Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA DE COMPUTADORES (G591)

Trabajo Fin de Grado

Desarrollo de Aplicación web para la recepción y publicación de los Informes Finales de los becarios Erasmus+ KA107 (estudiantes, profesores y administrativos) de la UAH

Autor: Sergio Manzano de la Torre

Tutor: Antonio Guerrero Baquero

2023

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA DE COMPUTADORES (G591)

Trabajo Fin de Grado

Desarrollo de Aplicación web para la recepción y publicación de los Informes Finales de los becarios Erasmus+ KA107 (estudiantes, profesores y administrativos) de la UAH

Autor: Sergio Manzano de la Torre

Tutor/es: Antonio Guerrero Baquero

TRIBUNAL:

Presidente: Javier Pedro Carracedo

Vocal 1º: Iván García Daza

Vocal 2º: Antonio Guerrero Baquero

FECHA: 07/2023

Índice

Resumen

En este trabajo se va a desarrollar una aplicación web para la cumplimentación de los informes finales de los participantes en el programa Erasmus+ KA107 de la UAH (estudiantes y personal universitario), informes que son presentados por dichos participantes una vez terminada la movilidad internacional.

El objetivo de este TGF es la implementación de un sistema que facilite el trámite y consulta posterior de dichos informes finales de una forma moderna y eficiente.

Abstract

In this work, a web application is going to be developed for the completion of the final reports of the participants in the Erasmus+ KA107 program of the UAH. These reports are presented once the international mobility is finished.

The objective of this TGF is the implementation of a system that facilitates the reports reviewing and publishing in a modern and efficient manner.

Resumen extendido

Actualmente cuando los participantes del programa Erasmus+ KA107 de la UAH terminan su estancia en el programa, deben cumplimentar un informe detallando diversos puntos de interés de su estancia. Estos informes de generan actualmente a través de plantillas de Word y se envían por correo electrónico al director del proyecto en la UAH. Posteriormente, tras su revisión y aprobación, el director se encarga de generar ficheros PDF y publicarlos en la página web (https://ka107.web.uah.es//2021-22/MainPage.pdf), con el objetivo de aportar información a futuros participantes, así como al propio proyecto para mejorarlo.

Este sistema tiene varios problemas. El primero consiste en que cada participante puede alterar la estructura de los informes, complicando la normalización y análisis de los mismos. Otro problema reside en el coste temporal y de esfuerzo que debe realizarse para la revisión de los informes y su publicación por parte del director del proyecto.

El objetivo de este trabajo consiste en el desarrollo de una aplicación web que agilice los procesos de cumplimentación y revisión de estos informes, así como facilitar su publicación y difusión al público en general.

Glosario de acrónimos y abreviaciones

MERN – Stack tecnológico para el desarrollo de aplicaciones web compuesto por: MongoDB, Express, React, Node

ORM – Object Relational Mapper

CRUD – Create, Read, Update, Delete

i18n – Internacionalización / sistema de traducciones

1.- Introducción

En este trabajo se describe el diseño y arquitectura de las diferentes partes que componen la aplicación.

A continuación, se detalla un análisis funcional con los requisitos del sistema:

- Tipos de usuario:
    - Público: Usuario sin registrar, únicamente puede consultar aquellos informes publicados como visibles.
    - Estudiante y Personal Universitario: puede consultar los informes visibles y propios, así como crear y editar sus propios informes.
    - Administrador: puede listar y consultar todos los informes (visibles y ocultos), editar la visibilidad de cualquier informe, y realizar el alta de usuarios en el sistema.
- Datos de usuario: nombre y apellidos, email (se utilizará como nombre de usuario).
- Listado de informes: Se podrán listar en pantalla los informes según los siguientes criterios/filtros:
    - Visibilidad (parámetro implícito según tipo de usuario)
    - Tipo de informe (Estudiante o Personal PDI/PAS)
    - Año académico
    - País extranjero de destino
    - Universidad de origen y destino
- Consulta/Edición de informes.
- Sistema de autenticación: Se empleará JWT (JSON Web Tokens) para la autenticación de peticiones de usuarios a la API REST.
- Sistema de informes con preguntas dinámicas: La aplicación soportara diferentes preguntas dependiendo del tipo de informe y la fecha de creación de los mismos.
- Sistema de almacenamiento de archivos: principalmente imágenes/fotos adjuntas en los informes.
- Administrador de visibilidades de informes de usuario.
- Sistema de Internalización: La aplicación estará disponible en inglés y español.
- Listado de universidades: Se almacenarán las universidades disponibles en BBDD con el fin de mejorar la normalización para la búsqueda y cumplimentación de informes.

1.2.- Modelo actual de Informe Final para estudiantes (a emular)

Universidad de Alcalá

ERASMUS+ KA107 FINAL REPORT FOR EXCHANGE STUDENTS

Co-funded by the Erasmus+ Programme of the European Union

Fill in only the White boxes. Press Ctrl+Z if you need to undo the last operation.

ACADEMIC YEAR:	20	-20	Date of submission of this report:	

DATA OF THE STUDENT:

Name (without Surname):	
Home University Name:	
Host University Name:	
Main Faculty/School in Host:	
Main Programme in Host:	
Programme coursed in Home:	

1. **GLOBAL ESTIMATION OF THE INTERNATIONAL EXCHANGE EXPERIENCE**

2. **TASKS BEFORE TRAVEL (INITIAL PAPERS AND PROCEDURES AT HOME, VISA)**

3. **ADVICE ABOUT THE TRAVEL (FLIGHT COMPANIES, PREPARATIONS…)**

4. **TASKS WHEN ARRIVING TO HOST INSTITUTION (PLACES TO GO, PAPERS, RESIDENCE PERMIT,…)**

5. **ADVICE ABOUT ACCOMMODATION (WHERE TO SEARCH, WHAT TO CARE ABOUT,…)**

6. **PRACTICAL INFORMATION ABOUT THE DESTINATION FACULTY (SOME PHOTOS IF POSSIBLE)**

7. **ADVICE ABOUT EXAMINATIONS (TYPICAL PROCEDURES, DIFFICULTY,…)**

8. **INFORMATION ABOUT THE LOCAL LANGUAGE COURSE (DURATION, TIMING, PRICE,…)**

9. **ADVICE ABOUT LOCAL TRANSPORT**

10. **ADVICE ABOUT DAILY LIFE (PLACES FOR SHOPING, LEISURE, FOOD, SOME PHOTOS...)**

11. **DATA ABOUT THE CITY (CUSTOMS, PLACES TO VISIT, NEARBY TOURISM, SOME PHOTOS...)**

12. **MONTHLY BUDGET REQUIRED (ACCOMMODATION, FOOD, TRANSPORT, OTHER,…)**

13. **OBLIGATORY: WEEKLY TIMETABLE OF YOUR CLASSES IN THE HOST UNIVERSITY**

(FOLLOW THIS MODEL)

Click HERE

14. **DATA OR ADVICE ABOUT EACH ONE OF THE COURSES ATTENDED BY THE STUDENT (DIFFICULTY LEVEL, EXAMINATIONS, TEACHERS,…)**

15. **NEGATIVE ASPECTS THAT YOU FOUND IN THIS EXPERIENCE**

This document has the purpose of informing future exchange students in UAH and may be published in UAH website.

