Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Grado en Ingeniería Informática

Trabajo Fin de Grado

Danubit: Desarollo de un sistema software web para la administración de asociaciones de la ETSIINF.

Autor: Borja Martinena Cepa
Tutor 1: Jose María Barambones Ramírez
Tutor 2: Ricardo Imbert Paredes

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Grado en Ingeniería Informática

Título: Danubit:
Desarollo de un sistema software web para la administración de asociaciones de la ETSIINF.

Junio 2024

Autor: Borja Martinena Cepa
Tutores: Jose María Barambones Ramírez
Ricardo Imbert Paredes
Dto. de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

La siguiente memoria de Trabajo de Fin de Grado contiene el diseño, desarrollo, detalles de implementación y evaluación de un software web para la gestión de asociaciones de alumnos de aquí en adelante llamado "Danubit".

Danubit surge de la necesidad de las asociaciones de la ETSIINF de una herramienta que facilite y centralice muchas de las tareas de gestión necesarias para existir como tal. De esta manera, además, podría facilitarse la comunicación con el organismo superior que se encarga de aprobar las actividades de las asociaciones, en este caso Subdirección de Alumnos y Rectorado de la UPM.

Danubit es un servicio web con la forma de una API Rest capaz de gestionar asociaciones y sus actividades, miembros, documentos, inventario y juntas directivas. Además, opcionalmente, Danubit permite que las asociaciones tengan un "gestor", esto es, un usuario capaz de aprovar actividades y materiales publicitarios.

El servicio está implementado en Rust con ayuda de una librería para crear APIs Rest llamada `poem-openapi`, centrada en seguridad de tipos y documentación generada a partir del código, para crear servicios mantenibles y con una alta tolerancia a fallos, además de eficientes.

El Trabajo de Fin de Grado incluye además un pequeño frontend implementado en React que se conecta con parte de la funcionalidad que ofrece el backend y puede servir como punto de partida que extender en futuros trabajos.

Aunque el frontend no ha podido completarse hasta la paridad de características con el backend, los resultados de la evaluación de usabilidad muestran que es un gran comienzo con potencial de crecimiento. En general, el proyecto ha tenido éxito en comenzar una plataforma open-source, y llevada por estudiantes que ayude a las asociaciones de alumnos a existir y manejarse.

Palabras Clave: Asociaciones, ETSIINF, API Rest, Rust, Poem, OpenAPI, React, Seguridad de tipos

Abstract

The following is the report for a Bachelors Thesis about Danubit: A web service for student association management. It details the service's design, development, implementation, and evaluation.

The idea for Danubit is born out of the necessity for a tool that centralises and automates, to a certain degree, many of the routine management tasks necessary for the normal operation of student associations at ETSIINF.

Danubit is a web service mostly in the form of a Restful API capable of storing information about student associations and their activities, members, documents, materials and management boards. Additionally, Danubit can keep track of "manager usersçapable of approving or rejecting an associations activities and public media.

The service is implemented in Rust with the help of `poem-openapi`, a library for building Restful APIs with a focus on type safety and documentation, which it generates automatically from Rust's rich type system.

The work also includes the details of a small React-powered frontend that uses the API to create a web portal for student association board members to easily manage activities and members. The frontend's functionality, however, is still limited in regards to the backend.

Although the frontend could not be finished to it's full potential, it's usability evaluation shows it's in the right track and has major growing opportunities. There are lots of additional features that the platform could have and people willing to develop them. The project has been a success in starting an open-source student-driven portal for asociations to exist and manage themselves in.

Keywords: Associations, ETSIINF, Rest API, Rust, Poem, OpenAPI, React, Type Safety

Índice general

Índice de Figuras

Índice de Tablas

Índice de Listings

Capítulo 1

Introducción

En este capítulo se expone la motivación detrás de la creación de Danubit, a continuación se explica el contexto en el que se ha desarrollado, la planificación desde un principio del proyecto y sus cambios, y algunos de los acontecimientos relevantes que han ocurrido durante su desarrollo. Además, se listan los objetivos principales y secundarios del proyecto, indicándose cuáles han sido alcanzados.

1.1. Motivación del proyecto

En la Escuela Técnica Superior de Ingenieros Informáticos, existen actualmente 6 asociaciones de alumnos con diversos intereses y formas de organización. Algunos de los principales problemas de cara a su continuada existencia son: la falta de comunicación con el alumnado en general, en la forma de publicidad efectiva para sus actividades y métodos de captación de nuevos miembros interesados; y la falta de comunicación con Subdirección de Alumnos que se ve empeorada por cambios de juntas, perdidas de documentos y cambios en los procedimientos oficiales para llevar a cabo tareas esenciales para la operación de las asociaciones.

Estos problemas, agravados por la pandemia del Covid-19 en 2020 y la consecuente suspensión temporal de clases presenciales, han sido la causa de que algunas asociaciones hayan estado al borde de la desaparición, y otras hayan tenido que unirse para sobrevivir.

En torno al final del curso académico 2022-2023 comenzó a circular la idea de crear una plataforma para facilitar y centralizar muchas de estas tareas en un único portal para asociaciones, que a su vez podía servir como web donde alojar la publicidad que se producía para las diversas actividades.

1.2. Objetivos

El objetivo principal de este proyecto es crear una base sólida sobre la que continuar creando esta plataforma. Dadas las condiciones en las que estará alojado el software y la dificultad para encontrar gente que lo mantenga, era importante crear un software que fuera altamente tolerante a fallos y relativamente eficiente. El resto de objetivos secundarios tienen que ver con las diversas funcionalidades que se propusieron en

su momento para la plataforma. Estas funcionalidades se recogieron y priorizaron en un *World Café* que está detallado en la sección de Desarrollo (3).

- Implementar la API básica de un servicio para manejar asociaciones.
- Crear un prototipo de usabilidad para un posible frontend de la plataforma.
- Implementar el comienzo de un frontend para la plataforma.

La totalidad de las funcionalidades que se propusieron fueron las siguientes, divididas según si finalmente han sido o no implementas:

Implementadas:

- Frontpage para las asociaciones.
- Manejo de juntas.
- Manejo de miembros.
- Manejo de actividades.
- Guardado y préstamo de materiales.

Parcialmente implementadas:

- Reserva de espacios.
- Aprobación de actividades.
- Guardado y comunicación de documentos.

No implementadas:

- Sellado de cartelería.
- Impresión de cartelería.
- Uso de las pantallas de la escuela.
- Pedir arreglos a la sala de asociaciones.
- Chat con subdirección.
- Newsletter de actividades.
- Log de reuniones.
- Avisos y notificaciones.

1.3. Planificación

Inicialmente, el trabajo se enmarcó para realizarse durante el primer cuatrimestre del curso académico 2023-2024. Sin embargo, por varias circunstancias imprevistas, su plazo tuvo que alargarse para así ser presentado en la convocatoria extraordinaria de ese mismo curso. A continuación se muestra la evolución de la planificación del proyecto en tres diagramas de Gantt, el primero corresponde al momento de inicio del trabajo, el segundo a mitad del primer semestre, y el último indica cuál ha sido la distribución final del tiempo dedicado al trabajo.

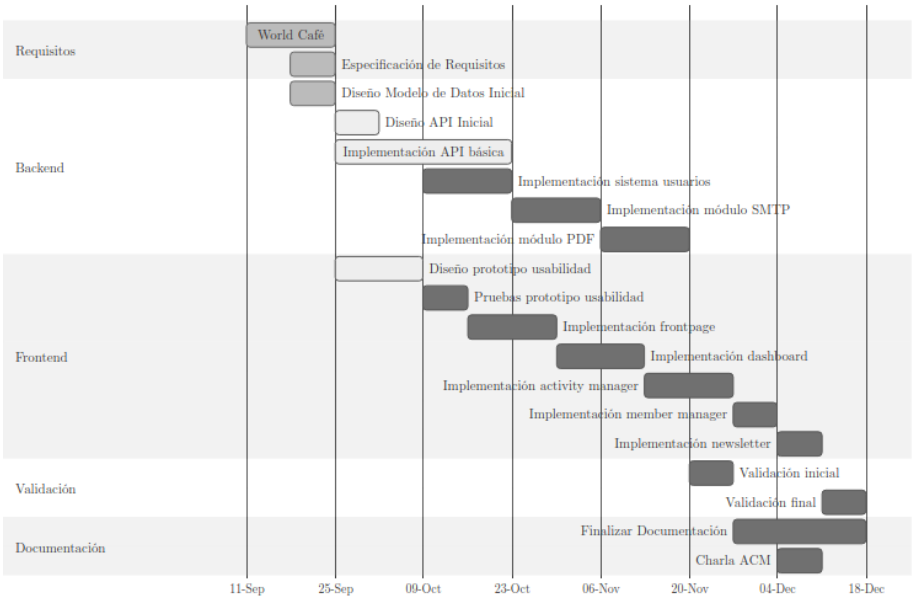

Figura 1.1: Diagrama de Gantt inicial.

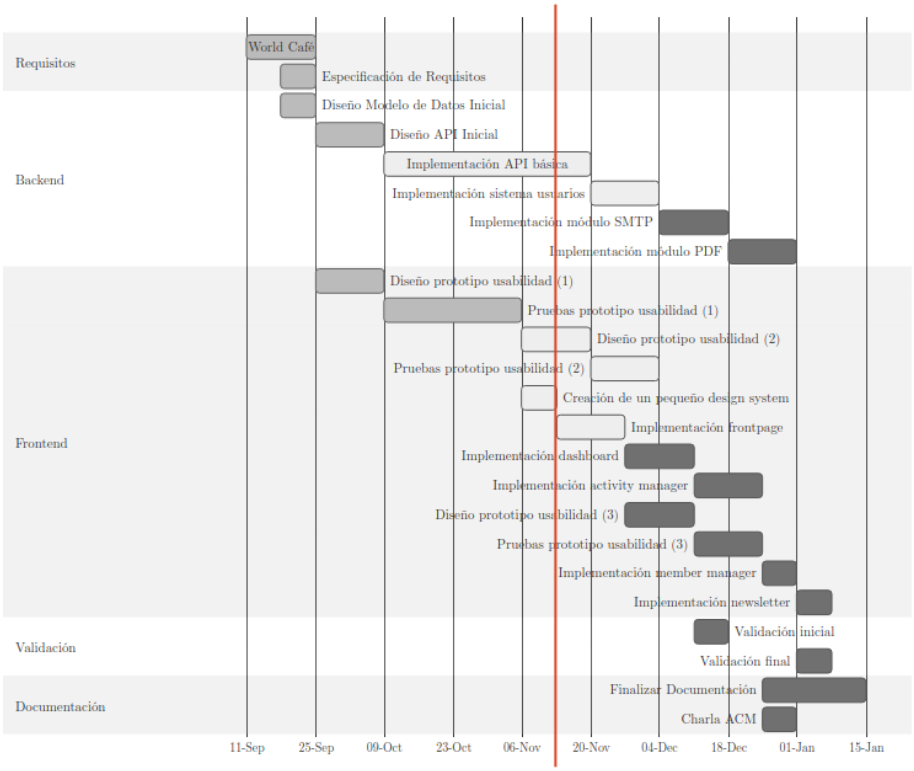

Figura 1.2: Diagrama de Gantt intermedio.

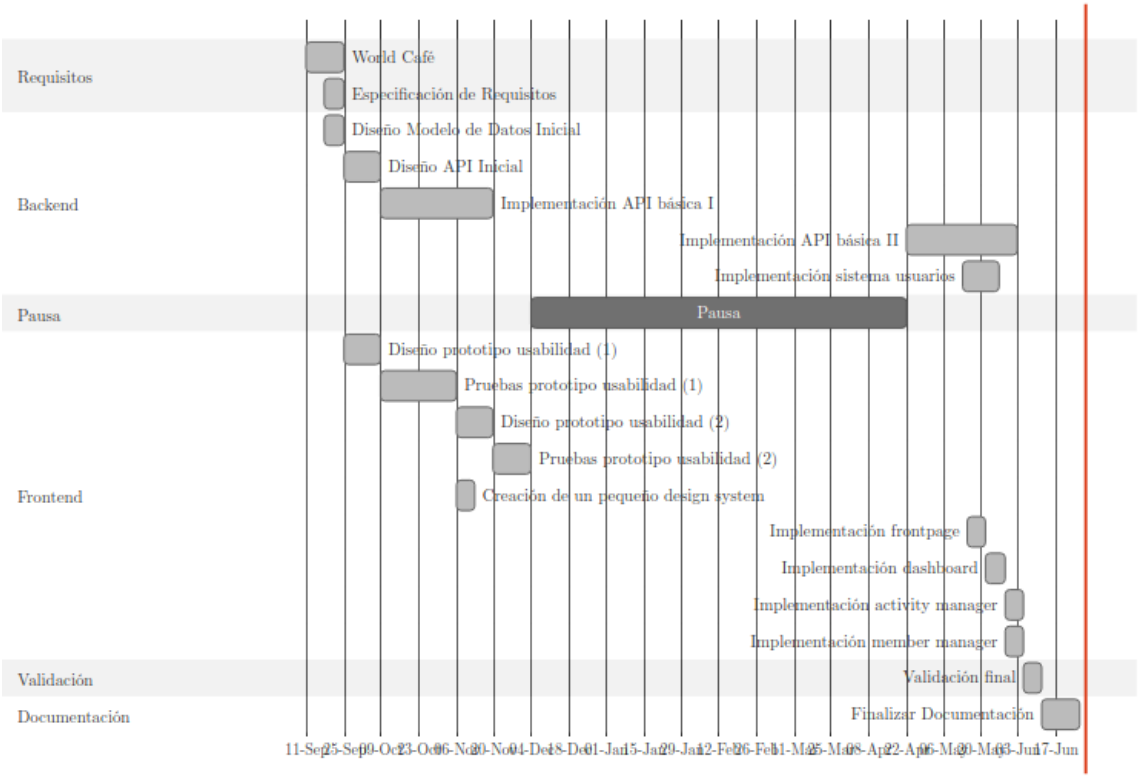

Figura 1.3: Diagrama de Gantt final. (Convocatoria extraordinaria)

1.4. Estructura del Documento

La presente memoria del trabajo se estructura de la siguiente manera: Primero, se detalla el estado de los procedimientos para asociaciones en la escuela tal y como existían al comienzo de este trabajo. Además, se muestran los productos alternativos que existen en el mercado que podrían cumplir una función similar y se mencionan algunas actualizaciones que ha habido a algunos de los procesos internos en el transcurso de este trabajo.

