (let userID of group) {
165                     socket.to(userID).emit('notifyBreakout', { breakoutRoomName:
166                     breakoutRoomName, originalRoomName: roomName });
167                     console.log('sending user ' + userID + ' to breakout room ' + breakoutRoomName);
168                 }
169             }
170         }

172         roomStore.getRoom(roomName).breakoutRooms = breakoutRoomNames;
173         socket.emit('roomsCreated', breakoutRoomNames);
174     }

176     //shuffles an array, taken from stackoverflow
177     function shuffle(array) {
178         var currentIndex = array.length, temporaryValue, randomIndex;

180         // While there remain elements to shuffle...
181         while (0 !== currentIndex) {

183             // Pick a remaining element...
184             randomIndex = Math.floor(Math.random() * currentIndex);
185             currentIndex -= 1;

187             // And swap it with the current element.
188             temporaryValue = array[currentIndex];
189             array[currentIndex] = array[randomIndex];
190             array[randomIndex] = temporaryValue;
191         }

193         return array;
194     }
```