WEEKLY CLASSES TIMETABLE

(Follow THIS MODEL)

TIME	*MONDAY*	*TUESDAY*	*WEDNESDAY*	*THURSDAY*	*FRIDAY*
8:00/ 8:55					
9:00/ 9:55					
10:00/10:55					

11:00/11:55					
12:00/12:55					
13:00/13:55					
14:00/14:55					
15:00/15:55					
16:00/16:55					
17:00/17:55					
18:00/18:55					
19:00/19:55					
20:00/20:55					

COURSES ATTENDED

COURSE		**ECTS Credits**	**Semester (1/2)**	**Teaching Language**	**PROGRAMME (Grade or Master)**	
Code	**Name**				**Code**	**Name**

TOTAL ECTS:

You can include the Erasmus Spanish Language Course of ALCALINGUA (code ALC, name Alcalingua, 6 ECTS, no Programme)

COMMENTS (optional):

If your Timetable corresponds to an existing "International Programme" coursed in UAH, say its name:

1.2.- Modelo actual de Informe Final para PDI y PAS (a emular)

Universidad de Alcalá

ERASMUS+ KA107 FINAL REPORT FOR EXCHANGE STAFF MEMBERS

Co-funded by the Erasmus+ Programme of the European Union

Fill in only the White boxes. Press Ctrl+Z if you need to undo the last operation.

ACADEMIC YEAR:	**20**		**-20**		**Date of submission of this report:**	

DATA OF THE STAFF MEMBER:

Name (without Surname):	
Home University:	
Host University:	
Faculty/Department in Host:	
Host teacher/staff name:	

Global estimation of the international exchange experience.

Please describe the activities that you have carried out:
For Teaching mobilities: dates, times, location and general contents of your teaching activity (minimum 8 hours).
For Training mobilities: dates, times, location and general contents of your training activity.
For both types: Any other type of collaboration activities realized with the host staff.

Which was your accommodation in the host city? What price? Please comment the positive and negative aspects of that accommodation.

Please outline the most valuable takeaways from your mobility week (in terms of content, prospects for further cooperation, valuable contacts established, etc.).

What are the possible areas where cooperation could continue in the future? Please mention concrete ideas/suggestions (for instance, new projects, new courses, common programmes, staff weeks, etc.). Please be specific, as this information will allow us to follow up your suggested activities and offer support for their possible implementation (at the university level, within Erasmus + project framework, etc.).

When you come back to your home university, how will you disseminate the results obtained in your mobility? With whom will you share them?

Will this mobility experience affect or modify in any way your normal work when you come back to your home university? How?

Should you have any other reflections, practical tips or content related suggestions regarding the experience you had during the staff mobility week, please share with us.

This document has the purpose of informing future exchange staff of this KA107 project and may be published in UAH website.

2.- Base teórica

Para el diseño de la aplicación se utiliza una versión modificada del stack tecnológico MERN, en concreto se realizan 2 modificaciones importantes:

- Se reemplaza el motor de base de datos MongoDB por PostgreSQL. Esto nos proporciona un mayor rendimiento a la hora de realizar operaciones CRUD y un menor consumo de memoria de almacenamiento, debido a la estructuración y normalización de las tablas presente en bases de datos relacionales (SQL) vs las no relacionales (NoSQL) a la escala de este proyecto.
- El uso de Typescript en lugar de Javascript. Typescript es un super-set de Javascript que principalmente añade un sistema de tipado en tiempo de compilación, lo cual nos permite detectar errores derivados de los lenguajes débilmente tipados. Sin embargo, esto requiere de un paso extra de "transpilación" (proceso de compilado que traduce de un leguaje a otro, en lugar de pasar a código máquina) a Javascript; y el sistema de tipado no se asegura en tiempo de ejecución.

La principal característica del stack elegido reside en la separación del frontend y el backend en 2 aplicaciones distintas:

- **Backend:** API REST desarrollada utilizando Node y Express y en una base de datos PostgreSQL.
- **Frontend:** consiste en un cliente web desarrollado utilizando React.

Esta separación nos permite ampliar los clientes independientemente de backend, pudiéndose reutilizar para diferentes clientes (aplicaciones web, integraciones, aplicaciones móviles, etc).

También nos permite escalar horizontalmente el backend, permitiéndonos aumentar la capacidad del sistema añadiendo más instancias de este, en lugar de escalar verticalmente requiriendo de un hardware con características superiores.

3.- Desarrollo del sistema

3.1-. Backend

En este apartado se explica el diseño y arquitectura del backend.

Como se explica en los apartados anteriores, el backend de la aplicación consiste en una API REST. Esto nos aporta una gran flexibilidad para implementar las diferentes operaciones que debe soportar la aplicación, así como nos permite

Esta API es responsable de la lógica de negocio de la aplicación, la validación de los datos y la interacción con los datos; los cuales serán posteriormente consumidos por el frontend. Debido a esto, en este apartado también se describe la estructura de la base de datos.

3.1.1.- Arquitectura

La arquitectura del backend se construye a partir del patrón de diseño software Modelo-Vista-Controlador (MVC).

Este patrón es el predominante a la hora de realizar aplicaciones web debido a que nos permite desacoplar el tratamiento y gestión de los datos de la presentación de los mismos.

Funcionalmente la estructura del backend se divide en:

- **entities:** Definen los modelos de datos asociados a la base de datos y son utilizados por el ORM para generar los esquemas de tablas de la base de datos.
- **routes:** Definen las operaciones de la API y sus middlewares.
- **controllers:** Implementan la lógica de las operaciones.
- managers: Responsables de la interacción con base de datos.
- middlewares: Implementan funciones de autenticación, validación y procesamiento de peticiones, que serán ejecutadas antes de las operaciones de los controladores para una ruta.
- **models:** Definen los modelos de datos en formato JSON asociados a las peticiones.
- **validators:** Definen operaciones para datos complejos y/o dependientes de base de datos.
- **requestValidators:** Definen schemas en formato JSON utilizando la librería JOI, que serán consumidos por el middleware “authenticator”.

3.1.2.- Sistemas

En este apartado se describen los diferentes sistemas que componen la aplicación backend agrupados por funcionalidad.