A continuación, se describe el desarrollo de la plataforma, que incluye: la recogida y especificación de requisitos; el diseño software de la plataforma; información sobre las tecnologías usadas; detalles de implementación, tanto del backend en Rust como del frontend en React; y una descripción de los prototipos de usabilidad creados para el frontend.

En la sección de evaluación se recogen los resultados de la evaluación de los prototipos de usabilidad del frontend, así como la evaluación del frontend final. Además, se incluyen las conclusiones finales del autor tras el desarrollo trabajo.

Por último, se describen la formas en las que el trabajo puede ser continuado, muchas de las cuales provienen de la recogida de requisitos, otras simplemente aspectos de la plataforma que no han podido desarrollarse como parte de este trabajo por las restricciones de tiempo.

Se incluyen además 3 anexos: la especificación completa de la API de Danubit, la API del pequeño sistema de autenticación, y los datos en bruto de las evaluaciones de usabilidad.

Capítulo 2

Trabajo relacionado y Estado del Arte

Dos temas son de vital importancia de cara al desarrollo de este proyecto: El estado de los procedimientos actuales en la ETSIINF que la plataforma pretende cambiar, y las alternativas para administración de asociaciones que existen en el mercado para justificar que esta solución es necesaria y no hay una alternativa viable que cumpla con las funcionalidades esperadas.

2.1. Procedimientos para las asociaciones de la ETSIINF

Este proyecto trata de facilitar muchos de los procesos necesarios para administrar una asociación de alumnos a base de centralizarlos y automatizarlos parcialmente. Para ello, era necesario tener un conocimiento cercano de la forma en la que las asociaciones llevan a cabo estas tareas sin el uso de la herramienta.

Manejo de miembros:

Cada asociación en la escuela tiene su propio sistema para llevar cuentas de quién es y quién no es miembro de la asociación. Asociaciones como ASCFI, donde la membresía ha de renovarse anualmente, tienen menos problemas con cambios de juntas y de sistemas, ya que cada junta solo ha de encargarse de los miembros presentes. Otras asociaciones como ACM, donde la membresía se considera permanente hasta terminar la carrera, tienen más problemas siguiendo qué alumnos son parte de la asociación y cuáles ya se han graduado. Además, tanto para unas como para las otras, se vuelve difícil tener un historial coherente del número de miembros que se han tenido cada año, de cara a crear reportes de actividad para Subdirección de Alumnos.

Comunicación con Subdirección de Alumnos y organización de actividades:

Todos los procedimientos que tengan algo que ver con la escuela han de pasar primero por Subdirección de Alumnos. Esto incluye, la organización de actividades, sellado de carteles para poner por la escuela, cambios de junta y peticiones de material o de arreglos a la sala. Además, Subdirección de Alumnos requiere periódicamente de ciertos documentos que certifiquen que las asociaciones siguen activas y tienen miembros.

En los últimos años, se ha dado varias veces la situación de que alguna de las asociaciones no ha sido capaz de proporcionar uno de estos documentos por haberse perdido en los procedimientos de algún cambio de junta, o por simplemente existir en la cuenta de Google privada de alguno de los miembros menos activos de la junta directiva.

Además, algunos trámites contienen una cantidad excesiva de pasos. Hasta el último cambio, para imprimir carteles que las asociaciones pudieran pegar por la facultad se tenían que seguir los siguientes pasos: Primero una primera copia del cartel tenía que llevarse presencialmente a subdirección de alumnos para que fuera sellada. Allí, además, había que pedir una orden de trabajo que informaba a Publicaciones de que la asociación tenía derecho a imprimir un número de copias de ese cartel sin coste, en blanco y negro. Una vez impresos con el sello, se podían pegar los carteles. Este aparente proceso de 3 partes muchas veces era imposible de completar en varios días: Publicaciones no se encuentra siempre operativo, o se niega a fiar la primera copia a la junta de la asociación, Subdirección de Alumnos no tiene constancia de que tengan que dar una orden de trabajo o no tienen el documento impreso necesario para hacerlo. Con la última reforma de procedimientos, esta tarea, al igual que muchas otras se han simplificado para que simplemente requieran de un correo, y el *scope* de este proyecto cambió para reflejarlo, poniendo más prioridad en ayudar a las asociaciones a administrarse internamente que a facilitar la comunicación con Subdirección de Alumnos.

2.2. Software de administración de asociaciones

Existen muy pocos productos software centrados en la administración de asociaciones de alumnos, y las funcionalidades de aquellos que sí existen son muy parcelares, por lo que sería necesario usar varios para cubrir todas las necesidades de una asociación, perdiendo así el carácter centralizador que Danubit quiere aportar.

Además, la mayoría de productos para administración de asociaciones son comerciales y de pago, un privilegio que la mayoría de asociaciones de la escuela no se puede permitir. No solo eso, sino que las actividades en las que se centran son las menos relevantes en este contexto: manejo de facturas, recaudaciones de fondos, etc.

Cabe mencionar algunas de las soluciones que existen en el mercado que sí coinciden en funcionalidad y tratan de resolver algunos de los problemas que han motivado la creación de Danubit.

Admidio [1] es una plataforma para el manejo de membresías a una asociación. Es de código abierto y gratuita, y está escrita en PHP. Además, tienen un sistema de plugins que permite aumentar su funcionalidad.

WildApricot [2] es otra alternativa con una versión gratis que facilita la administración de miembros y además ofrece la posibilidad de crear una página pública para la asociación. Solía tener una versión gratuita, pero actualmente es software de pago.

Aunque la implantación de alguna de estas alternativas como medio oficial con el cual tienen que manejarse las asociaciones podría, sin duda, resolver muchos problemas, ninguna de ellas está centrada específicamente en asociaciones de alumnos, y sus necesidades de comunicación con el organismo educativo en el que existen, en nuestro caso, la ETSIINF. Danubit tiene como objetivo ser una plataforma *open*

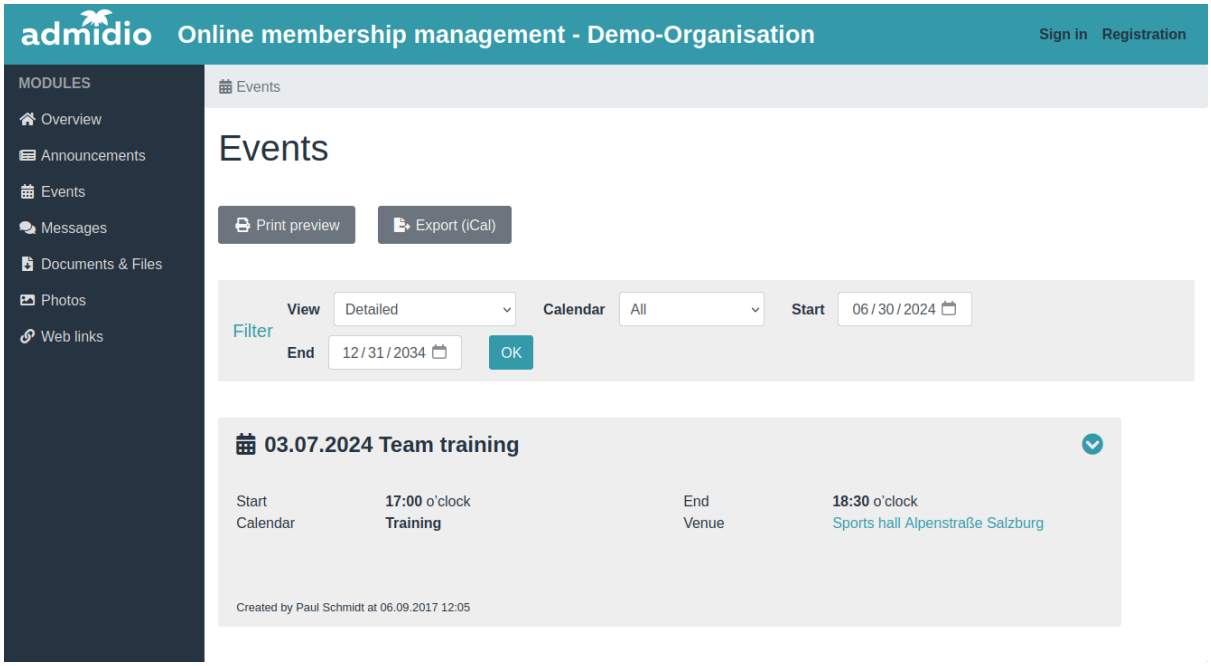

Figura 2.1: Captura de pantalla de la Demo gratuita de Amidio.

source llevada por alumnos y extensible por los mismos para ser capaz de resolver muchos tipos distintos de problemas.

Capítulo 3

Diseño y Desarrollo

3.1. Especificación de Requisitos

Dada la amplia cantidad de funcionalidades que una plataforma para el manejo de asociaciones podría tener, uno de los primeros pasos necesarios para llevar a cabo este trabajo era recoger información sobre las necesidades y prioridades de las asociaciones. Para esto, se llevó a cabo una dinámica de grupo llamada "World Café"[3] en la que participaron miembros de las juntas de ACM UPM, ASCFI y Rugby, así como Subdirección de Alumnos. El resto de asociaciones fueron avisados, pero no pudieron acudir el día de la dinámica.

De esta actividad se sacaron las siguientes funcionalidades, ordenadas de más a menos importantes, con su número de votos:

1. Información pública sobre las asociaciones. (Indispensable):
   La plataforma tendría que servir como portal para que nuevos alumnos de la facultad descubriesen las asociaciones, y ofrecer información sobre las mismas.
2. Ayuda para la organización de actividades y eventos (8)
   La plataforma tendría que ofrecer una interfaz única para la creación de actividades y eventos que permita a alumnos interesados apuntarse a dichas actividades. Además, debería tener en cuenta las siguientes opciones:
   - Actividades acreditables.
   - Guardar los carteles y promoción.
   - Reserva de espacios para las actividades.
   - Comunicación con la facultad para promoción.
   - Creación de actas de las actividades una vez terminadas.
3. Registro de miembros (7)
   La plataforma debería llevar cuentas de los miembros de una asociación y ser el método principal por el que los alumnos se apuntan a una asociación.
4. Newsletter / Tablón de anuncios (7)
   La plataforma debería incluir una forma de anunciar nuevas actividades a los miembros de las asociaciones.

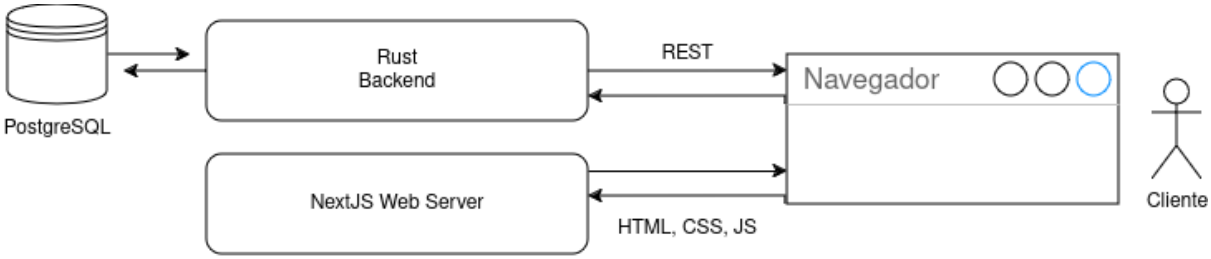

Figura 3.1: Diagrama de la arquitectura de Danubit.

5. Avisos entre subdirección y asociaciones (7)
   La plataforma podría servir como medio de comunicación oficial entre las asociaciones y Subdirección de Alumnos.

6. Manejo de redes sociales (6)
   La plataforma podría hacer interfaz con las distintas redes sociales y facilitar la promoción de actividades automáticamente según se crean.

7. Almacenamiento de documentos (4)
   La plataforma debería servir de almacenamiento digital de documentos de las asociaciones.

8. Manejo de normas de convivencia (3)
   La plataforma podría servir como portal público para las normas de convivencia de la Sala de Asociaciones.

9. Foro de alumnos (0)
   La plataforma podría servir además como un foro para los alumnos de la facultad.

10. Plataforma para torneos (0)
    La plataforma podría tener pequeños *applets* como una plataforma para torneos, para facilitar los formatos de actividad más comunes.

3.2. Backend

El backend de Danubit se compone de 2 partes: El servidor web en Rust, y una base de datos relacional PostgreSQL. El servidor web se compone de 2 APISs: la general y la de autenticación. Todo el código se puede encontrar en github: https://github.com/aafrecct/danub

3.2.1. Tecnologías

El backend de Danubit está desarrollado con la ayuda de varias tecnologías, que se detallan aquí para facilitar la comprensión de las siguientes partes.

PostgreSQL [4] El proyecto utiliza una base de datos PostgreSQL, por su comodidad de uso y naturaleza open source.

Rust [5] Danubit está escrito en Rust. El gran sistema de tipos de Rust permite que todo el modelado de datos ocurra dentro del lenguaje y las garantías que ofrece

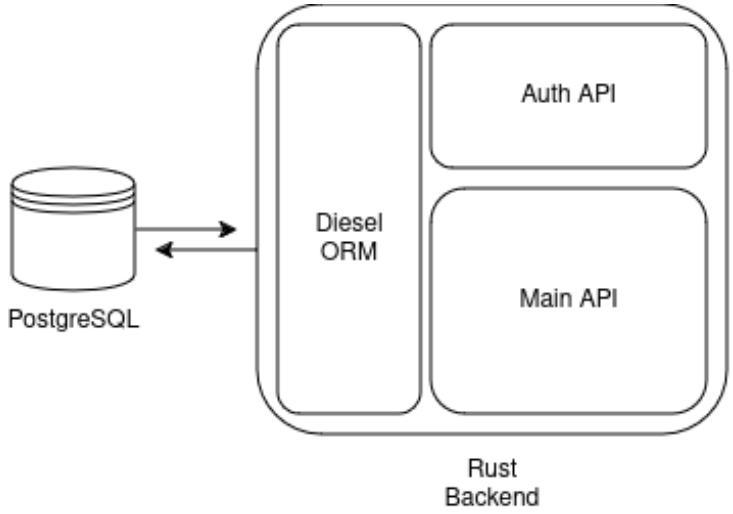

Figura 3.2: Diagrama de la estructura del backend de Danubit.

su sistema de manejo de errores son de gran ayuda de cara a crear un sistema robusto y resistente, con gran tolerancia a fallos.