3.1.2.1.- Autenticación

Como por definición una API REST no guarda el estado en sesión, para la autenticación de los usuarios se utiliza JWT.

Este protocolo nos permite identificar a los usuarios mediante el uso de "tokens", los cuales consisten en cadenas de texto que codifican datos del usuario necesarios para su identificación, así como datos que nos permiten determinar la validez de los propios datos.

Para aumentar la seguridad de la aplicación, se utiliza un sistema de 2 tokens:

- **Token de refresco:** se genera cuando se hace login en la app (ruta API: /auth/login), y sirve para autenticar al usuario y generar los tokens de refresco. Este token se envía al cliente en el cuerpo de la petición y es almacenado en memoria en el cliente, teniendo una larga duración (aproximadamente un día).
- **Token de acceso:** se genera al hacer login (ruta API: /auth/login) o al refrescar cuando este expira (ruta API: /auth/refresh), y sirve como clave de un solo uso para las operaciones que requieren de un usuario autenticado. debido a la rápida expiración de este (apenas unos segundos). Este token se envía en una cookie en la petición, haciendo que solamente sea accesible por la API.

Esta aproximación nos garantiza una mayor resistencia ya que cada operación un token de acceso diferente, haciendo que sea más difícil un ataque por suplantación de identidad; salvo que se intercepte el token de refresco.

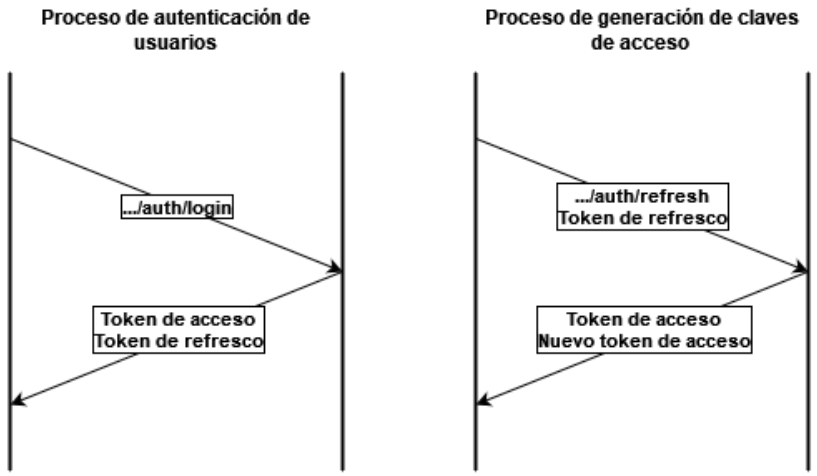


Figura 1.- Diagramas de peticiones de autenticación y generación de claves de acceso.

Para el uso de dichos tokens, según el tipo de petición deberá incluirse uno de estos, generalmente el token de acceso, en la cabecera “Authentication” definida en el protocolo HTTP siguiendo el formato “Bearer <token>”.

Para facilitar la validación de los tokens en el backend, se implementa el middleware “autheticator”.

3.1.2.2 Usuarios

Debido a los requisitos del sistema, la aplicación debe dar soporte tanto a los participantes activos del proyecto, como al público general. A continuación, se especifican los permisos de según el rol de cada usuario:

- Administrador (admin):
    - Registrar nuevos usuarios
    - Cambiar el nombre y/o apellidos de usuarios registrados
    - Cambiar su propia contraseña
    - Consultar informes públicos y ocultos
    - Edición de informes
    - Cambiar visibilidad de los informes
- Estudiantes y Personal
    - Cambiar su propia contraseña
    - Consultar informes públicos
    - Crear informes de estudiantes o personal (dependiendo del rol)
    - Edición de informes propios
- De consulta:
    - Consultar los informes públicos.

Este último usuario de consulta es un “meta-usuario”, dado que dicho rol no existe en la aplicación; usándose solamente como termino para describir las operaciones disponibles para usuarios no autenticados.

Para poder realizar la autenticación de los usuarios descrita en el apartado anterior, es necesario definir estructuras que nos permitan almacenar los datos del usuario. Para garantizar un determinado grado de privacidad, se recolectan los datos mínimos imprescindibles para la identificación de los usuarios.

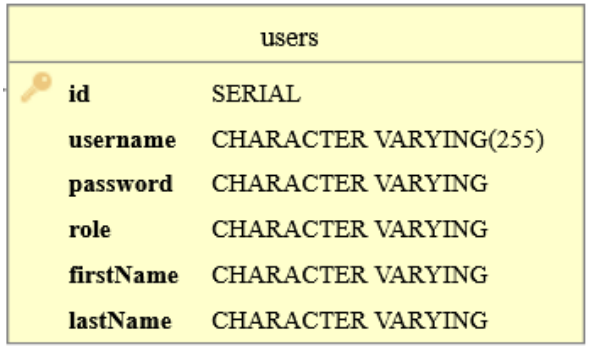


Figura 2.- Estructura de la tabla de usuarios en base de datos.

Por último, cabe destacar el almacenamiento de las contraseñas. Para ello se emplea la librería bcript, la cual nos permite encriptar y "saltear" la contraseña antes de guardarla en base de datos. Este paso de "salteado" consiste en la generación de pequeñas cadenas de texto que se mezclan con la contraseña, lo cual aumenta la seguridad haciendo que, aunque se guarde varias veces la misma contraseña, los datos guardados en base de datos sean completamente diferentes. Además, la comparación de las contraseñas en el paso de autenticación se realiza mediante los hashes de las cadenas de texto recibidas en la petición y base de datos, haciendo que una vez guardada en base de datos nunca se vuelva a convertir a texto plano.

3.1.2.3.- Internacionalización (I18N)

Este sistema es responsable dar soporte de idiomas en el cliente. Para ello se utiliza una tabla diccionario que nos permite realizar consultas de forma eficiente en función del idioma seleccionado por el usuario.

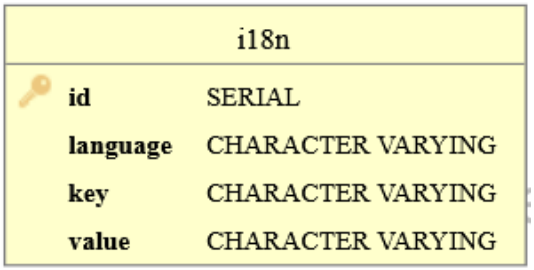


Figura 3.- Definición de tabla 1i8n en base de datos

3.1.2.4 Informes

En este apartado se explican los diferentes tipos de informes y como se modelan en la aplicación y base de datos.

Para definir los tipos de datos necesarios en base de datos, se utiliza TypeORM. Esta librería implementa un ORM que permite transformar las estructuras de datos de la aplicación en tablas de base de datos, así como métodos auxiliares para realizar la conexión con base de datos y ejecutar sentencias SQL de forma funcional en código.

Para ahorrar espacio de almacenamiento, se normalizan las tablas de base de datos agrupando los elementos comunes a ambos informes.

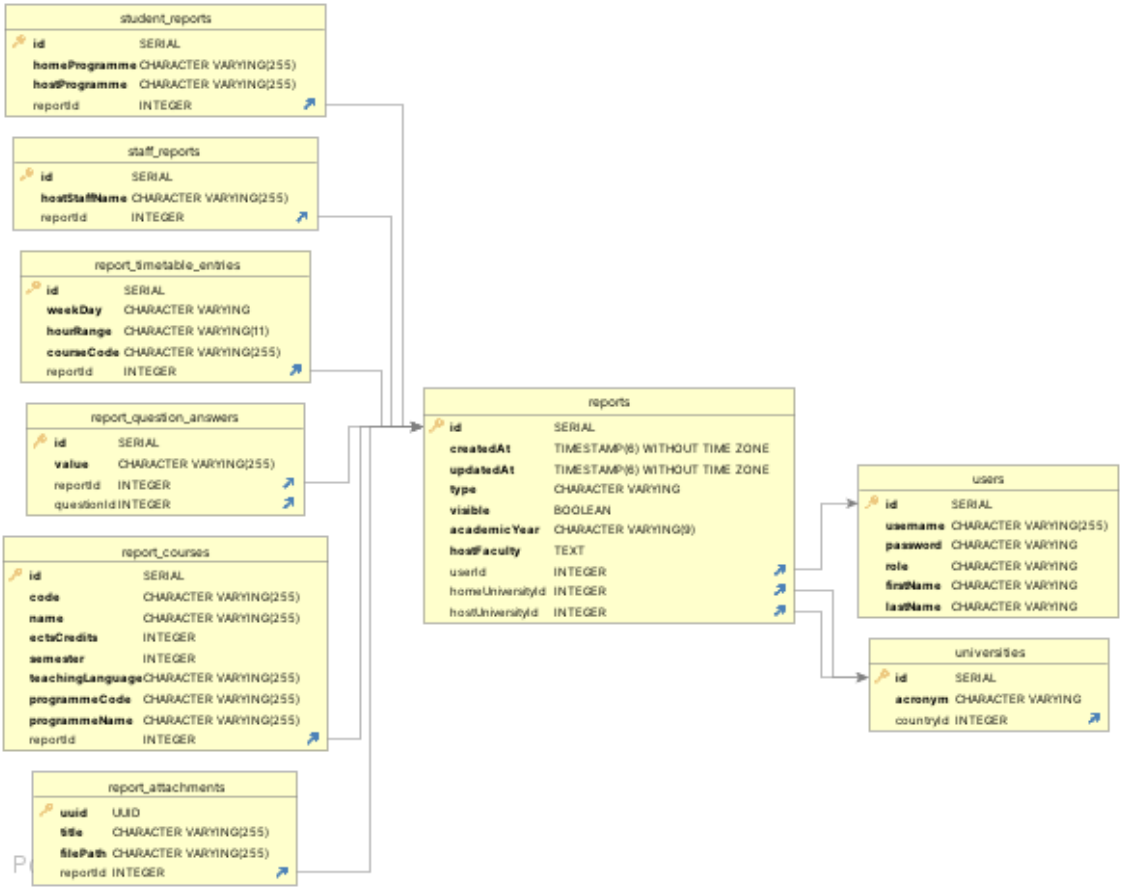


Figura 4.- Diagrama de relaciones entre las tablas de base de datos

Una ventaja utilizar Typescript durante el desarrollo de la aplicación, consiste en que nos permite definir los modelos de datos en formato JSON. Esto nos elimina el paso de transformar las diferentes estructuras de datos que maneja la API a formatos compatibles con el protocolo HTTP, aumentando el rendimiento de la aplicación y reduciendo posibles errores resultantes de la traducción. Esto último resulta en una mayor capacidad de mantenimiento a futuro.

Por último, cabe destacar la estructura de las peticiones CRUD de informes, las deben separar en 2 fases: primero una petición para los datos del informe y después, si no se produce ningún error, una petición por cada adjunto.

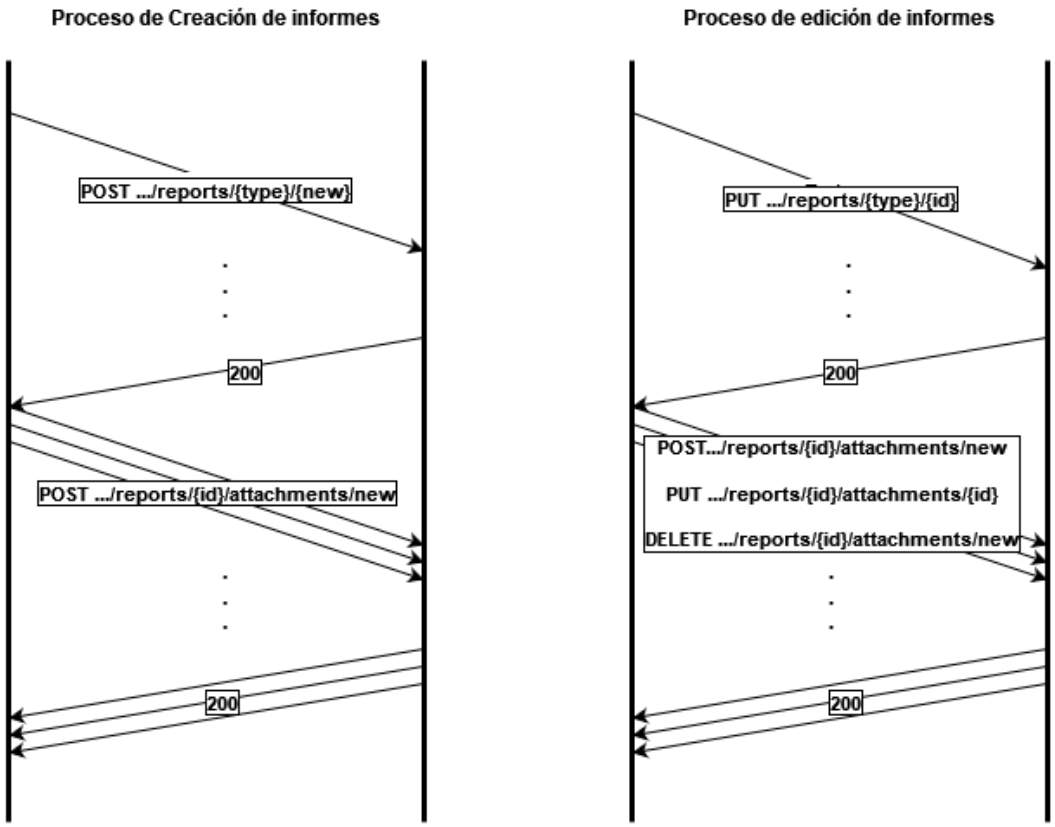


Figura 5.- Diagrama de peticiones HTTP necesarias para la creación/edición de informes.