Diesel ORM [6] Diesel es una librería de Rust que ofrece una abstracción sobre la base de datos que permite que el proyecto no tenga que pensar tanto en SQL, la interacción con la base de datos se hace a través de modelos de Rust y es comprobada en tiempo de compilación.

Poem [7] Poem es una librería de Rust que facilita la creación de servicios web. En particular, `poem-openapi` ofrece una manera sencilla de crear APIs Rest que se conforman a la especificación de OpenAPI 3.0 [8]. Además, gracias al estricto sistema de tipos de Rust, `poem-openapi` es capaz de generar la documentación de la API a partir del código, asegurando así que nunca está desactualizada.

3.2.2. Diseño

3.2.2.1. Base de Datos

La base de datos de Danubit se crea ejecutando un número determinado de migraciones, en forma de archivos `.sql`, en orden. Cada migración se encarga de crear la tabla y relaciones de un tipo de datos y todas son reversibles. De esta manera, si se quisieran añadir nuevas funcionalidades a la plataforma que requiriesen de nuevas tablas, el proceso sería sencillo y solo involucraría la creación de 2 nuevos archivos: `up.sql`, que crearía la tabla; y `down.sql` para eliminarla.

Estas migraciones surgen naturalmente con el desarrollo progresivo de las distintas funcionalidades del software, y muestran, por tanto, su avance. A continuación se incluyen un diagrama entidad-relación de la base de datos y descripciones de cada una de las tablas que se crean.

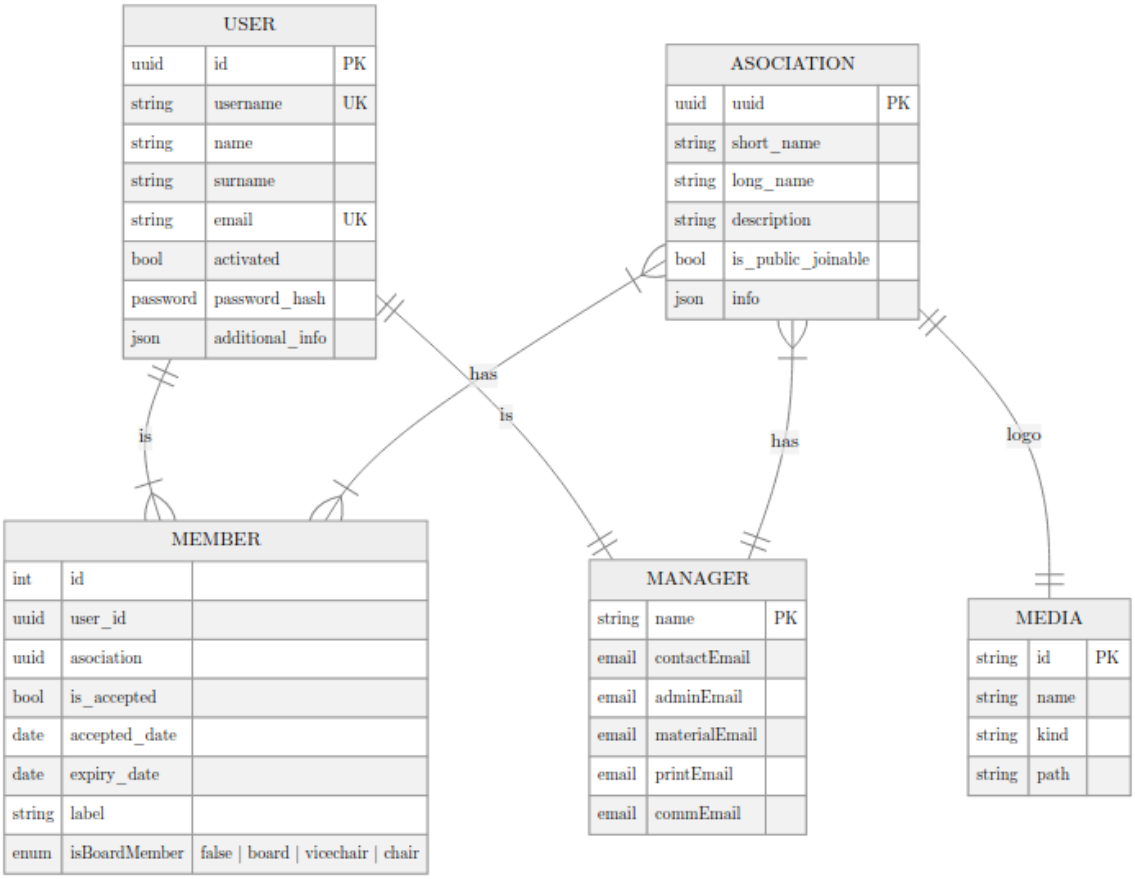

Figura 3.3: Diagrama Entidad-Relación de la base de datos de Danubit (Parte 1).

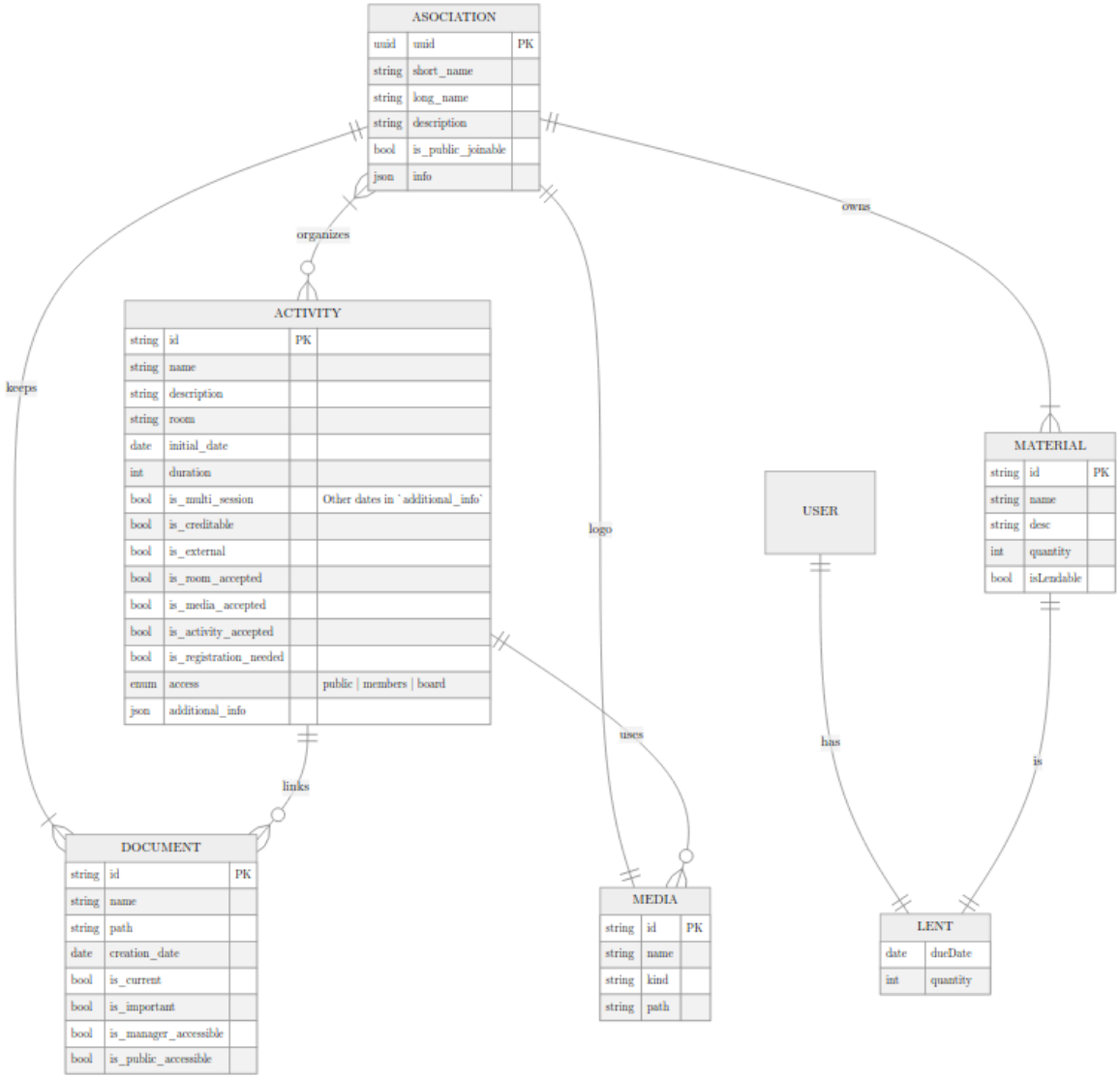

Figura 3.4: Diagrama Entidad-Relación de la base de datos de Danubit (Parte 2).

Users:

uuid	id	PK
string	username	UK
string	name	
string	surname	
string	email	UK
bool	activated	
password	password_hash	
json	additional_info	

Cuadro 3.1: Tabla para usuarios en la base de datos.

Asociations:

uuid	uuid	PK
string	short_name	
string	long_name	
string	description	
bool	is_public_joinable	
json	info	

Cuadro 3.2: Tabla para asociaciones en la base de datos.

Members:

int	id	
uuid	user_id	
uuid	asociation	
bool	is_accepted	
date	accepted_date	
date	expiry_date	
string	label	
enum	isBoardMember	false | board | vicechair | chair

Cuadro 3.3: Tabla para miembros en la base de datos.

Activities:

string	id	PK
string	name	
string	description	
string	room	
date	initial_date	
int	duration	
bool	is_multi_session	Other dates in `additional_info`
bool	is_creditable	
bool	is_external	
bool	is_room_accepted	
bool	is_media_accepted	
bool	is_activity_accepted	
bool	is_registration_needed	
enum	access	"public | members | board"
json	additional_info	

Cuadro 3.4: Tabla para actividades en la base de datos.

Documents:

string	id	PK
string	name	
string	path	
date	creation_date	
bool	is_current	
bool	is_important	
bool	is_manager_accessible	
bool	is_public_accessible	

Cuadro 3.5: Tabla para documentos en la base de datos.

Materials:

string	id	PK
string	name	
string	desc	
int	quantity	
bool	is_lendable	

Cuadro 3.6: Tabla para materiales en la base de datos.

Media:

string	id	PK
string	name	
string	kind	
string	path	

Cuadro 3.7: Tabla para multimedia en la base de datos.

Manager:

string	name	PK
email	contactEmail	
email	adminEmail	
email	materialEmail	
email	printEmail	
email	commEmail	

Cuadro 3.8: Tabla para *managers* en la base de datos.

3.2.2.2. API

La API de Danubit utiliza muchos de los mismos modelos de datos que usa la base de datos, pero no replica exactamente su esquema, por tema de permisos y de facilidad de acceso a los datos. La especificación completa de la API se encuentra en el Anexo

A. A continuación se explican algunas de las decisiones que se han tomado de cara al diseño de la API.

La gestión de miembros de una asociación no sigue un esquema CRUD tradicional sino que los usuarios hacen una petición `POST` al endpoint `/asociation/{id}/membershipRequests`. Esto registra una petición de membresía que luego puede ser aceptada o rechazada por la junta. Una vez la petición ha sido aceptada, el miembro se vuelve accesible desde el endpoint `/asociation/{id}/members`.

Además, algunos recursos, como los documentos o las actividades, tienen distintos endpoints para distintos niveles de permisos: los documentos públicos de una asociación, como lo pueden ser las bases, estatutos o normas de convivencia se pueden acceder sin autenticación desde el endpoint `/asociation/{id}/publicDocuments`. Para acceder a todos los documentos posibles se ha de acceder con un token de autorización al endpoint más general `/asociation/{id}/publicDocuments`.

3.2.3. Implementación

Danubit está escrito en Rust con la ayuda de varias librerías, a continuación se incluye un listing del archivo `cargo.toml` que incluye los metadatos del paquete y sus dependencias.

Listing 3.1: cargo.toml

```
[package]
name = "danubit"
version = "0.0.2"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.
    html

[dependencies]
time = { version = "^0.3", features = ["serde"] }
uuid = { version = "^1.5", features = ["serde"] }
dotenvy = "^0.15"
tracing = "^0.1"
tokio = { version = "1", features = ["macros", "rt-multi-thread"] }
serde = { version = "1.0", features = ["derive"] }
serde_json = { version = "1.0" }
diesel = { version = "2.1.0", features = [
    "postgres",
    "serde_json",
    "uuid",
    "time",
    "r2d2",
] }
diesel-derive-enum = { version = "2.1.0", features = ["postgres"] }
poem = "1.3"
poem-openapi = { version = "3.0", features = ["swagger-ui", "uuid", "time"] }
jwt = "0.16.0"
sha2 = "0.10.8"
hmac = "0.12.1"
argon2 = "0.5.3"
tracing-subscriber = "0.3.18"
```

Además de los ya explicados anteriormente, es apropiado mencionar `serde`, la librería principal de serialización y deserialización de Rust, y `tokio`, un *runtime* para programación asíncrona.

Como muchos frameworks de desarrollo web, poem se encarga de asociar cada endpoint de la API a una función asíncrona, que recibe como argumentos, los datos necesarios. La localización de los datos se indica a través del sistema de tipos de Rust, es decir, un dato del tipo `Path<Uuid>` indica que en la sección del *path* de la URL, hay que parsear un dato del tipo Uuid, `serde` se encarga de la deserialización.

Además, Rust no tiene un sistema de excepciones, los errores están integrados en el sistema de tipos. Una función que puede fallar, tiene que decirlo explícitamente devolviendo un tipo `Result`, que tiene que ser tratado para que el programa compile. En el ejemplo se puede ver como los errores que pueden ir surgiendo durante la ejecución de la función se transforman en errores HTTP para ser devueltos por `poem`.