Esto es necesario desde el punto de vista técnico, ya que, para poder enviar ficheros en las peticiones, el tipo de petición (cabecera Content-Type) debe ser multipart, y la librería utilizada para el procesamiento de este tipo de mensajes nos obliga a definir si se va a enviar ninguna, exactamente una o un array de ficheros, lo cual es incompatible con los requerimientos (los adjuntos deben ser opcionales). Además, la estructura de un mensaje multipart no nos asegura que, si enviamos múltiples adjuntos con sus correspondientes títulos, podamos discernir que título va con que fichero.

3.1.2.5 Validaciones

Por razones técnicas, la validación de los datos de las peticiones se realiza en 2 pasos:

1. Validación de la petición: se utiliza la librería JOI para definir esquemas de los argumentos que deben ir en las diferentes partes de la petición (header, body y/o query) y hacer validaciones simples de los datos (aquellas que no requieren consultas en base de datos). También se implementa un middleware que se alimenta de los esquemas anteriores y traduce los errores detectados por la librería en códigos i18n que serán consumidos por el cliente.
2. Validación de datos: es realizada por los validators y los controladores dependiendo de la complejidad de los datos. Es responsable de validar los datos dependientes de base de datos.

3.2.- Frontend

El frontend de la aplicación implementa el cliente web. Para ello se implementa una Single-Page-Application (aplicación de una sola página) utilizando React.

Esta solución nos permite crear páginas web interactivas que responden de forma parecida a una aplicación de escritorio, ya que en vez de navegar por diferentes páginas web, el contenido de esta se modifica dinámicamente en tiempo de ejecución.

Para la implementación de este cliente se utiliza React, que mediante JSX permite definir componentes HTML dinámicos y reactivos utilizando Typescript. En este punto caben destacar 2 librerías esenciales:

- **Axios:** librería que implementa un cliente HTTP mediante el uso de las promesas de JS y nos permite "hidratar" el contenido de la página mediante peticiones contra el backend, así como realizar las diferentes operaciones soportadas por la API.
- **Bootstrap:** librería de estilos CSS que nos permite tener un diseño moderno y minimalista de forma rápida y sencilla, sin la necesidad de tener conocimientos de CSS.

3.3.- Servidor

En este punto se describe la configuración del servidor físico donde se despliega la aplicación.

Por las características de la aplicación, a la hora del despliegue en producción se necesitarían de 3 puertos para la ejecución de la aplicación: uno para base de datos, otro para el cliente web y otro para la API; a lo cual hay que añadir un cuarto puerto en el cual se despliega un administrador de base de datos. Este último nos permite administrar la base de datos de forma remota para facilitar el mantenimiento de la aplicación.

Para facilitar el despliegue y el mantenimiento de los diferentes servicios, se emplea Docker y docker-compose. Esta herramienta nos permite virtualizar las diferentes aplicaciones en contenedores (como máquinas virtuales, pero consumiendo una fracción de recursos que estas), pudiéndose replicar los entornos fácilmente en diferentes máquinas; así como definir redes virtuales para interconectar los diferentes contenedores entre sí. Esto último nos permite controlar que contenedores están expuestos al exterior.

Por último, para poder acceder de forma más sencilla a las diferentes partes de la aplicación y exponer los puertos de las diferentes aplicaciones a Internet, se utiliza NGINX. Esta herramienta nos permite crear un proxy reverso para redireccionar los diferentes puertos a través de rutas HTTP, facilitando la experiencia de los usuarios.

4.- Conclusiones y trabajo futuro

Para mejorar la seguridad de la aplicación, habría que generar un certificado SSL del dominio de la aplicación para poder utilizar HTTPS, lo cual incrementaría considerablemente la seguridad en el momento de autenticarse en la aplicación.

Una posible ampliación a futuro sería migrar el cliente web de React a React-Native; lo cual permitiría la creación de aplicaciones móviles para Android y IOS utilizando Flutter (https://flutter.dev/), y de escritorio mediante Electron (https://www.electronjs.org/es/).

Otra posibilidad que brinda la arquitectura de la aplicación consiste en el desarrollo de integraciones con otras aplicaciones. Una integración que convendría estudiar sería la creación de un cliente en Blackboard que se hidrate utilizando las operaciones que brinda la API.

5.- Bibliografía

Tecnologías backend:

- Node: https://nodejs.org/es
- Express: https://expressjs.com/
- PostgreSQL: https://www.postgresql.org/
- TypeORM: https://typeorm.io/
- JWT: https:/jwt.io/
- bcrypt: https://www.npmjs.com/package/bcrypt
- JOI: https://joi.dev/

Tecnologías frontend

- Vite: https://vitejs.dev/
- React: https://es.react.dev/
- Bootstrap: https://getbootstrap.com/
- Axios: https://axios-http.com/es/docs/intro

Servidor:

- Docker: https://www.docker.com/
- NGINX: https://www.nginx.com/
- Hetzner: https://www.hetzner.com/

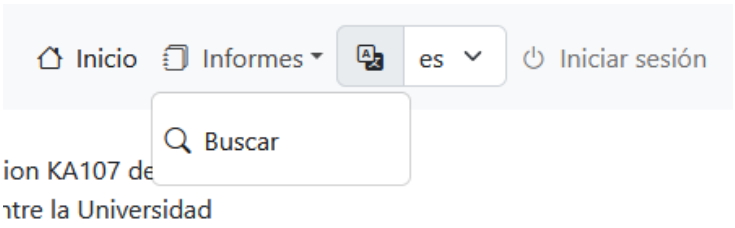
Consultas:

- Stackoverflow: https://stackoverflow.com/
- Youtube: https://www.youtube.com/
- Reddit: https://www.reddit.com/

ANEXO I - Manual de usuario

Búsqueda de informes

Accesible para usuarios registrados y el público general desde el menú “Informes” de la barra de navegación.



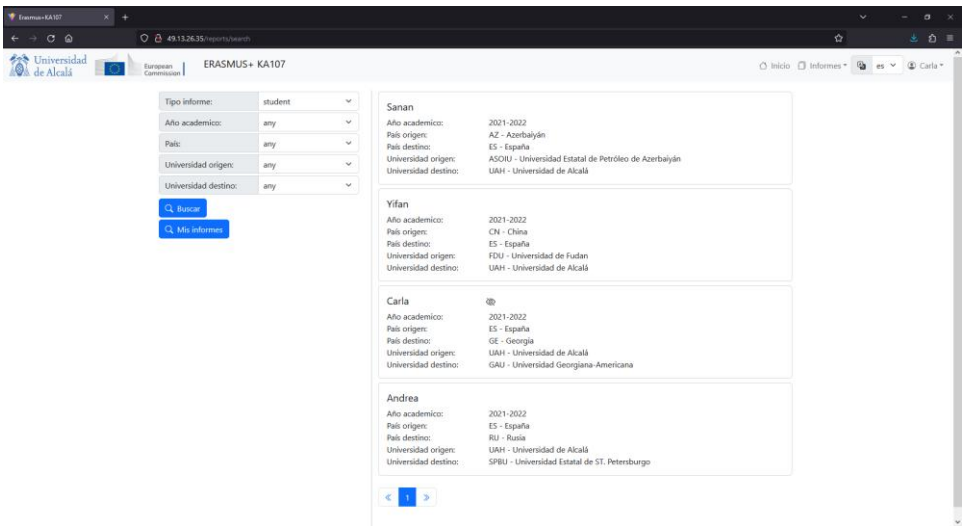
Dispone de los siguientes filtros para afinar las consultas por:

- Tipo de informe: estudiante o personal universitario
- Año académico
- País: origen y/o destino
- Universidad origen
- Universidad destino

En los resultados se muestra la siguiente información de los informes:

- Nombre de usuario sin apellidos: al pinchar sobre este se redirige a la página del informe para su consulta, edición o gestión.
    - Para usuarios registrados como administrador o dueños de los informes se mostrará un icono al lado para indicar que el informe esta oculto al público general.
- País de origen y destino
- Universidad de origen y destino

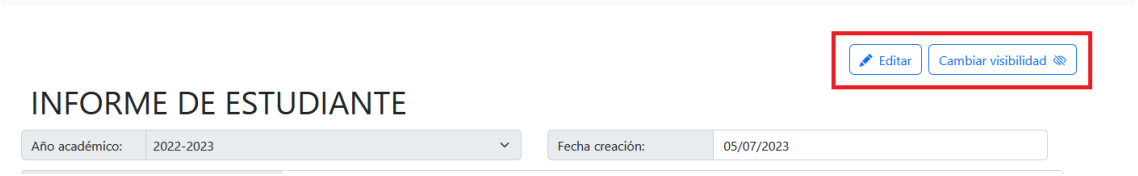
Adicionalmente, los usuarios registrados como estudiante y personal disponen de un botón de búsqueda adicional: “Mis Informes”, el cual únicamente muestra los informes creados por dicho usuario.



Modificación y gestión de informes

Tras la búsqueda de un informe, los dueños de los informes y los administradores tienen disponibles controles adiciones que permiten la modificación y gestión de estos.

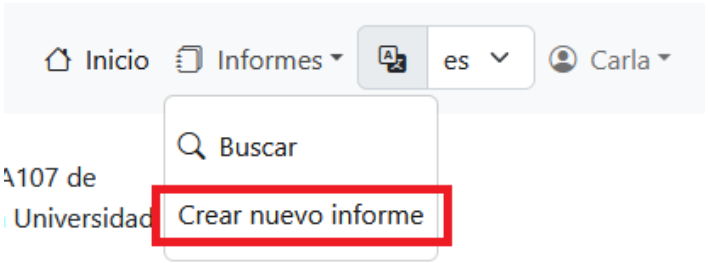
- Editar: este botón está disponible para administradores y los dueños de los informes.
    - **NOTA:** Actualizar los datos se ocultará el informe para el público general hasta que lo publique un administrador tras su revisión y aprobación.
- Cambiar visibilidad: este botón está disponible solo para los administradores y permite mostrar u ocultar un informe para el público general.
    - Al lado del texto aparece un botón que muestra la visibilidad actual del informe.



Cumplimentación de informes

Para la entrega de los informes finales, los usuarios de tipo estudiante y personal universitario deben pichar en la opción “Crear nuevo informe” del menú de “Informes” de la barra de navegación.

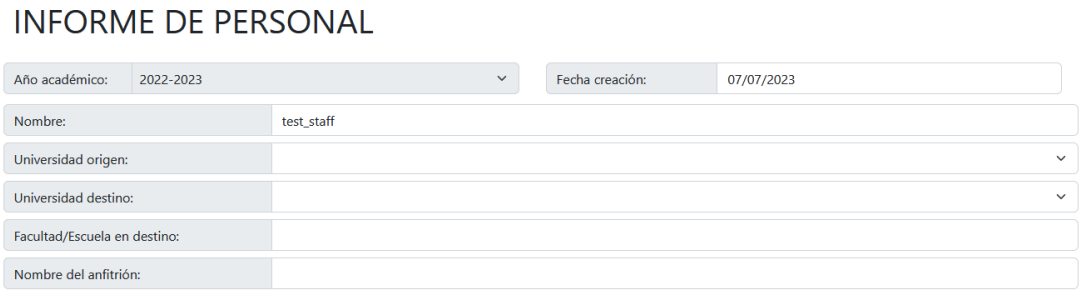
- **NOTA:** Esta opción solamente estará disponible tras haberse autenticado en la aplicación para usuarios de tipo estudiantes y personal universitario.



Tras esto, será redirigido a la página de cumplimentación de informes finales específica para su tipo de usuario.

En esta página deberá rellenar todos los datos del formulario. Los cuales se dividen en:

- Información del usuario
    - **NOTA:** El campo “Nombre” no puede ser modificado. Si detecta algún error contacte con algún administrador para que modifique dicho dato. (Ver apartado “Administración”).



- Preguntas acerca de la estancia, actividades realizadas, experiencia del intercambio,
    - **NOTA:** Es obligatorio contestar a todas las preguntas.

Adicionalmente los estudiantes se deben rellenar 2 secciones más: cursos y horarios. Para más información acerca del funcionamiento de esta sección consultar los apartados “Cursos” y “Horario”.

Por último, al final de la página existe una sección opcional en la que se pueden adjuntar imágenes con un título explicativo.

Animamos a los participantes a que suban fotos de su estancia: alogamiento, instalaciones de las universidades, sitios de ocio y turismo, etc.

- **NOTA:** Una vez enviado el informe, solamente se puede modificar el título de las fotos. Si se quiere reemplazar, se tendrá que eliminar la foto y añadir otra.

Una vez se hayan cumplimentado todas las secciones obligatorias del informe, y comprobado que todos los datos sean correctos, podrá guardar los cambios pulsando en el botón “Crear” o “Actualizar” dependiendo de si se esta creando un nuevo informe o editando uno ya existente.