Listing 3.2: Ejemplo de código de un endpoint

```
#[oai(
    path = "/asociations/:asociation_id/membershipRequests",
    method = "post",
    tag = "ApiTags::Members"
)]
async fn request_membership(
    &self,
    asociation_id: Path<Uuid>,
    post_data: Json<models::api::MembershipRequest>,
    data: Data<&ServerData>,
    auth: JWTBearerAuth,
) -> Result<Json<models::database::Member>> {
    use schema::members::dsl::*;

    let conn = &mut data.data_pool.get().map_err(error::InternalServerError)?;

    if auth.0.sub != post_data.0.user_id {
        return Err(error::Error::from_string(
            "Cannot request membership for another user.",
            poem::http::StatusCode::BAD_REQUEST,
        ));
    }

    let member = models::database::NaiveMember {
        user_id: post_data.0.user_id,
        asociation: asociation_id.0,
        is_accepted: false,
        accepted_date: None,
        expiry_date: None,
        label: None,
        board_status: models::database::BoardStatus::False,
    };
    let result = diesel::insert_into(members)
        .values(member)
        .returning(models::database::Member::as_returning())
        .get_result(conn)
        .map_err(error::InternalServerError)?;

    Ok(Json(result))
}
```

Este sistema de tipos es lo que permite a `poem` generar la especificación de *OpenAPI 3.0* directamente a partir del código.

3.2.4. Seguridad y sistema de autenticación

Danubit utiliza un sistema de autenticación propio, en vez de depender en un sistema externo, como podría haber sido el Sistema de Autenticación Única (SIU) de la UPM.

Aunque la capacidad de utilizar uno de estos sistemas es una funcionalidad que podría ser útil en el futuro, tener un sistema de usuarios propio permite que Danubit se pueda adaptar a otros contextos, distintos de aquel para el que fue diseñado.

Se ha puesto especial atención en la implementación de un sistema de autenticación que, aunque sencillo, sea seguro y fiable. La especificación de la API del sistema de autenticación, que está implementado por separado a la API principal del sistema, se encuentra detallada en el anexo B.

El sistema de autenticación requiere de que la comunicación entre el cliente y el servidor este cifrada, es decir, cualquier despliegue de la aplicación tendría que ser *HTTPS-Only*, lo cual es estándar en internet a día de hoy. A continuación se describen los procesos de registro y de login.

Los usuarios se registran con un nombre de usuario, nombre completo, email y contraseña. La contraseña se cifra usando el algoritmo Argon2id[9], que provee la máxima seguridad contra ataques entre los algoritmos modernos. El uso de este algoritmo significa que la operación de registro, así como la de login, son lentas, tardando aproximadamente 1 segundo. Esto no es un gran problema, dada la alta capacidad de paralelismo del servidor, y es uno de los componentes que hace que el algorimo sea más resistente a ataques de fuerza bruta.

Para hacer login, los usuarios han de proveer su correo y su contraseña, la contraseña se cifra y los *hashes* se comparan. En caso de que el login sea correcto, se genera un JWT [10], que detalla los permisos del usuario, que tiene 24h hasta su caducidad. Este sistema de login tiene algunas desventajas, como por ejemplo, el hecho de que un usuario no puede hacer *logout* e invalidar un token ya generado, ya que el servidor no guarda ningún tipo de estado o de sesión. Una vez generado un token, con tal de que esté correctamente firmado con la clave privada del servidor, Danubit se fía de que los permisos descritos en el son correctos. Esto permite evitar que se hagan muchas peticiones a la base de datos, como sería el caso si, por ejemplo, se comprobasen las credenciales con cada petición, pero también significa que cualquier cambio de permisos, como un cambio de junta, tarda 24 horas en hacerse efectivo.

3.3. Frontend

El frontend de Danubit tiene varios propósitos importantes, por una parte tenía que servir de plataforma para que las asociaciones se diesen a conocer, por otra tenía que servir a los alumnos como fuente de información sobre qué actividades se estaban organizando y cuando, y por último, tenía que ofrecer una experiencia intuitiva a las juntas de las asociaciones para manejar sus miembros, actividades, documentos y materiales.

Aunque la idea del proyecto inicialmente era terminar con un frontend completo que permitiese todas estas actividades, las limitaciones de tiempo y la complejidad de desarrollo del backend, han hecho que esto sea imposible. El producto final es pequeño pero funcional, se han limitado las funcionalidades mucho para que fuera posible probarlo y tener métricas sobre su usabilidad y accesibilidad. Afortunadamente, hay interés por continuar este trabajo entre los miembros de asociaciones, y ya se han planteado lineas de trabajo a futuro para completar la aplicación. Estas lineas se comentan en el capítulo 5.

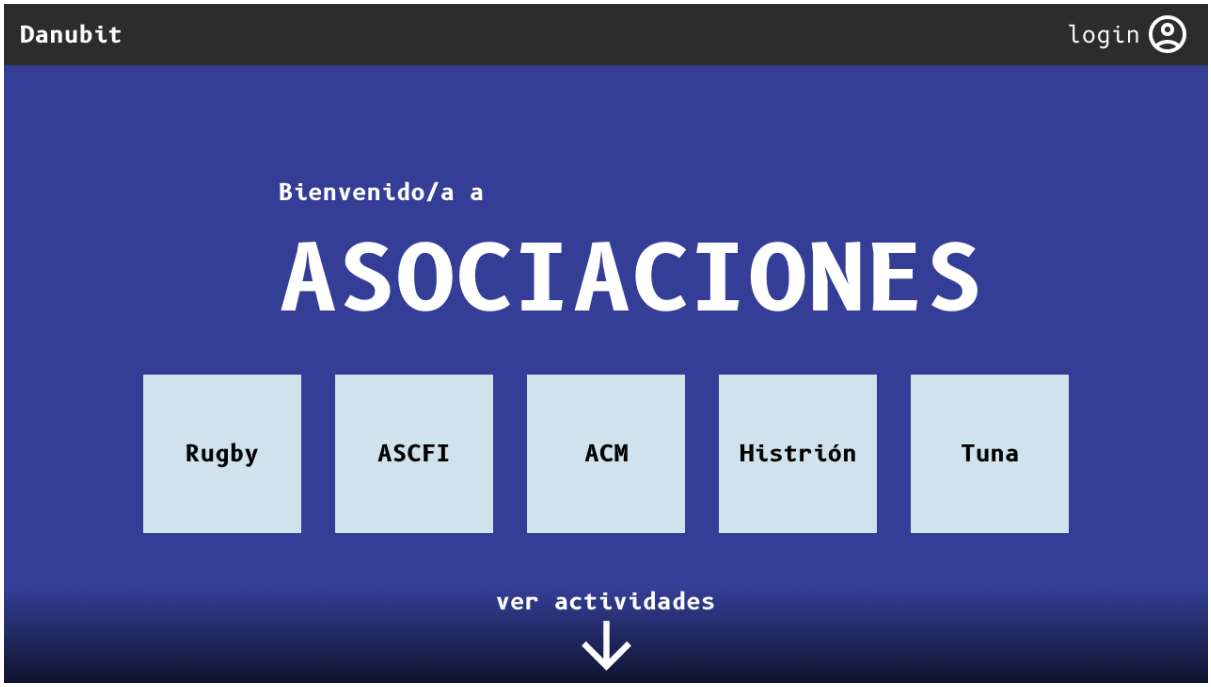

Figura 3.5: Página principal del prototipo de baja fidelidad.

3.3.1. Prototipos de usabilidad

De cara a desarrollar el frontend de Danubit se crearon 3 prototipos de usabilidad: uno de bajo nivel del portal público, otro de alto nivel, y uno de la parte de manejo de asociaciones, que solo es accesible cuando se tienen permisos de junta de una asociación.

El primer prototipo de baja fidelidad se desarolló con la herramienta de prototipado Figma [11]. Sirvió para decidir qué páginas debían de existir en el portal público, y qué maneras de presentar la información eran más intuitivas. Además de este prototipo, se dibujaron una serie de *sketches* con distintas posibles organizaciones y posibilidades para las paginas iniciales y se decidió de forma colaborativa entre miembros de las juntas de algunas asociaciones como seguir hacia delante.

Para el prototipo de alta fidelidad se pasó a utilizar una herramienta alternativa y *open source* llamada PenPot [12] tiene mucho en común con el de baja fidelidad, ya que las pruebas que se hicieron con el primero dieron resultados satisfactorios. Para este prototipo se creo una pequeña guía de colores y tipografías que pretendía dar coherencia a la plataforma, limitando el número de estilos aplicables a cada elemento. Esta guía estilística, sin embargo, se vio bastante modificada de cara al producto final, por temas estéticos y de accesibilidad.

Por último, el prototipo de la dashboard esta desarrollado en PenPot y vuelve a ser de bajo nivel. En este caso, se prototiparon muchas funcionalidades que finalmente no serían parte de la aplicación final como parte de este proyecto. El diseño del dashboard es muy simple ya que prácticamente todas las secciones consisten de una tabla donde enseñar la información con un par de botones para las acciones relevantes.

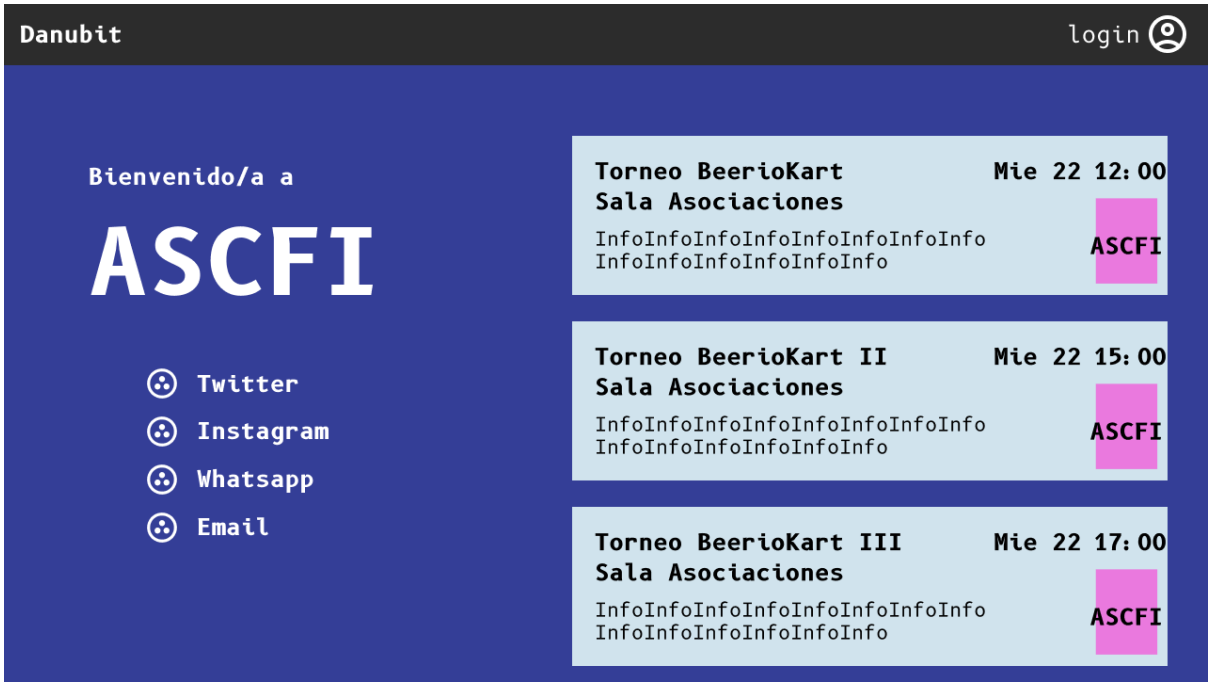

Figura 3.6: Página de asociación del prototipo de baja fidelidad.

Figura 3.7: Página principal del prototipo de alta fidelidad.

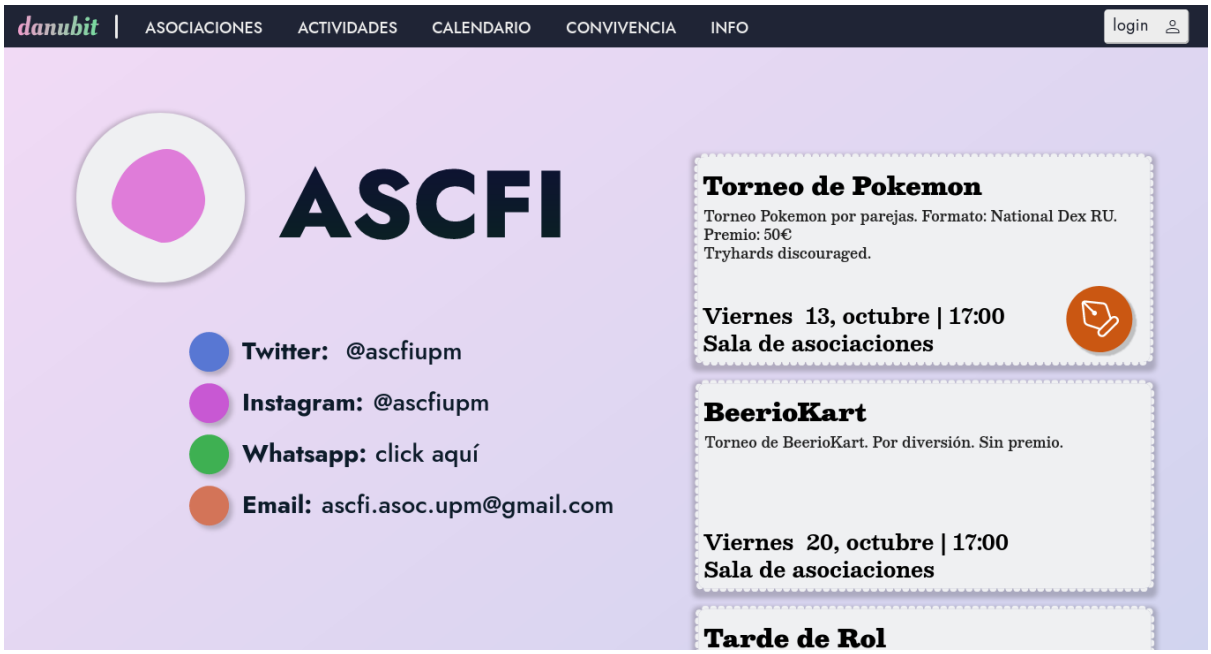

Figura 3.8: Página de asociación del prototipo de alta fidelidad.

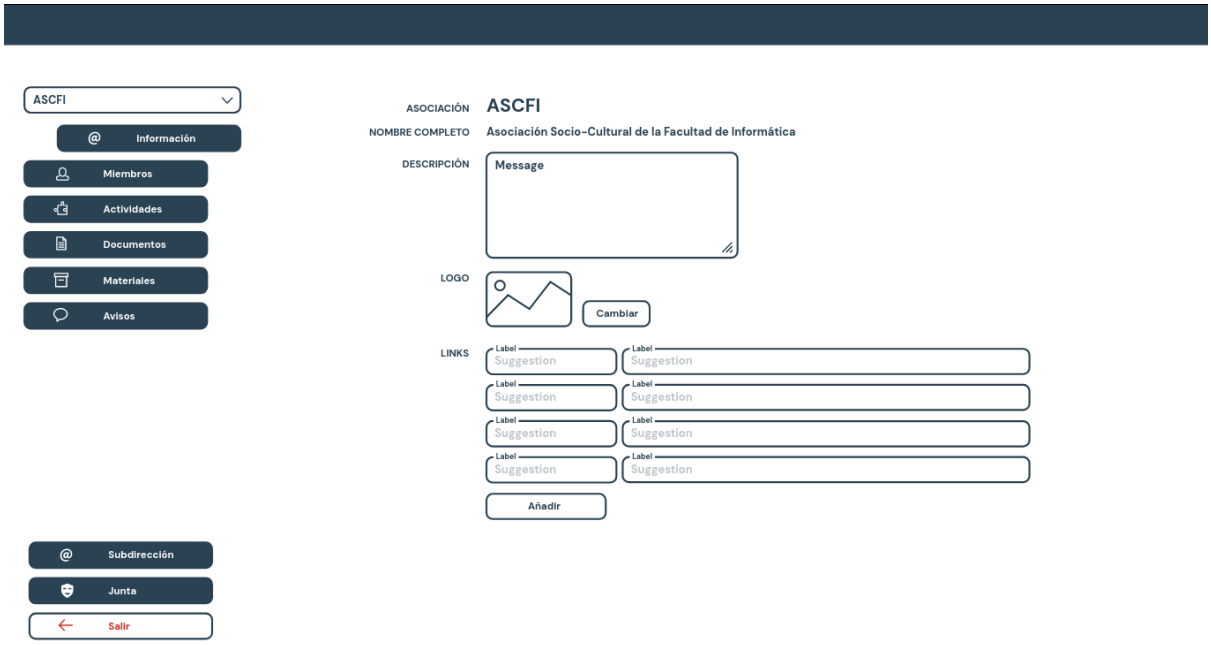

Figura 3.9: Prototipo del dashboard, sección de información.

Figura 3.10: Versión final del frontend.

3.3.2. Implementación

El frontend final es una versión reducida de los prototipos implementada en React. El servidor web de `next.js` se encarga de servir la aplicación al navegador, pero las peticiones al backend se hacen directamente desde el navegador, y los componentes de react se rellenan con los datos obtenidos del backend. Esto significa que casi todos ellos tienen que poder reaccionar a recibir un numero extraño de datos: listas vacías o tan llenas que se salen de la pantalla, o errores de la API.

Actualmente el sistema de autenticación es de tipo *Bearer*, y por tanto significa que el token tiene que ser añadido manualmente a todas las peticiones que lo quieran. Por ahora el token se pierde al recargar la página porque no se guarda ni en el *local storage* del navegador ni en una *cookie*.

Capítulo 4

Evaluación y resultados

En este capítulo se recogen los resultados de 2 evaluaciones de usabilidad: La primera pertenece al prototipo de alta fidelidad de la parte pública y el prototipo del dashboard, la segunda es del frontend final. Además, en la segunda sección se detallan las conclusiones personales sacadas del desarrollo de este proyecto.

Las evaluaciones de usabilidad han seguido el formato que se sigue en la asignatura de Interacción Persona-Ordenador y se han usado los cuestionarios de la asignatura para realizarlos. Cada evaluación se ha hecho con 5 personas [13], por lo que era necesario extraer información de forma eficiente. Se ha prestado bastante atención a los comentarios que los participantes, ya que todos han tenido unos resultados bastante positivos en el uso de la plataforma.

4.1. Evaluación

4.1.1. Prototipos de usabilidad

4.1.1.1. Datos demográficos

En la primera evaluación de usabilidad participaron 3 hombres, 1 mujer y 1 persona no binarie, entre los 20 y los 26 años. Todos estudiantes de la escuela y miembros de asociaciones. 4 de las 5 personas además pertenecían a la junta de alguna de las asociaciones. De media usaban ordenadores 6,6h al día, y el móvil 7,2h.

4.1.1.2. Tareas y métricas

La primera evaluación consistió de 4 tareas:

1. Encuentra el Twitter de ASCFI.
2. Encuentra qué actividad se organiza el 26 de octubre.
3. Inscríbete en la próxima actividad de ASCFI.
4. Acepta una de las peticiones de membresía que tiene pendientes ASCFI.

Todos los participantes terminaron con éxito todas las tareas, a continuación se presenta la media de clicks y de tiempo para cada tarea.

Tarea	Media Clicks	Varianza Clicks	Media Tiempo	Varianza Tiempo
Tarea 1	1.2	0.2	6.4	16.3
Tarea 2	1.2	0.2	16.6	194.3
Tarea 3	2.6	1.8	18.4	86.8
Tarea 4	3.7	1.7	26.2	124.6

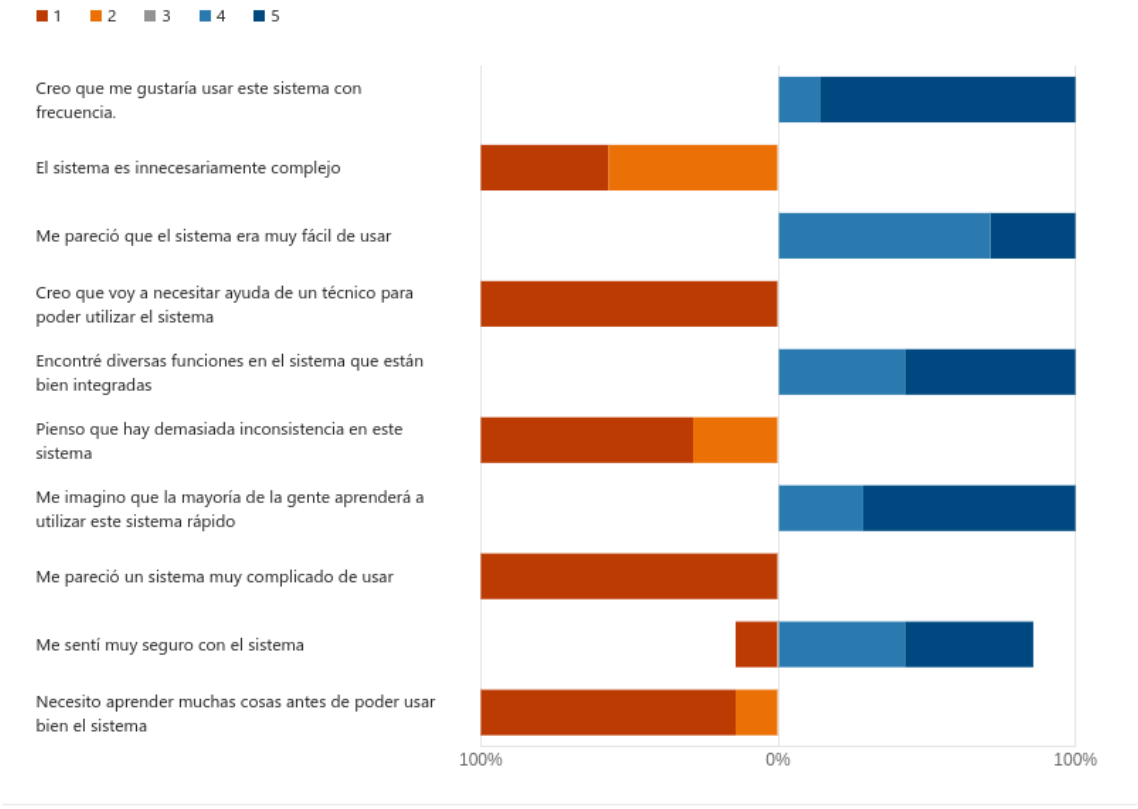

Figura 4.1: Resultados del cuestionario de *System Usability Scale* para el prototipo

4.1.1.3. Cuestionario SUS

Como se puede observar en la figura 4.3 los resultados del cuestionario de *System Usability Scale* son muy positivos, y reflejan que el sistema es usable. Esto se debe probablemente a la naturaleza limitada a nivel de funcionalidades del prototipo y, por tanto, a la interfaz simple y limpia.

4.1.1.4. Cuestionario UEQ

Ocurre algo similar con el cuestionario UEQ, los resultados son en general muy positivos, lo cual fue buen indicador de que era adecuado seguir adelante con esta estructura de frontend.

4.1.1.5. Valoraciones de los participantes

¿Cuáles son los problemas principales que has encontrado al usar este prototipo?

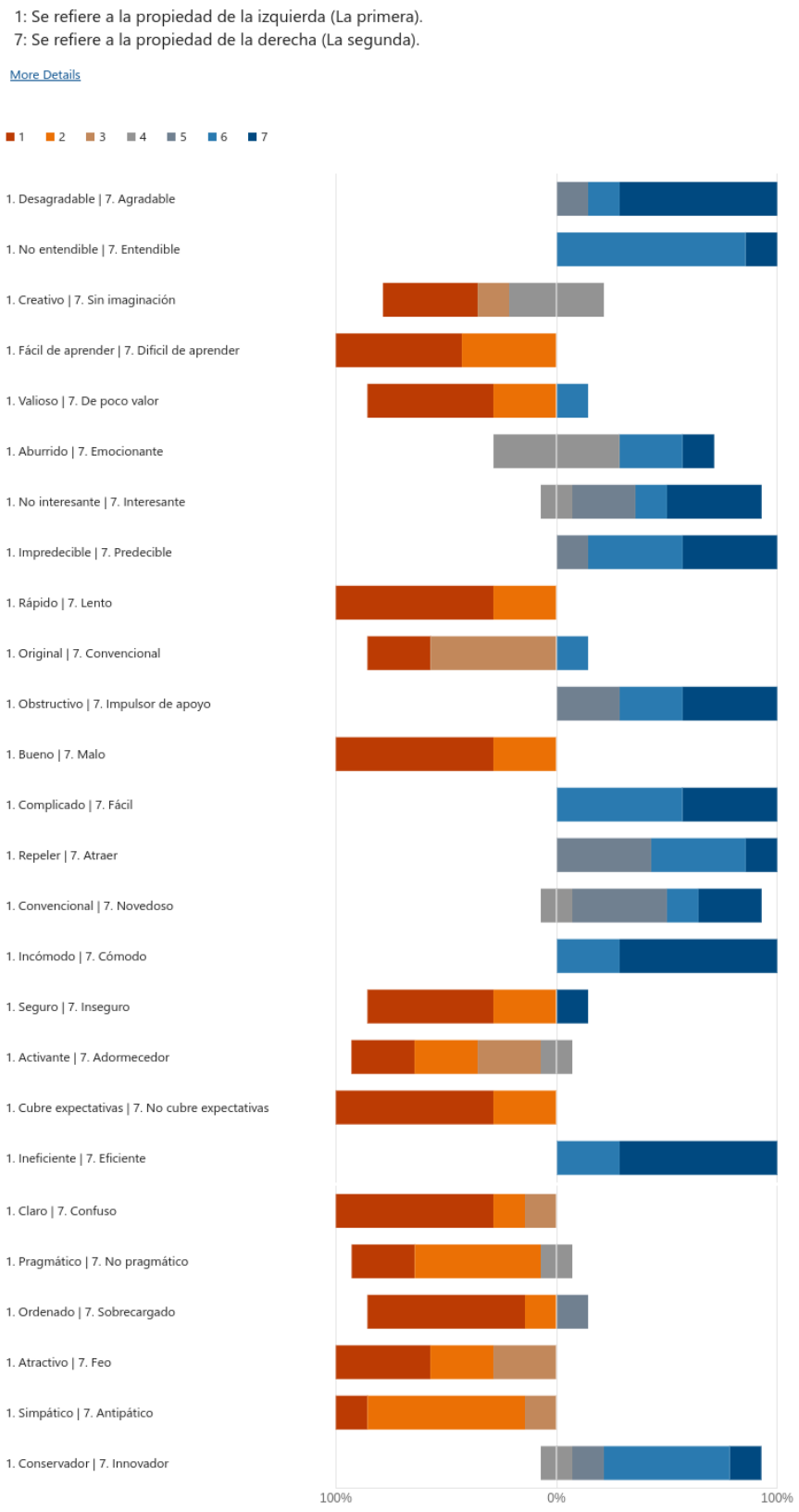

Figura 4.2: Resultados del cuestionario de *User Experience Questionaire* para el prototipo

- Dudé sobre las diferencias entre el apartado de Actividades y Calendario
- No me he enfrentado a ningún problema. Me podría faltar un prototipo más completo para trastear un poco más.
- Daltónicos pueden tener problemas con los colores de los símbolos del menú principal
- El orden de los eventos
- Puede que el prototipo se encuentre sobrecargado un poco de "fancinessz resulte confuso en una primera toma de contacto.

¿Cuál es la parte del prototipo que crees que es la más oscura o difícil de entender? ¿Por qué?

- Quizás el signo para inscribirse a una tarea, que no supe muy bien qué era. La confusión fue muy breve.
- Al principio no sabía como se ordenaban las actividades.
- La barra de botones de arriba; hay algunas opciones que son demasiado parecidas (actividades y calendario por ejemplo).
- No creo que tenga o no la he visto.
- Las ventanas de calendario y actividades tienen conceptos muy similares y se confunden fácilmente.

¿Qué te ha gustado más del prototipo? ¿Por qué?

- Me gustó mucho la interfaz sencilla y vistosa, especialmente cómo está dividida la información en bloques agradables a la vista, con una gama cromática también muy agradable. También me gustó mucho la página de inicio en la que aparecen las distintas asociaciones a elegir, expresa la relevancia que busca aportar este proyecto.
- Era muy eficiente y directo, con un par de clics se llegaba a cualquier cosa que buscaras.
- Su utilidad; la gente puede saber mejor las actividades de asos y la gente que organiza dichas actividades pueden saber el número de participantes.
- Templado, no te presiona la interfaz.
- La simpleza que tiene y todo el potencial. Puede evitar muchísimos quebraderos de cabeza al usuario.

¿Puedes describir tu experiencia general al usar el producto?

- Me ha resultado muy agradable y divertido usar una herramienta dedicada a las asociaciones y sus actividades; como miembro de ellas y parte de su junta, me ha resultado muy relajante ver lo ordenada que podría estar la programación y lo fácil y rápido que resultaría inscribirse a las actividades.
- Muy cómoda y directa, todo ha sido muy fácil de entender.
- Muy buena y intuitiva. Me gusta su diseño simple.

Tarea	Media Clicks	Varianza Clicks	Media Tiempo	Varianza Tiempo
Tarea 1	3.6	9.6	54.6	686.3
Tarea 2	2	0	7.8	10.2
Tarea 3	2.8	1.7	12.4	29.8
Tarea 4	4.8	1.7	28.6	131.8
Tarea 5	4.2	4.7	81.4	722.8

- Buena, he podido realizar las tareas de forma eficiente y sin distracciones, además de ser capaz de consultar la información con claridad y proceso.
- En general ha sido buena, excepto, tal vez, por los fallos descritos anteriormente. Tiene mucho potencial

4.1.2. Frontend

4.1.2.1. Datos demográficos

En la primera evaluación de usabilidad participaron 2 hombres, 2 mujeres y 1 persona no binarie, entre los 21 y los 25 años. Tres de ellos estudiantes de la escuela y miembros de asociaciones. 2 de las 5 personas además pertenecían a la junta de alguna de las asociaciones. De media usaban ordenadores 7,8h al día, y el móvil 6,6h.

4.1.2.2. Tareas y métricas

La segunda evaluación consistía en 5 tareas:

1. Registrarse en la plataforma.
2. Pedir unirse a una asociación.
3. Registrarse para una actividad.
4. Con una cuenta de administrador aceptar la petición de membresía.
5. Con una cuenta de administrador crear una nueva actividad.

Todos los participantes terminaron con éxito todas las tareas, a continuación se presenta la media de clicks y de tiempo para cada tarea.

4.1.2.3. Cuestionario SUS

Como se puede observar en la figura 4.3 los resultados del cuestionario System Usability Scale son bastante positivos. Es remarcable que los usuarios no se sintieron tan seguros con el software como sería optimo. Esto es probablemente atribuible a la falta de pulido que tenía el frontend en el momento de la evaluación. Un frontend más terminado y con más funcionalidades debería dar una sensación de producto completo mayor y por tanto dar más confianza.

4.1.2.4. Cuestionario UEQ

Los resultados del cuestionario UEQ que se pueden observar en la figura 4.4 son igual de favorables, aunque cabe resaltar que los usuarios no encontraron el fron-

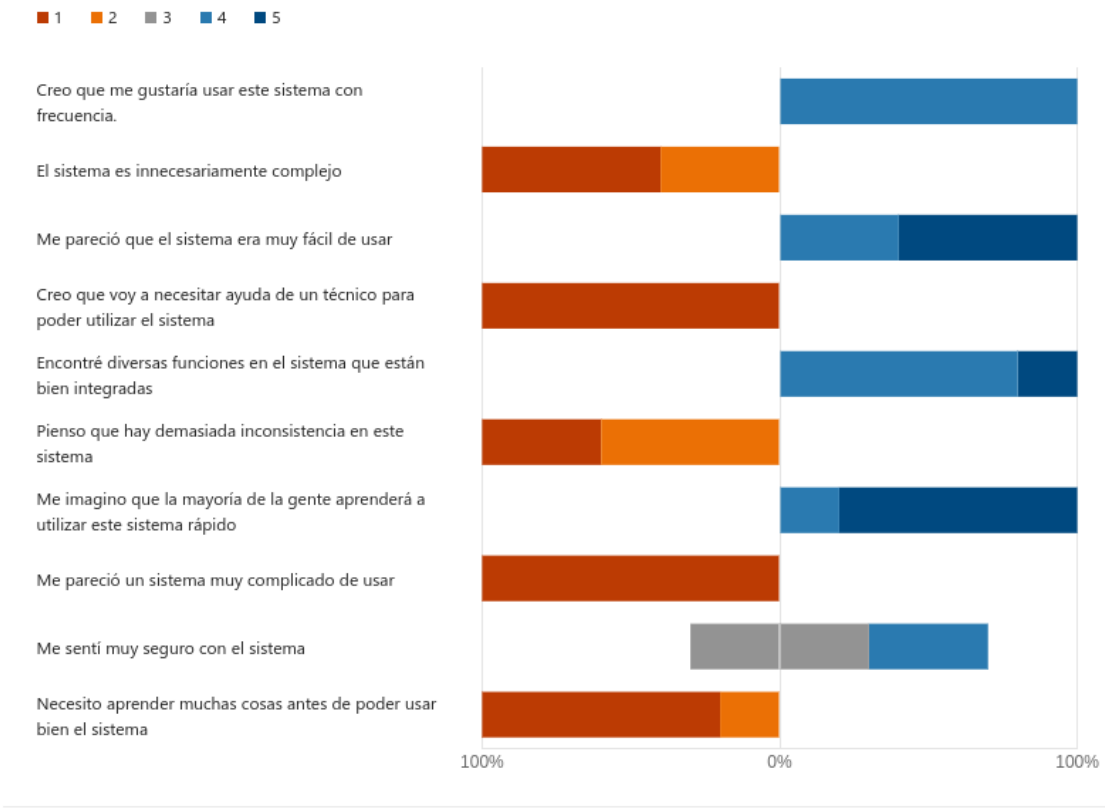

Figura 4.3: Resultados del cuestionario de *System Usability Scale* para el frontend final

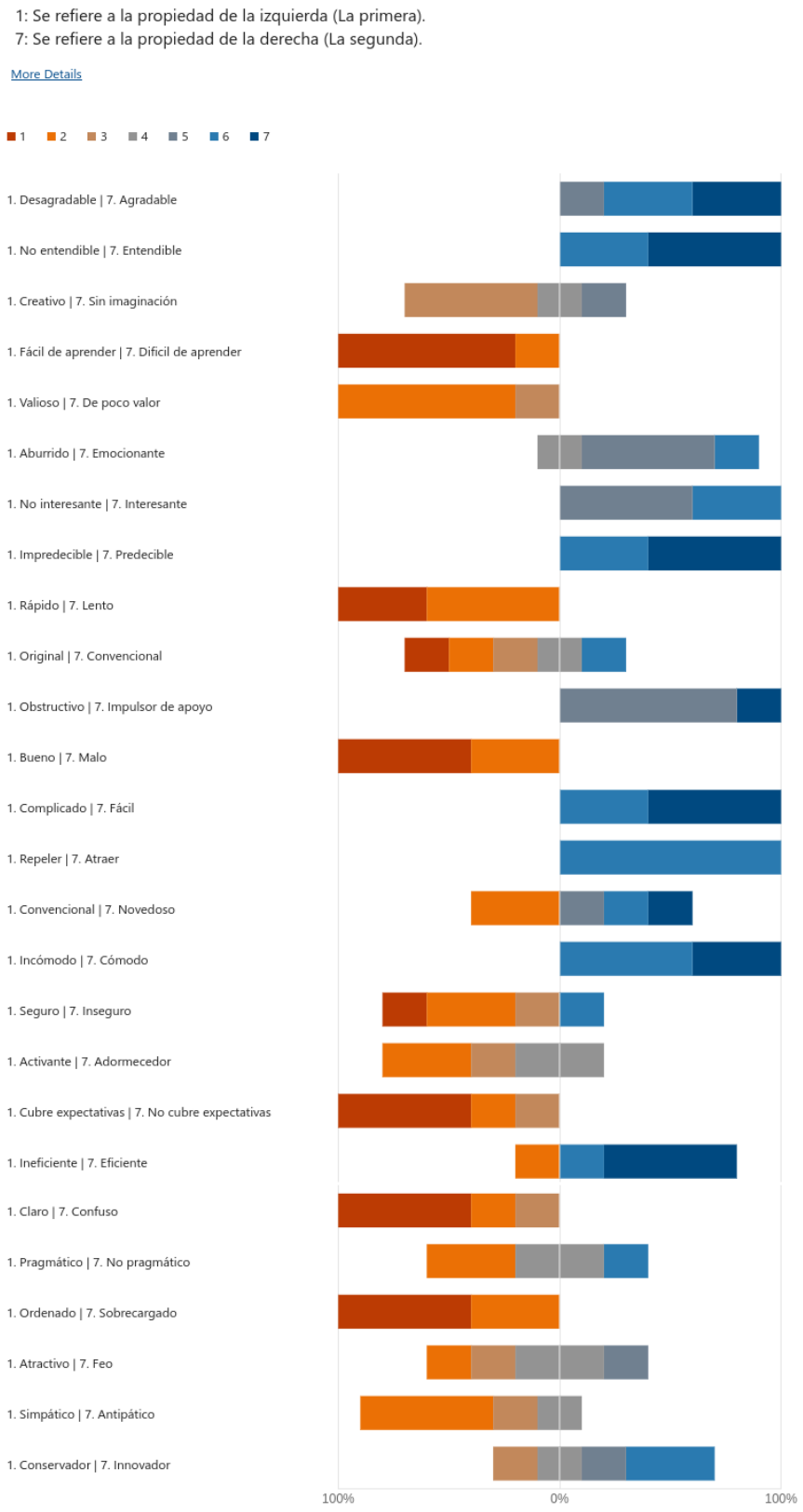

Figura 4.4: Resultados del cuestionario de *User Experience Questionaire* para el frontend final

tend particularmente bonito u original. Estos resultados, contrastados con los del prototipo, indican que quizás se podría poner algo de mayor atención en la parte estética de la plataforma. Esto no era uno de los objetivos principal del trabajo, pero de cara a crear un software que sea agradable de usar es importante.

4.1.2.5. Valoraciones de los participantes

¿Cuáles son los problemas principales que has encontrado al usar este prototipo?

- No hay botón de atrás, pero luego me di cuenta que era lo mismo que ir al aparto de asociaciones.
- Faltaba cierta información en algunas pantallas, como el nombre de la asociación en el panel de gestión de la misma.
- Las faltas de ortografía.
- Hay faltas ortográficas.
- En vez de inscrita/o ? Sería mejor un termino neutro como ïnscripción realizada".

¿Cuál es la parte del prototipo que crees que es la más oscura o difícil de entender? ¿Por qué?

- La seguridad, no se cómo va eso, pero parece funcionar bien.
- Diría que salvo pequeñas cosas, ya explicadas, en general es bastante entendible y sencillo de utilizar.
- Nada.
- Nada, es muy sencillo.
- No hay títulos de página pantalla como "dashboard.^{en}cima de los logos de asociaciones algo más de especificar el lugar en que está en cada momento.

¿Qué te ha gustado más del prototipo? ¿Por qué?

- La sencillez, ya que el resto de apps de universidad tiene mucha feature y poca chica. Esta es sencilla y cumple su tarea principal de manera eficiente.
- La comodidad que tiene para realizar las distintas operaciones. Creo que se encuentra bien estructurada.
- La eficiencia
- La simplicidad.
- Lo sencillo pero interactivo.

¿Puedes describir tu experiencia general al usar el producto?

- Lo prefiero antes que el de mi universidad.
- Ha sido una buena experiencia, considero que es una aplicación lo suficientemente sencilla para que cualquiera pudiera usarla. Además es muy útil y creo que puede ayudar muchísimo a las asociaciones de la universidad.

- Me parece una idea innovadora y bien ejecutada.
- Puede q el hecho de que todos los administradores puedas crear actividades, incluso de otras asociaciones de las que no son nada, no este bien implementado.
- Me ha gustado mucho la fluidez y rapidez de los pasos a completar de las tareas

Capítulo 5

Impacto y trabajo futuro

Dado el carácter limitado del frontend, el sistema aún no se ha puesto en uso para las asociaciones de la escuela, por lo que su impacto actual es muy limitado. Durante su desarrollo, varios miembros de las juntas de varias asociaciones han mostrado su interés porque se instale y se use una vez terminado, lo cual parece apuntar a que existe demanda. Además, varios miembros de algunas asociaciones han mostrado interés en continuar el trabajo para llevarlo hasta un estado más operable.

5.1. Trabajo futuro

A partir de la dinámica de grupo que se realizó con Subdirección de Alumnos para extraer los requisitos del sistema, se sacaron muchas posibles funcionalidades de la plataforma que finalmente no han sido implementadas. Además, durante el desarrollo de la misma han surgido diversas ideas que podrían ser útiles para las asociaciones. A continuación, se listan posibles adiciones al sistema que podrían construirse en futuros trabajos, sin ningún orden particular.

- Completar el frontend para llevarlo hasta la paridad de funcionalidades con el backend actual.
- Añadir un módulo SMTP al backend para que sea capaz de mandar correos de aviso a los distintos usuarios.
- Crear una newsletter a partir de los datos de actividades de la base de datos que se envíe opcionalmente a miembros que lo pidan una vez al mes.
- Diseñar un sistema de conexión entre usuarios dentro de la plataforma para poder enterarse de las actividades a las que van a ir personas cercanas.
- Añadir la posibilidad de comunicarse con un sistema de autenticación como el SIU de la UPM para limitar el número de logins asociados a la escuela que tienen que recordar los usuarios.
- Añadir un sistema de comunicados para que las asociaciones puedan enviar mensajes importantes a sus miembros de una manera oficial y unidireccional.
- Añadir un sistema de tarjetas de membresía digitales que permitan a los miembros conocer y demostrar su pertenencia a las distintas asociaciones.

Capítulo 6

Discusión

Danubit es un proyecto personal que llevaba queriendo hacer desde hace ya unos años como una forma de dejar algo de mi parte a las Asociaciones de Alumnos a las que he pertenecido durante mis cinco años de carrera. Además como miembro de la junta de una asociación muchos de los problemas de Danubit intenta solventar han sido problemas diarios y muy cercanos durante algunos de esos 5 años.

Desafortunadamente, por razones ajenas al trabajo y a la carrera, el trabajo se ha topado con bastantes contratiempos, su desarrollo se ha tenido que ver fraccionado en dos partes, y su entrega pospuesta a la evaluación extraordinaria. Además por disputas personales con ciertos miembros de las juntas de algunas asociaciones, el desarrollo y la evaluación del software final se ha visto algo dificultado.

6.1. Puntos fuertes

Este ha sido mi primer proyecto trabajando con Rust, y ha sido una experiencia bastante contrastante con el desarrollo que he hecho en otros lenguajes para la carrera y para mi trabajo. El modelo de programación contra el compilador, en el que el mismo proceso de compilación te guía sobre como arreglar problemas, es bastante curioso. Y es particularmente sorprendente encontrarse con que una vez consigues que el programa compile, muy frecuentemente, hace exactamente lo que quieres sin tener que estar buscando muchos *bugs* escondidos en el código.

La elección de tecnologías para el backend ha ralentizado mucho el trabajo, pero gracias a esas elecciones Danubit tiene un backend robusto, que era una de las preocupaciones principales de cara a desarrollar un software que se iba a utilizar activamente por asociaciones.

6.2. Puntos débiles

Aunque me hubiera gustado dejar la herramienta en un estado de completitud mayor, estoy contento con el trabajo que he sido capaz de hacer, y con la motivación que he visto dentro de asociaciones para continuarlo, pero un frontend más completo hubiera permitido hacer una prueba real de uso de la plataforma, y su instalación para que se comenzase a usar de cara al Proyecto Inicio del año que viene.

6.3. Conclusión personal

Hacer este trabajo ha sido una oportunidad importante para conectar muchos de los conocimientos de la carrera en un único proyecto, desde "Bases de Datos.º "Sistemas Orientados a Servicios"hasta "Programming Scalable Systems"dado el estilo de programación muy funcional que tiene Rust. Como ya he mencionado, me quedo satisfecho de poder haber desarrollado un buen comienzo para una herramienta que, con algo de interés por parte de las asociaciones de la escuela, tiene el potencial de convertirse en una gran ayuda para las asociaciones en muchas áreas.

Referencias Técnicas

[1] Admidio Developers. *Admidio*. URL: https://www.admidio.org (visitado 25-06-2024).

[2] WildApricot Inc. *Wild Apricot*. URL: https://www.wildapricot.com (visitado 25-06-2024).

[3] The World Café Community Foundation. *World Café Method*. URL: https://theworldcafe.com/key-concepts-resources/world-cafe-method/ (visitado 25-06-2024).

[4] PostgreSQL Global Development Group. *PostgreSQL*. URL: https://www.postgresql.org/ (visitado 25-06-2024).

[5] Rust Foundation. *Rust Programming Language*. URL: https://www.rust-lang.org/ (visitado 25-06-2024).

[6] Diesel ORM Developers. *Diesel ORM*. URL: https://diesel.rs/ (visitado 25-06-2024).

[7] Poem Developers. *Poem Web Framework*. URL: https://github.com/poem-web/poem (visitado 25-06-2024).

[8] OpenAPI Iniciative. *OpenAPI Specification*. URL: https://spec.openapis.org/oas/latest.html (visitado 25-06-2024).

[9] Dmitry Khovratovich Daniel Dinu. *Argon2*. URL: https://github.com/p-h-c/phc-winner-argon2 (visitado 25-06-2024).

[10] et al. Michael B. Jones Microsoft. *Jason Web Tokens*. URL: https://jwt.io/ (visitado 25-06-2024).

[11] Figma Inc. *Figma*. URL: https://www.figma.com (visitado 25-06-2024).

[12] Kaleidos. *Penpot*. URL: https://penpot.app/ (visitado 25-06-2024).

Bibliografía

[13] Jakob Nielsen. *Why You Only Need to Test with 5 Users*. 2000. URL: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ (visitado 25-06-2024).

Apéndice A

Especificación de la API de Danubit

Danubit

Overview

Version

0.0.2

GET /asociations

Response 200:

Content: application/json; charset=utf-8 | Array<Asociation>

```
{
 id: string;
 short_name: string;
 long_name: string;
 email: string;
 description: string;
 is_public_joinable: boolean;
 info: undefined;
 manager?: integer;
 logo?: integer;
}
```

POST /asociations

Request Body:

Content: application/json; charset=utf-8 | NaiveAsociation

```
{
 short_name: string;
 long_name: string;
 email: string;
 description: string;
 is_public_joinable: boolean;
 info?: undefined;
 manager?: integer;
 logo?: integer;
}
```

Response 200:

Content: application/json; charset=utf-8 | Asociation

```
{
```

```
id: string;
short_name: string;
long_name: string;
email: string;
description: string;
is_public_joinable: boolean;
info: undefined;
manager?: integer;
logo?: integer;
}
```

GET /asociations/{asociation_id}

Request Parameters:

```
asociation_id: string;
```

Response 200:

Content: application/json; charset=utf-8 | Asociation

```
{
id: string;
short_name: string;
long_name: string;
email: string;
description: string;
is_public_joinable: boolean;
info: undefined;
manager?: integer;
logo?: integer;
}
```

PUT /asociations/{asociation_id}

Request Parameters:

```
asociation_id: string;
```

Request Body:

Content: application/json; charset=utf-8 | Asociation

```
{
id: string;
short_name: string;
long_name: string;
email: string;
```

```
 description: string;
 is_public_joinable: boolean;
 info: undefined;
 manager?: integer;
 logo?: integer;
}
```

Response 200:

Content: application/json; charset=utf-8 | Association

```
{
 id: string;
 short_name: string;
 long_name: string;
 email: string;
 description: string;
 is_public_joinable: boolean;
 info: undefined;
 manager?: integer;
 logo?: integer;
}
```

GET /asociations/byName/{asociation_short_name}

Request Parameters:

```
asociation_short_name: string;
```

Response 200:

Content: application/json; charset=utf-8 | Association

```
{
 id: string;
 short_name: string;
 long_name: string;
 email: string;
 description: string;
 is_public_joinable: boolean;
 info: undefined;
 manager?: integer;
 logo?: integer;
}
```

GET /managers

Response 200:

Content: application/json; charset=utf-8 | Array<Manager>

```
{
 id: integer;
 user_id: string;
 name: string;
 contact_email: string;
 admin_email?: string;
 material_email?: string;
 print_email?: string;
 comms_email?: string;
}
```

GET /asociations/{asociation_id}/membershipRequests

Request Parameters:

```
asociation_id: string;
```

Response 200:

Content: application/json; charset=utf-8 | Array<FullMember>

```
{
 id: integer;
 user: User;
 asociation: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

POST /asociations/{asociation_id}/membershipRequests

Request Parameters:

```
asociation_id: string;
```

Request Body:

Content: application/json; charset=utf-8 | MembershipRequest

```
{
 user_id: string;
}
```

Response 200:

Content: application/json; charset=utf-8 | Member

```
{
 id: integer;
 user_id: string;
 association: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

PUT /associations/{association_id}/membershipRequests/{member_id}

Request Parameters:

```
association_id: string;
member_id: string;
```

Response 200:

Content: application/json; charset=utf-8 | Member

```
{
 id: integer;
 user_id: string;
 association: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

DELETE /associations/{association_id}/membershipRequests/{member_id}

Request Parameters:

```
association_id: string;
member_id: string;
```

Response 200:

GET /associations/{association_id}/members

Request Parameters:

```
association_id: string;
```

Response 200:

```
Content: application/json; charset=utf-8 | Array<FullMember>
```

```
{
 id: integer;
 user: User;
 association: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

PUT /associations/{association_id}/members/{member_id}

Request Parameters:

```
association_id: string;
member_id: string;
```

Request Body:

```
Content: application/json; charset=utf-8 | Member
```

```
{
 id: integer;
 user_id: string;
 association: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

Response 200:

```
Content: application/json; charset=utf-8 | Member
```

```
{
 id: integer;
 user_id: string;
 association: string;
 is_accepted: boolean;
 accepted_date?: string;
```

```
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

DELETE /asociations/{association_id}/members/{member_id}

Request Parameters:

```
association_id: string;
member_id: string;
```

Response 200:

GET /asociations/{association_id}/board

Request Parameters:

```
association_id: string;
```

Response 200:

Content: application/json; charset=utf-8 | Array<Member>

```
{
 id: integer;
 user_id: string;
 asociation: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

PUT /asociations/{association_id}/board/{member_id}

Request Parameters:

```
association_id: string;
member_id: string;
```

Request Body:

Content: application/json; charset=utf-8 | Member

```
{
 id: integer;
 user_id: string;
 asociation: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

Response 200:

Content: application/json; charset=utf-8 | Member

```
{
 id: integer;
 user_id: string;
 asociation: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

DELETE /asociations/{association_id}/board/{member_id}

Request Parameters:

```
association_id: string;
member_id: string;
```

Response 200:

GET /asociations/{association_id}/publicDocuments

Request Parameters:

```
association_id: string;
```

Response 200:

Content: application/json; charset=utf-8 | Array<Document>

```
{
 id: integer;
 asociation: string;
```

```
activity?: integer;
name: string;
description: string;
path: string;
creation_date: string;
is_current: boolean;
is_important: boolean;
is_manager_accessible: boolean;
is_public_accessible: boolean;
}
```

GET /asociations/{asociation_id}/documents

Request Parameters:

```
asociation_id: string;
```

Response 200:

```
Content: application/json; charset=utf-8 | Array<Document>
```

```
{
id: integer;
asociation: string;
activity?: integer;
name: string;
description: string;
path: string;
creation_date: string;
is_current: boolean;
is_important: boolean;
is_manager_accessible: boolean;
is_public_accessible: boolean;
}
```

POST /asociations/{asociation_id}/documents

Request Parameters:

```
asociation_id: string;
```

Request Body:

```
Content: multipart/form-data | {
file_data: DocumentDescription;
upload: string;
}
```

Response 200:

Content: application/json; charset=utf-8 | Document

```
{
 id: integer;
 association: string;
 activity?: integer;
 name: string;
 description: string;
 path: string;
 creation_date: string;
 is_current: boolean;
 is_important: boolean;
 is_manager_accessible: boolean;
 is_public_accessible: boolean;
}
```

PUT /asociations/{asociation_id}/documents/{document_id}

Request Parameters:

```
asociation_id: string;
document_id: string;
```

Request Body:

Content: application/json; charset=utf-8 | DocumentDescription

```
{
 asociation: string;
 activity?: integer;
 name: string;
 description: string;
 is_current: boolean;
 is_important: boolean;
 is_manager_accessible: boolean;
 is_public_accessible: boolean;
}
```

Response 200:

Content: application/json; charset=utf-8 | Array<Document>

```
{
 id: integer;
 asociation: string;
 activity?: integer;
 name: string;
 description: string;
 path: string;
```

```
 creation_date: string;
 is_current: boolean;
 is_important: boolean;
 is_manager_accessible: boolean;
 is_public_accessible: boolean;
}
```

GET /asociations/{asociation_id}/materials

Request Parameters:

```
 asociation_id: string;
```

Response 200:

Content: application/json; charset=utf-8 | Array<Material>

```
{
 id: integer;
 asociation: string;
 name: string;
 description: string;
 quantity: integer;
 is_lendable: boolean;
}
```

POST /asociations/{asociation_id}/materials

Request Parameters:

```
 asociation_id: string;
```

Request Body:

Content: application/json; charset=utf-8 | NaiveMaterial

```
{
 asociation: string;
 name: string;
 description: string;
 quantity: integer;
 is_lendable: boolean;
}
```

Response 200:

Content: application/json; charset=utf-8 | Material

```
{
```

```
id: integer;
asociation: string;
name: string;
description: string;
quantity: integer;
is_lendable: boolean;
}
```

GET /asociations/{asociation_id}/lendableMaterials

Request Parameters:

```
asociation_id: string;
```

Response 200:

Content: application/json; charset=utf-8 | Array<Material>

```
{
id: integer;
asociation: string;
name: string;
description: string;
quantity: integer;
is_lendable: boolean;
}
```

PUT /asociations/{asociation_id}/materials/{material_id}

Request Parameters:

```
asociation_id: string;
material_id: string;
```

Request Body:

Content: application/json; charset=utf-8 | Material

```
{
id: integer;
asociation: string;
name: string;
description: string;
quantity: integer;
is_lendable: boolean;
}
```

Response 200:

Content: application/json; charset=utf-8 | Material

```
{
 id: integer;
 asociation: string;
 name: string;
 description: string;
 quantity: integer;
 is_lendable: boolean;
}
```

DELETE /asociations/{asociation_id}/materials/{material_id}

Request Parameters:

```
asociation_id: string;
material_id: string;
```

Response 200:

GET /publicActivities

Request Parameters:

```
asociation_filter?: string;
```

Response 200:

Content: application/json; charset=utf-8 | Array<FullActivity>

```
{
 activity: Activity;
 organizers: Array<Asociation>;
 people_in_charge: Array<User>;
}
```

GET /memberActivities

Request Parameters:

```
asociation_filter?: string;
```

Response 200:

Content: application/json; charset=utf-8 | Array<FullActivity>

```
{
```

```
 activity: Activity;
 organizers: Array<Asociation>;
 people_in_charge: Array<User>;
}
```

GET /boardActivities

Request Parameters:

```
asociation_filter?: string;
```

Response 200:

```
Content: application/json; charset=utf-8 | Array<FullActivity>
```

```
{
 activity: Activity;
 organizers: Array<Asociation>;
 people_in_charge: Array<User>;
}
```

POST /activities

Request Body:

```
Content: application/json; charset=utf-8 | NewFullActivity
```

```
{
 activity: NaiveActivity;
 organizers: Array<string>;
 people_in_charge: Array<string>;
}
```

Response 200:

```
Content: application/json; charset=utf-8 | Activity
```

```
{
 id: integer;
 name: string;
 description: string;
 room: string;
 initial_date: string;
 duration: integer;
 is_multi_session: boolean;
 is_creditable: boolean;
 is_external: boolean;
 is_accepted: boolean;
 is_room_accepted: boolean;
```

```
 is_media_accepted: boolean;
 is_registration_needed: boolean;
 access: ActivityAccess;
 additional_info: undefined;
}
```

GET /activities/{activity_id}

Request Parameters:

```
activity_id: integer;
```

Response 200:

```
Content: application/json; charset=utf-8 | FullActivity
```

```
{
 activity: Activity;
 organizers: Array<Asociation>;
 people_in_charge: Array<User>;
}
```

PUT /activities/{activity_id}

Request Parameters:

```
activity_id: integer;
```

Request Body:

```
Content: application/json; charset=utf-8 | NewFullActivity
```

```
{
 activity: NaiveActivity;
 organizers: Array<string>;
 people_in_charge: Array<string>;
}
```

Response 200:

```
Content: application/json; charset=utf-8 | Activity
```

```
{
 id: integer;
 name: string;
 description: string;
 room: string;
 initial_date: string;
 duration: integer;
```

```
 is_multi_session: boolean;
 is_creditable: boolean;
 is_external: boolean;
 is_accepted: boolean;
 is_room_accepted: boolean;
 is_media_accepted: boolean;
 is_registration_needed: boolean;
 access: ActivityAccess;
 additional_info: undefined;
}
```

DELETE /activities/{activity_id}

Request Parameters:

```
activity_id: integer;
```

Response 200:

POST /activities/{activity_id}/media

Request Parameters:

```
asociation_id: string;
```

Request Body:

```
Content: multipart/form-data | {
 file_data: MediaDescription;
 upload: string;
}
```

Response 200:

```
Content: application/json; charset=utf-8 | Activity
```

```
{
 id: integer;
 name: string;
 description: string;
 room: string;
 initial_date: string;
 duration: integer;
 is_multi_session: boolean;
 is_creditable: boolean;
 is_external: boolean;
 is_accepted: boolean;
 is_room_accepted: boolean;
```

```
 is_media_accepted: boolean;
 is_registration_needed: boolean;
 access: ActivityAccess;
 additional_info: undefined;
}
```

DELETE /activities/{activity_id}/media/{media_id}

Request Parameters:

```
asociation_id: string;
media_id: string;
```

Response 200:

Content: application/json; charset=utf-8 | Activity

```
{
 id: integer;
 name: string;
 description: string;
 room: string;
 initial_date: string;
 duration: integer;
 is_multi_session: boolean;
 is_creditable: boolean;
 is_external: boolean;
 is_accepted: boolean;
 is_room_accepted: boolean;
 is_media_accepted: boolean;
 is_registration_needed: boolean;
 access: ActivityAccess;
 additional_info: undefined;
}
```

GET /activities/{activity_id}/registration

Request Parameters:

```
activity_id: integer;
```

Response 200:

Content: application/json; charset=utf-8 | Array<Registration>

```
{
 activity: integer;
 user_id?: string;
```

```
 registration_data: undefined;
}
```

POST /activities/{activity_id}/registration

Request Parameters:

```
activity_id: integer;
```

Request Body:

```
Content: application/json; charset=utf-8 | Registration
```

```
{
 activity: integer;
 user_id?: string;
 registration_data: undefined;
}
```

Response 200:

```
Content: application/json; charset=utf-8 | Registration
```

```
{
 activity: integer;
 user_id?: string;
 registration_data: undefined;
}
```

DELETE /activities/{activity_id}/registration

Request Parameters:

```
activity_id: string;
```

Response 200:

```
Content: application/json; charset=utf-8 | Activity
```

```
{
 id: integer;
 name: string;
 description: string;
 room: string;
 initial_date: string;
 duration: integer;
 is_multi_session: boolean;
 is_creditable: boolean;
 is_external: boolean;
```

```
 is_accepted: boolean;
 is_room_accepted: boolean;
 is_media_accepted: boolean;
 is_registration_needed: boolean;
 access: ActivityAccess;
 additional_info: undefined;
}
```

GET /session/board_of

Response 200:

Content: application/json; charset=utf-8 | Array<Asociation>

```
{
 id: string;
 short_name: string;
 long_name: string;
 email: string;
 description: string;
 is_public_joinable: boolean;
 info: undefined;
 manager?: integer;
 logo?: integer;
}
```

GET /session/member_of

Response 200:

Content: application/json; charset=utf-8 | Array<Asociation>

```
{
 id: string;
 short_name: string;
 long_name: string;
 email: string;
 description: string;
 is_public_joinable: boolean;
 info: undefined;
 manager?: integer;
 logo?: integer;
}
```

Schemas

Activity

```
{
 id: integer;
 name: string;
 description: string;
 room: string;
 initial_date: string;
 duration: integer;
 is_multi_session: boolean;
 is_creditable: boolean;
 is_external: boolean;
 is_accepted: boolean;
 is_room_accepted: boolean;
 is_media_accepted: boolean;
 is_registration_needed: boolean;
 access: ActivityAccess;
 additional_info: undefined;
}
```

ActivityAccess

```
string
Values: Public, Members, Board
```

Asociation

```
{
 id: string;
 short_name: string;
 long_name: string;
 email: string;
 description: string;
 is_public_joinable: boolean;
 info: undefined;
 manager?: integer;
 logo?: integer;
}
```

BoardStatus

```
string
Values: False, Board, ViceChair, Chair
```

Document

```
{
 id: integer;
 asociation: string;
 activity?: integer;
 name: string;
 description: string;
 path: string;
 creation_date: string;
 is_current: boolean;
 is_important: boolean;
 is_manager_accessible: boolean;
 is_public_accessible: boolean;
}
```

DocumentDescription

```
{
 asociation: string;
 activity?: integer;
 name: string;
 description: string;
 is_current: boolean;
 is_important: boolean;
 is_manager_accessible: boolean;
 is_public_accessible: boolean;
}
```

FullActivity

```
{
 activity: Activity;
 organizers: Array<Asociation>;
 people_in_charge: Array<User>;
}
```

FullMember

```
{
 id: integer;
 user: User;
 asociation: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
```

```
 label?: string;
 board_status: BoardStatus;
}
```

Manager

```
{
 id: integer;
 user_id: string;
 name: string;
 contact_email: string;
 admin_email?: string;
 material_email?: string;
 print_email?: string;
 comms_email?: string;
}
```

Material

```
{
 id: integer;
 asociation: string;
 name: string;
 description: string;
 quantity: integer;
 is_lendable: boolean;
}
```

MediaDescription

```
{
 name: string;
 activity?: integer;
 kind: MediaKind;
}
```

MediaKind

```
string
Values: Logo, Digital, Print, Screen, Banner, Extra
```

Member

```
{
 id: integer;
 user_id: string;
 asociation: string;
 is_accepted: boolean;
 accepted_date?: string;
 expiry_date?: string;
 label?: string;
 board_status: BoardStatus;
}
```

MembershipRequest

```
{
 user_id: string;
}
```

NaiveActivity

```
{
 name: string;
 description: string;
 room: string;
 initial_date: string;
 is_multi_session: boolean;
 is_creditable: boolean;
 is_external: boolean;
 is_accepted: boolean;
 is_room_accepted: boolean;
 is_media_accepted: boolean;
 is_registration_needed: boolean;
 access: ActivityAccess;
 additional_info?: undefined;
}
```

NaiveAsociation

```
{
 short_name: string;
 long_name: string;
 email: string;
 description: string;
 is_public_joinable: boolean;
 info?: undefined;
 manager?: integer;
 logo?: integer;
```

```
}
```

NaiveMaterial

```
{
 asociation: string;
 name: string;
 description: string;
 quantity: integer;
 is_lendable: boolean;
}
```

NewFullActivity

```
{
 activity: NaiveActivity;
 organizers: Array<string>;
 people_in_charge: Array<string>;
}
```

Registration

```
{
 activity: integer;
 user_id?: string;
 registration_data: undefined;
}
```

User

```
{
 id: string;
 username: string;
 name: string;
 surname: string;
 email: string;
 activated: boolean;
 password_hash?: string;
 additional_info: undefined;
}
```

Apéndice B

Especificación de la API de autenticación de Danubit

Danubit Auth

Overview

Version

0.0.2

POST /login

Request Body:

Content: application/json; charset=utf-8 | Login

```
{
 email: string;
 password: string;
}
```

Response 200:

Content: application/json; charset=utf-8 | LoginResponse

```
{
 id: string;
 username: string;
 token: string;
 expires_at: integer;
 manager_of: Array<string>;
 chair_of: Array<string>;
 board_of: Array<string>;
 member_of: Array<string>;
}
```

POST /signup

Request Body:

Content: application/json; charset=utf-8 | UserSignup

```
{
 username: string;
 name: string;
 surname: string;
 email: string;
 password: string;
 additional_info?: undefined;
}
```

Response 200:

POST /standin

Request Body:

Content: application/json; charset=utf-8 | Standin

```
{
 username: string;
 name: string;
 surname: string;
 email: string;
}
```

Response 200:

POST /change_username

Request Body:

Content: application/json; charset=utf-8 | UsernameChange

```
{
 email: string;
 new_username: string;
 password: string;
}
```

Response 200:

Content: text/plain; charset=utf-8 | string

POST /change_password

Request Body:

Content: application/json; charset=utf-8 | PasswordChange

```
{
 email: string;
 old_password: string;
 new_password: string;
}
```

Response 200:

Content: text/plain; charset=utf-8 | string

Schemas

Login

```
{
 email: string;
 password: string;
}
```

LoginResponse

```
{
 id: string;
 username: string;
 token: string;
 expires_at: integer;
 manager_of: Array<string>;
 chair_of: Array<string>;
 board_of: Array<string>;
 member_of: Array<string>;
}
```

PasswordChange

```
{
 email: string;
 old_password: string;
 new_password: string;
}
```

Standin

```
{
 username: string;
 name: string;
 surname: string;
 email: string;
}
```

UserSignup

```
{
 username: string;
 name: string;
```

```
 surname: string;
 email: string;
 password: string;
 additional_info?: undefined;
}
```

UsernameChange

```
{
 email: string;
 new_username: string;
 password: string;
}
```

Apéndice C

Datos brutos recogidos de la evaluación de usabilidad

10/20/2023 17:56	10/20/2023 18:06	10/20/2023 18:15	11/6/2023 17:00:	11/6/2023 17:06:		
user 1	user 2	user 3	user 4	user 5		
Sí	Sí	Sí	Sí	Sí		
1	1	1	2	1	1.2	0.2
4	12	5	9	2	6.4	16.3
Sí	Sí	Sí	Sí	Sí		
1	1	1	1	2	1.2	0.2
16	7	15	5	40	16.6	194.3
Sí	Sí	Sí	Sí	Sí		
2	1	2	4	4	2.6	1.8
26	16	8	12	30	18.4	86.8

	user 1	user 2	user 3	user 4	user 5
1. Desagradable | 7. Agradable	7	7	7	7	5
1. No entendible | 7. Entendible	7	6	6	6	6
1. Creativo | 7. Sin imaginación	1	4	1	1	4
1. Fácil de aprender | 7. Dificil de aprender	1	1	2	2	1
1. Valioso | 7. De poco valor	1	2	1	2	1
1. Aburrido | 7. Emocionante	7	4	6	6	4
1. No interesante | 7. Interesante	7	7	7	6	5
1. Impredecible | 7. Predecible	6	6	7	6	7
1. Rápido | 7. Lento	1	1	1	1	2
1. Original | 7. Convencional	1	3	3	1	3
1. Obstructivo | 7. Impulsor de apoyo	7	7	5	6	6
1. Bueno | 7. Malo	1	1	2	1	1
1. Complicado | 7. Fácil	7	6	6	6	6
1. Repeler | 7. Atraer	7	6	6	6	5
1. Convencional | 7. Novedoso	7	5	5	7	4
1. Incómodo | 7. Cómodo	7	7	7	7	6
1. Seguro | 7. Inseguro	1	1	7	2	2
1. Activante | 7. Adormecedor	1	3	1	2	3
1. Cubre expectativas | 7. No cubre expectativas	1	1	1	1	2
1. Ineficiente | 7. Eficiente	7	7	6	7	7
1. Claro | 7. Confuso	1	1	1	2	3
1. Pragmático | 7. No pragmático	1	2	1	2	4
1. Ordenado | 7. Sobrecargado	1	1	1	2	5
1. Atractivo | 7. Feo	1	1	3	1	3
1. Simpático | 7. Antipático	1	2	2	2	2
1. Conservador | 7. Innovador	7	4	6	6	6

		Tarea 1	Tarea 2	Tarea 3	Tarea 4	Tarea 5
Usuario 1	Clicks	2	2	2	4	3
	Tiempo	95	7	21	40	59
Usuario 2	Clicks	3	2	2	4	3
	Tiempo	38	4	14	23	127
Usuario 3	Clicks	9	2	5	7	8
	Tiempo	46	10	11	20	75
Usuario 4	Clicks	2	2	2	4	3
	Tiempo	29	6	7	18	65
Usuario 5	Clicks	2	2	3	5	4
	Tiempo	65	12	9	42	81
		3.6	2	2.8	4.8	4.2
		54.6	7.8	12.4	28.6	81.4
		9.3	0	1.7	1.7	4.7
		686.3	10.2	29.8	131.8	722.8

Timestamp	6/10/2024 20:55:3	6/11/2024 15:18:3	6/12/2024 0:45:26	6/28/2024 13:53:3	6/28/2024 17:07:0
Edad	24	25	21	21	22
Género	Hombre	Hombre	Mujer	Mujer	No binarie
Tiempo diario de uso de ordenador(es)	12	12	5	4	6
Tiempo diario de uso de móvil(es)	3	10	6	8	6
Creo que me gustaría usar este sistema con frecuencia.	4	4	4	4	4
El sistema es innecesariamente complejo.	1	2	1	2	1
Me pareció que el sistema era muy fácil de usar.	4	5	5	5	4
Creo que voy a necesitar ayuda de un técnico para poder utilizar el sistema.	1	1	1	1	1
Encontré diversas funciones en el sistema que están bien integradas	5	4	4	4	4
Pienso que hay demasiada inconsistencia en este sistema	2	2	1	1	2
Me imagino que la mayoría de la gente aprenderá a utilizar este sistema rápido.	5	5	5	5	4
Me pareció un sistema muy complicado de usar.	1	1	1	1	1
Me pareció un sistema muy complicado de usar.	1	1	1	1	3
Necesito aprender muchas cosas antes de poder usar bien el sistema.	1	2	1	1	1
1	7	5	7	6	6
2	6	6	7	7	7
3	4	5	3	3	3
4	1	2	1	1	1
5	2	2	3	2	2
6	5	4	5	5	6
7	5	5	6	6	5
8	7	6	7	7	6
9	2	2	2	1	1
10	4	6	2	3	1
11	5	5	7	5	6
12	2	2	1	1	1
13	6	6	7	7	7
14	6	6	6	6	6
15	2	2	7	5	6
16	6	6	7	7	6
17	2	3	1	2	6
18	4	4	3	3	2
19	1	3	2	1	2
20	1	6	7	7	7
21	2	3	1	1	1

22	2	4	2	4	6
23	2	1	1	1	2
24	4	5	4	2	3
25	2	4	3	2	2
26	4	3	6	5	6

Apéndice D

Link al repositorio en github

Todo el código se puede encontrar en el repositorio público de github `https://github.com/aafrecct/danubit`.

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jul 01 17:45:31 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)