- **IMPORTANTE:** Asegúrese de haber introducido todos los datos correctamente antes de pulsar los botones de “Crear” / “Actualizar” para evitar errores y/o pérdidas de datos.

Cursos

- Para añadir una entrada, introduzca los datos en sus campos correspondientes y piche en “Añadir”. Si desea “Cancelar” el proceso pulse el botón cancelar para limpiar los campos.

- Para editar una entrada, seleccione el botón “Editar” de la fila correspondiente. En el cuadro superior se podrán editar los campos deseados. Para guardar los cambios pulse el botón “Actualizar”. Si desea cancelar la acción pulse el botón “Cancelar”.

Para eliminar una entrada, seleccione el botón “Eliminar” de la fila correspondiente.

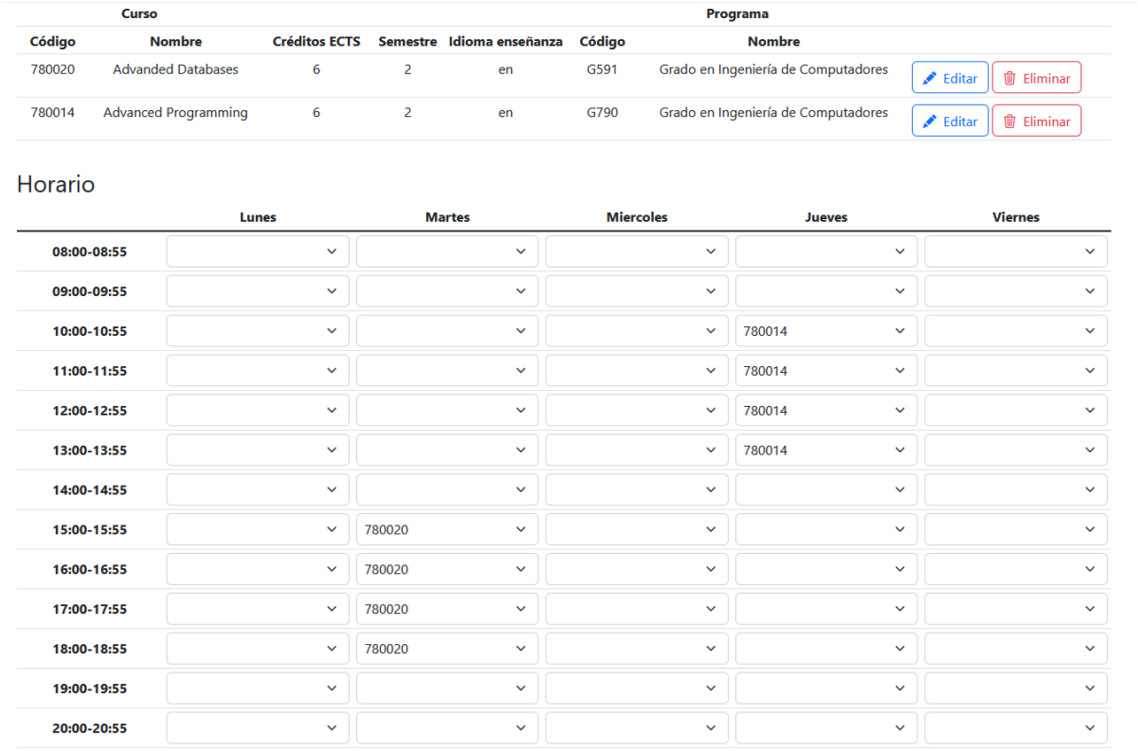
- **IMPORTANTE:** Asegúrese de querer realizar dicha opción ya que es irreversible.

Horario

Para añadir una nueva entrada o editar una ya existente, debe seleccionar el desplegable correspondiente al día y franja horaria correspondientes y seleccionar el código del curso correspondiente.

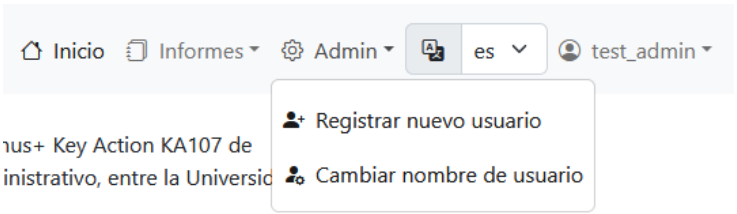
Tenga en cuenta que solamente se podrá asignar un curso por franja horaria y día.

Es obligatorio introducir al menos una entrada.



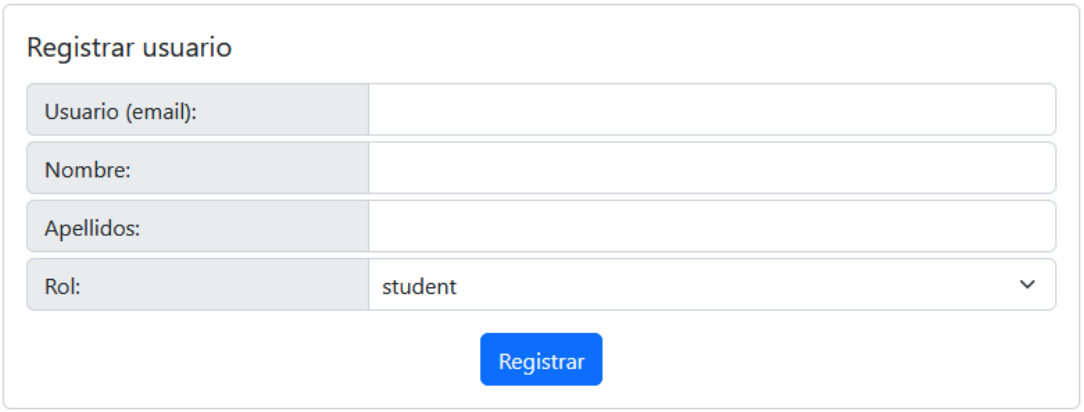
Administración

Los usuarios registrados como administrador disponen de un menú adicional para realizar tareas de administración: “Admin”.



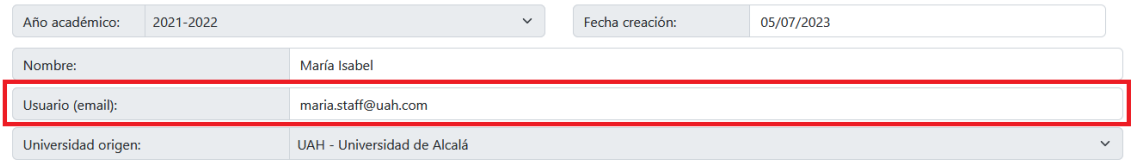
Desde este menú se pueden acceder a los siguientes formularios:

- Registro de usuarios
    - Desde esta pantalla se pueden dar de alta nuevos usuarios en el sistema.



- Cambio de datos de usuario
    - Aquí se pueden modificar el nombre y apellidos de los usuarios.

Consejo: Los usuarios se identifican por el su correo electrónico. Si duda de cual corresponde al usuario, se puede consultar debajo del nombre en cualquier informe cumplimentado por el usuario.



ANEXO II – Base de datos

En este apartado se va a profundizar en la estructura de la base de datos utilizando extractos de código de los modelos de datos y las entities de base de datos.

Sistema de preguntas

Con el paso de tiempo los informes se van modificando para incluir y mejorar la información relevante de las experiencias vividas por los participantes del programa. Esto se refleja principalmente en las preguntas de presentes en los formularios.

Debido a esto es necesario la implementación de un sistema que nos permita modificar las preguntas sin alterar los informes previos.

Para ello las preguntas se modelan utilizando 3 datos fundamentales:

- **code:** clave i18n que permite la localización de las preguntas a los diferentes idiomas soportados por la aplicación.
- **reportType:** clave que identifica el tipo de informe a la que pertenece la pregunta.
- **active:** flag que indica si la pregunta debe incluirse en los informes que se cumplimenten en el momento de la consulta.

```
import {
    Entity,
    PrimaryGeneratedColumn, Column
} from "typeorm";

@Entity("report_questions")
export class ReportQuestionEntity {

    @PrimaryGeneratedColumn()
    id: number;

    @Column({ length: 255, unique: true })
    code: string;

    @Column({ length: 255 })
    reportType: string;

    @Column()
    active: boolean;

}
```

Figura 1.- Modelado en código de la tabla report_questions

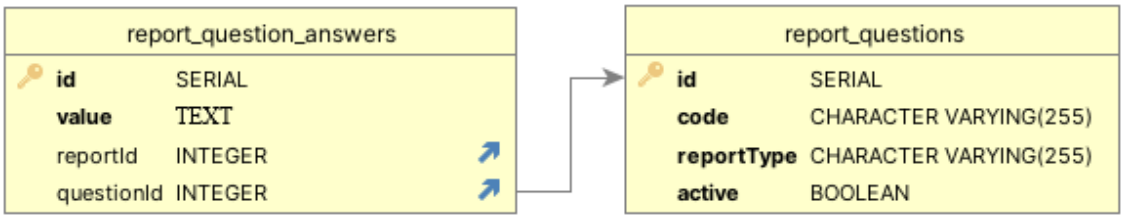


Figura 2.- Relación entre las tablas asociadas a las preguntas de los informes

En las figuras anteriores se puede observar la presencia de un campo adicional: "id". Si se aplica la normalización de la tabla este campo es redundante, ya que el campo "code" ya identifica de forma única las preguntas. Sin embargo, esto es un artefacto técnico resultante del ORM empleado (TypeORM); ya que, para poder añadir de forma dinámica nuevas entradas, la librería requiere de un valor autonumérico para asegurar que no se generen duplicidades en las "primary keys".

Sistema de adjuntos de informes

Citando al refranero español: *"Una imagen vale más que mil palabras".* Aunque los usuarios de la aplicación contesten a las preguntas del informe de forma exhaustiva, aquellos informes que aportan fotos tomadas durante el intercambio aportan más información a los futuros participantes y animan a muchos otros a vivir la experiencia.

Debido a esto, un requerimiento fundamental de la aplicación consiste en la inclusión de un sistema que nos permita adjuntar imágenes en los informes. Para ello el sistema debe ser capaz de mandar y almacenar imágenes entre la API y el cliente web. Esto se consigue mediante el uso de la librería Multer junto a la base de datos en 2 fases.

En la primera fase se utiliza la librería Multer para realizar el procesado de peticiones HTTP de tipo "multipart" (necesarias para el envío de imágenes y texto) y posteriormente guardarlas en el sistema de archivos del servidor.

Tras esto, se guardan en base de datos el ID del informe, la ruta del fichero en el sistema y un UUID que actúa como identificador para poder recuperar la imagen en el cliente.

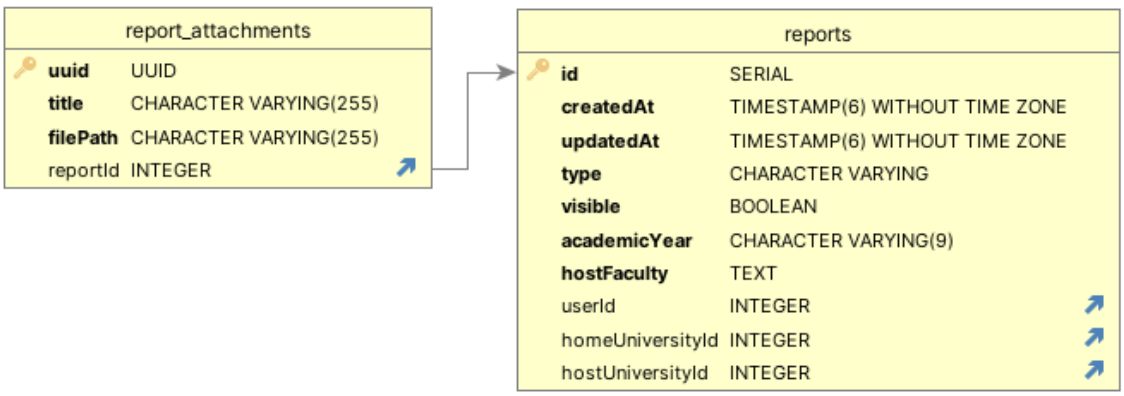


Figura 3.- Relación entre las tablas asociadas al sistema de adjuntos

```
import {
    Entity,
    PrimaryGeneratedColumn, Column,
    JoinColumn, ManyToOne, PrimaryColumn
} from "typeorm";
import { ReportEntity } from "./ReportEntity";

@Entity("report_attachments")
export class ReportAttachmentEntity {

    @PrimaryColumn({type: "uuid"})
    uuid: string;

    @Column({ length: 255 })
    title: string;

    @Column({ length: 255, nullable: false })
    filePath: string;


    @ManyToOne(() => ReportEntity, (report) => report.id)
    @JoinColumn()
    report: ReportEntity;


}
```

Figura 4.- Modelado en código de la tabla "report_attachments"

La mayor ventaja que presenta este sistema consiste en que, aunque en la actualidad solamente se admiten imágenes en formato JPG, JPEG, PNG y SVN; en un futuro se podría implementar soporte para otro tipo de archivos con un coste de esfuerzo bajo.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR