TRABAJO FIN DE GRADO

GRADO DE INGENIERÍA INFORMÁTICA

Diseño y Desarrollo de una Aplicación Web para la Gestión de Actividades y Asociaciones Erasmus

Autor
Cristian Fernández Jiménez

Tutor
Juan Manuel Fernández Luna

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

—

Granada, Septiembre de 2022

Diseño y Desarrollo de una Aplicación Web para la Gestión de Actividades y Asociaciones Erasmus.

Autor
Cristian Fernández Jiménez

Tutor
Juan Manuel Fernández Luna

Diseño y Desarrollo de una Aplicación Web para la Gestión de Actividades y Asociaciones Erasmus

Cristian Fernández Jiménez

Palabras clave: Erasmus, Asociación, Estudiante, Evento, Administrador, Laravel,...

Resumen

Durante el periodo de intercambio de cientos de miles de estudiantes, desde el momento que arriban en destino, buscan relacionarse de una manera internacional con otros estudiantes y personas. En este aspecto, suelen entrar en juego diversas asociaciones Erasmus, algunas comunes por toda Europa y otras particulares de la ciudad.

Sin embargo, y como he podido descubrir de primera mano, la publicación de actividades se hace en todos los casos de una manera distinta, mayoritariamente con el uso de redes sociales. Además, la inscripción a la asociación no se puede realizar y manejar de una manera directa desde el mismo canal. Igualmente para la participación en eventos.

El objetivo principal del siguiente proyecto es pues, desarrollar una aplicación web que mejore la experiencia de estudiantes Erasmus a la hora de consultar las diferentes actividades que ocurren en su destino.

Como resultado, se obtendrá una aplicación web totalmente operativa en la que estudiantes y asociaciones Erasmus contarán con un software centralizado donde publicar y consultar eventos en sus ciudades de intercambio. Permitiendo a su vez la participación en dichas actividades, la inscripción a asociaciones y, por consiguiente, la completa gestión por parte de asociaciones y estudiantes Erasmus.

El proceso de desarrollo consta de diversas fases, presentes de forma general en trabajos de desarrollo software: análisis y planificación de requisitos, diseño, implementación y evaluación. Todas ellas llevadas a cabo con un ciclo de vida de desarrollo en cascada en el periodo de un curso académico y con el cual culmino mi etapa universitaria en el Grado de Ingeniería Informática en la Universidad de Granada.

Design and Development of a Web App for Erasmus Activities and Associations Management

Cristian Fernández Jiménez

Key words: Erasmus, Association, Student, Event, Admin, Laravel,...

Abstract

Hundreds of thousands of students go in exchange every year and, since they arrive to their destination, look for connecting people in an international aspect. For this reason, different Erasmus associations show up, some of them common all around Europe or specific from a particular city.

However, and as I found out first-hand, activities are posted differently by every single one of the associations, mainly using social-networks. In addition, subscriptions are not handled directly from these channels, neither are event participations.

The main goal of the present project is to develop a web app to ease Erasmus students' experiences, whenever they would like to know what is happening in their destinations.

As a result, a completely working web app will be deployed, where students and Erasmus associations have a centralized software to post and see activities in their destination cities. Besides, users will be able to participate in events, subscribe to associations and, on top of that, completely manage these in the same place.

The development process consists of various phases, usually present in any software development project: analysis and planning of requirements, design, implementation and evaluation. All of them carried out on a waterfall model, throughout a whole academic year, meaning the end of my Bachelor's Degree in Computer Engineering for the Univerisity of Granada.

Yo, **Cristian Fernández Jiménez**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 20623057S, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Cristian Fernández Jiménez

Granada a 6 de septiembre de 2022 .

D. **Juan Manuel Fernández Luna**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Diseño y Desarrollo de una Aplicación Web para la Gestión de Eventos y Asociaciones Erasmus***, ha sido realizado bajo su supervisión por **Cristian Fernández Jiménez**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 6 de septiembre de 2022.

El director:

Juan Manuel Fernández Luna

Agradecimientos:

En la realización del presente proyecto ha tomado una gran labor un maravilloso grupo de personas. No solo de forma directa, pero también indirecta. Aquellos que considero pilares fundamentales en mi vida, que comprenden y respetan lo que soy y se suben a mi barco con todo lo que tenga en mente para mi futuro.

Por ello, en primer lugar, quisiera dedicar este proyecto a mi familia, a mis padres Manuel y Mari Carmen, que siempre confiaron en mí y nunca han dudado de aquello que soy capaz de lograr. A mi hermano Fran, de quien me siento orgulloso de seguir sus pasos, pues siempre ha sido un mentor para mí, entregado y consejero en cada paso importante de mi carrera. A pesar de la distancia que nos ha separado a todos desde hace años, no me he podido sentir más arropado por su cariño y sé que se mantendrá intacto por los años venideros.

A aquel precioso grupo de almas que he encontrado en Roma, que han hecho de este el mejor año de mi vida. Siendo esenciales en este periodo de altibajos lejos de mi zona de confort. Gracias a su empatía, su forma de afrontar los problemas, su mente abierta y su sonrisa de oreja a oreja cada vez que los veía, han hecho que evolucione drásticamente como ser humano.

A mis amigos de casa, los que se interesan por mis quehaceres allá donde esté, desde mi infancia hasta hoy dispuestos y volcados en mis logros cada día que pasa.

Quisiera agradecer también a mi tutor, Juan Manuel Fernández Luna. Sin su asesoramiento y apoyo constante desde el día uno no habría emprendido un camino tan llevadero y apacible como ha resultado ser.

Por último, pero no por ello menos importante, quiero agradecer a todos los profesores que he tenido, por haber fomentado, aunque algunos más que otros, el desarrollo de mi curiosidad. A los compañeros y amigos de Granada, con los que he pasado buenos ratos tanto en clase como fuera de ella, de los cuales me siento feliz de llevarme de esta bonita etapa.

A todos ellos, mil gracias.

Acknowledgments:

Carrying out this project wouldn't have been the same without a marvelous group of people. Not only directly, but also indirectly. Those who I consider main pillars in my life, and understand and respect what I am.

Thus, in the first place, I would like to dedicate this project to my family, my parents Manuel and Mari Carmen, who always trusted on me and never doubted what I am capable of. To my brother Fran, who I feel proud of following his steps, as he always has been a mentor to me, devoted and counselor to every major step of my career. Despite the distance that separated us for years, I could not have felt more supported by their love and I am sure that it will remain intact in the years to come.

To that lovely group of souls that I found in Rome, who have made of this year the best of my life. Essential in this period of ups and downs, away from my comfort zone. Thanks to their empathy, their way of approaching problems, their open minds and an ear-to-ear smile every single time I met them. I have evolved drastically as a human being because of them.

To those friends in my hometown, thoughtful to my duties wherever I am. From our childhood until today, dedicated to my achievements every day that goes by.

I would like to express my gratitude as well to my tutor Juan Manuel Fernández Luna. Without his advice and support from day one I could not have set out this bearable and pleasant path.

Last but not least, I would like to thank all the teachers that I have had, for encouraging the development of my curiosity. To those classmates and friends from Granada, who I have spent the best of times with, outside and inside the university, for who I feel happy to borrow from this gorgeous phase.

To all of them, a thousand thanks.

Índice general

Resumen I

Autorización de publicación III

Autorización de defensa IV

Agradecimientos V

Índice general VI

Acrónimos X

1. Introducción 1
- 1.1. Motivación . . . 1
- 1.2. Descripción del problema y solución propuesta . . . 1
- 1.3. Objetivos . . . 2
- 1.4. Descripción de la memoria . . . 2

2. Planificación y presupuesto 3
- 2.1. Planificación . . . 3
  - 2.1.1. Etapas del proyecto . . . 3
  - 2.1.2. Planificación temporal . . . 4
- 2.2. Presupuesto . . . 5
  - 2.2.1. Recursos hardware . . . 5
  - 2.2.2. Recursos software . . . 5
  - 2.2.3. Recursos humanos . . . 6

3. Análisis 7
- 3.1. Entidades principales . . . 7
  - 3.1.1. Estudiantes Erasmus . . . 7
  - 3.1.2. Asociaciones . . . 7
  - 3.1.3. Eventos . . . 7
  - 3.1.4. Administración del sistema . . . 7
- 3.2. Requisitos de información . . . 8
- 3.3. Requisitos funcionales . . . 12
- 3.4. Requisitos de datos . . . 23
- 3.5. Requisitos no funcionales . . . 27
- 3.6. Casos de uso . . . 28
- 3.7. Diagramas de Caso de Uso . . . 51

3.7.1. Gestión de usuarios 51
3.7.2. Gestión de asociaciones 52
3.7.3. Gestión de inscripciones 53
3.7.4. Gestión de eventos 54
3.8. Diagramas de secuencia 55

4. Diseño **72**
4.1. Diseño de la base de datos 72
4.1.1. Diagrama Entidad-Relación 72
4.1.2. Paso a tablas 73
4.2. Diseño visual del sistema 78
4.2.1. Guidelines 78
4.2.2. Prototipado 79
4.3. Diseño arquitectónico del sistema 98

5. Implementación **100**
5.1. Tecnologías y herramientas a usar 100
5.2. Diagrama de clases UML 102
5.2.1. Paquete Controllers 102
5.2.2. Paquete App 103
5.2.3. Paquete Views 103
5.3. Descripción de clases y módulos 106
5.3.1. Models 106
5.3.2. Controllers 109
5.3.3. Views 111
5.3.4. Roles y funciones 113

6. Pruebas **115**
6.1. Tests de caja blanca 115
6.1.1. Controlador de Login 115
6.1.2. Controlador de Registro 118
6.1.3. Controlador de Administradores de destinos 120
6.1.4. Controlador de Asociaciones Erasmus 122
6.1.5. Controlador de Administradores por destino 127
6.1.6. Controlador de Estudiantes Erasmus 130
6.2. Tests de caja negra 135
6.2.1. Administradores de destinos 135
6.2.2. Asociaciones Erasmus 136
6.2.3. Administradores por destino 145
6.2.4. Usuarios Estudiantes Erasmus 147
6.2.5. Criterio responsive 155
6.3. Conclusiones tras las pruebas 159

7. Despliegue **160**
7.1. Hosting seleccionado 160
7.2. Subida de la aplicación 160
7.3. Link de acceso a la aplicación y código 160

8. Conclusiones, disciplinas adquiridas y líneas futuras **161**
8.1. Conclusiones 161
8.2. Disciplinas adquiridas 163
8.3. Líneas futuras 163

A. Anexo I: Manual de Usuario **164**
A.1. Introducción 164
A.2. Instalación 164
A.3. Acceso 164
A.4. Utilización 166
A.4.1. Administrador de destinos 166
A.4.2. Administrador por destino 167
A.4.3. Asociación Erasmus 169
A.4.4. Estudiante Erasmus 173

Bibliografía **179**

Acrónimos

RI Requisito de Información

SHA Secure Hash Algorithm

BD Base de Datos

RF Requisito Funcional

GIF Graphics Interchange Format

RD Requisito de Datos

RNF Requisito No Funcional

CU Caso de Uso

CP Clave Primaria

CC Clave Candidata

CE Clave Externa

FN Forma Normal

FNBC Forma Normal de Boyce-Codd

IU Interfaz de Usuario

MVC Modelo Vista Controlador

HTML HyperText Markup Language

CSS Cascading Style Sheets

SASS Syntactically Awesome Style Sheets

AJAX Asynchronous JavaScript And XML

XML Extensible Markup Language

PHP Hypertext Preprocessor

SQL Structured Query Language

FTP File Transfer Protocol

UML Unified Modeling Language

WAE Web Application Extensions

CRUD CREATE, READ, UPDATE, DELETE

URL Uniform Resource Locator

SSL Secure Sockets Layer

API Application Programming Interface

Capítulo 1

Introducción

1.1. Motivación

El periodo en el que he realizado mi Proyecto de Fin de Grado ha coincidido con mi estancia Erasmus en la ciudad de Roma. Cuando toqué tierra en la capital italiana, a la hora de comenzar a ganar confianza con la ciudad y crear relaciones internacionales pensé en visitar cualquier actividad de asociaciones Erasmus. Sin embargo, estas eran bastantes y todas ellas comunicaban sus eventos en plataformas distintas, como Facebook, Whatsapp o Instagram. A su vez cada asociación debería gestionar todas sus redes constantemente, en cada ocasión que ideasen una nueva actividad. Es por ello que una plataforma web que ayude a asociaciones y estudiantes, a centralizar la información en un mismo lugar, resultaría beneficiosa para ambos, en cuanto a comodidad y sencilla visibilidad. Así surge la idea del presente proyecto, que detallará el análisis, diseño e implementación de una web que cumpla estas expectativas, y sea lanzada al público a su finalización.

1.2. Descripción del problema y solución propuesta

Los inconvenientes producidos por la descentralización de asociaciones y eventos hace dificultosa la experiencia Erasmus. El estudiante debe estar pendiente de varios canales de información y redes sociales para planificar su semana en destino.

Un software que reúna la información de todas las asociaciones disponibles en un destino, junto a los eventos que lancen, será un gran beneficio para ambos estudiantes y asociaciones.

Las especificaciones de esta aplicación se centran principalmente en la gestión de eventos y asociaciones, realizados por los miembros de esta, y permitiendo que estudiantes Erasmus formen parte fácilmente y puedan conocer lo que ocurre en su destino de intercambio.

Principalmente esta aplicación se encargará de:

- Gestión de asociaciones Erasmus.
- Creación y publicación de actividades.

Por otra parte, si el usuario es estudiante:

- Subscripción a asociaciones Erasmus.
- Participación en sus eventos.

- Visionado interactivo de los eventos próximos en destino.

La experiencia erasmus quedaría unificada en un mismo lugar, simple y atractivo para los usuarios.

1.3. Objetivos

Me gustaría subrayar los objetivos que tengo intención de lograr con el presente proyecto:

Del Trabajo de Fin de Grado

1. Conseguir un software completamente operativo que cumpla con lo especificado.
2. Desplegar el proyecto públicamente, para que pueda ser usado.
3. Lograr superar aspectos de seguridad mínimos, para el correcto uso de la aplicación.
4. Adquirir una interfaz visual atractiva, que se traduzca en la positiva recepción de la web.

Académicos

1. Demostrar los conocimientos y habilidades adquiridas en el Grado de Ingeniería Informática.
2. Conseguir trabajar en un ambiente cercano a la ingeniería, desarrollando un proyecto de mayor escala.
3. Mejorar mis conocimientos y aprender el uso de herramientas para el desarrollo web.
4. Superar con éxito la tesis y con ello, el Grado de Ingeniería Informática.

1.4. Descripción de la memoria

En esta memoria puede encontrase toda la información necesaria para comprender cómo fue el proceso de desarrollo del proyecto y como está definida, construida y puesta a prueba la aplicación.

El proyecto consta de diversas fases, presentes de forma general en trabajos de desarrollo software: análisis y planificación de requisitos, diseño, implementación y evaluación. En la primera de las partes se especifican las funcionalidades con las que va a contar el software desarrollado. Seguidamente, se idea un diseño general del sistema, en cuestión de almacenamiento de datos, arquitectura e interfaz. En el último proceso son seleccionadas las tecnologías necesarias para crear aquello diseñado y se lleva a cabo, para finalizar con la puesta a prueba y verificación de requisitos de aquello desarrollado.

Capítulo 2

Planificación y presupuesto

En la presente sección se incluye una estimación del coste que supone el desarrollo del proyecto, así como la planificación temporal del mismo.

En la planificación, se comenta el ciclo de vida del proyecto y las etapas que lo conforman, junto a su planificación temporal.

El presupuesto por otra parte señala los recursos hardware, software y humanos, seguido al coste desglosado del proyecto.

2.1. Planificación

En el desarrollo de este proyecto el modelo de ciclo de vida del software seguido es el **modelo en cascada**. En este las diferentes etapas del proyecto funcionarán una detrás de la otra de manera lineal, pues solo cuando una fase finalice se podrá continuar con la posterior. Comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento. (Pressman & Maxim, 1998)

2.1.1. Etapas del proyecto

Las etapas que se llevarán a cabo en la evolución del proyecto son:

- **Análisis** software de la aplicación a desarrollar, identificando sus componentes, requisitos e interacción entre sí. Se trata de el proceso de reunión de requisitos, donde comprenderemos completamente la naturaleza del programa a construir, para así desarrollar la aplicación.

- **Diseño** software. Este se refiere al establecimiento de las estructuras de datos, la arquitectura general del software y la representación de interfaces.

- **Implementación software** a partir de lo modelado en la fase de diseño. Esta actividad consiste en traducir el diseño en una forma legible por la máquina, de forma que genere los comportamientos de la aplicación web deseados.

- **Evaluación del sistema**, realizando pruebas sistemáticas para comprobar el correcto funcionamiento de aquello implementado. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todo el código se ha comprobado, y en los procesos externos funcionales, es decir, la realización de las pruebas para la detección de errores.

- **Despliegue** del software, donde se pondrá en funcionamiento y comprobará los posibles problemas que surjan. Esto se realizará subiendo la página web a internet, contando con un dominio y hosting online, e importando la base de datos.
- **Documentación** del proceso y toda la información en este documento. Realizada a la par de cada fase de este proyecto.

2.1.2. Planificación temporal

A continuación, se muestra una tabla que sintetiza las etapas del proyecto junto a sus fechas de inicio y fin deseadas.

Cronología	Octubre				Noviembre				Diciembre				Enero				Febrero				Marzo			
Semana	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Análisis																								
Diseño																								
Implementación																								
Pruebas																								
Despliegue																								
Documentación																								

Tabla 2.1: Planificación temporal del proyecto

Cronología	Abril				Mayo				Junio				Julio				Agosto			
Semana	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Análisis																				
Diseño																				
Implementación																				
Pruebas																				
Despliegue																				
Documentación																				

Tabla 2.2: Planificación temporal del proyecto. Continuación.

La extensión ideal del proyecto es la de un curso académico completo, dividiendo cada fase según la carga de trabajo de cada una. La documentación se actualizará durante todo el proceso.

2.2. Presupuesto

2.2.1. Recursos hardware

Descripción	Unidades	Precio unitario (€)
Ordenador Personal	1 (3 años de vida útil)	1000
Total		**250**[1]

Tabla 2.3: Desglose de recursos hardware

[1] Siguiendo la planificación temporal del proyecto y suponiendo que la vida útil de un computador personal es de 3 años, viendo que la utilización del ordenador supondrá un total de 9 meses, el gasto por su uso asciende a los 250€.

2.2.2. Recursos software

Los recursos software son aquellos necesarios y orientados al desarrollo web.

Descripción	Unidades	Precio unitario (€)
Ubuntu 20	-	0
Visual Studio Code	-	0
Bootstrap 5.1.3	-	0
Laravel Framework 8.83.12	-	0
Google Maps API	3 meses	0
Mozilla Firefox	-	0
Google Chrome	-	0
GitHub	-	0
Hosting LucusHost	3 meses	8.35
Total		**8.35**

Tabla 2.4: Desglose de recursos software

Partiendo del sistema operativo **Ubuntu** del ordenador personal, la codificación se hará en **Visual Studio Code** con la ayuda de los frameworks **Bootstrap** y **Laravel**, que nos permitirán acelerar el proceso de generación de la web con una interfaz atractiva, **Laravel** además permite desplegar un servidor local para su desarrollo. En la aplicación se dará uso de la **API de Google Maps** para permitir la localización de eventos y la modificación interactiva por parte de las asociaciones.

Seguidamente los navegadores **Firefox** y **Chrome** se utilizarán para la realización en tiempo real de pruebas y ajustar el aspecto responsive.

GitHub a su vez nos permitirá subir el código a internet de una forma privada, para así garantizar el control de versiones y asegurar una forma de realizar copias de seguridad.

Por último el hosting **LucusHost** será el seleccionado para desplegar la aplicación final, pues tiene un coste gratuíto durante 3 meses, siendo lo único necesario a pagar la asignación del dominio.

2.2.3. Recursos humanos

Los labores de dirección técnica y realización han sido efectuadas por el autor del proyecto. Se estima la dedicación a estas laboras en diez horas semanales aproximadamente desde el inicio al fin del proyecto.

Desglosando por horas y función las labores realizadas, y usando como referencia los salarios medios para cada profesión (Talent.com, s.f.); el presupuesto teórico en mano de obra sería el siguiente:

Persona	Horas	Sueldo por hora (€)
Analista software	160	15,38
Diseñador web	100	12,44
Desarrollador web	130	11,79
Software tester	60	17,95
Total	450	**6315,50**

Tabla 2.5: Desglose de recursos humanos

Capítulo 3

Análisis

3.1. Entidades principales

3.1.1. Estudiantes Erasmus

Para cada estudiante Erasmus, el sistema almacenará diversos campos necesarios para su **identificación**. Además, tendrán asignado un destino Erasmus y universidad, dentro de los disponibles en la aplicación al momento de registro. El sistema permitirá al usuario **consultar** su información y **modificarla**. Además, cada usuario tiene la opción de **solicitar inscribirse** a las diferentes asociaciones de su destino, que podrá encontrar en su feed o realizando una búsqueda. Sin olvidar que se podrá **dar de baja** de la aplicación en cualquier momento desde su cuenta. Formando parte de una asociación además podrá **participar en eventos**.

3.1.2. Asociaciones

Existirá la opción de **dar de alta** asociaciones en la pantalla inicial. El alta requerirá alguna información de identificación, junto a motivación de la asociación y el destino donde opera. Se debe elegir además si se permite la **inscripción automática** de usuarios o **requiere de verificación**. El usuario que crea la asociación será el administrador de esta, pudiendo **consultar** y **modificar** esta información. Además de poder **crear, modificar o borrar eventos** y **recibir inscripciones** de estudiantes. A su vez se comunicará con la administración del sistema por medio de **notificaciones**.

3.1.3. Eventos

Desde la cuenta de usuario de una asociación, será posible **crear eventos**. Su información será básica para la descripción de la actividad a desarrollar. Asimismo han de ser de un tipo predefinido (Deportivos, viajes, tandems, fiestas, tours, noches y otros). Asociados además a una localización en un destino predeterminado.

3.1.4. Administración del sistema

Existirán usuarios especiales que administren el sistema. En primer lugar existirá un tipo de usuario que se encargue de **añadir destinos** a la aplicación, vinculando a un usuario con el destino en cuestión para su gestión. Estos últimos dispondrán de un acceso normal a

la aplicación, con **vista general de asociaciones por destino** y la opción de **consultarlas** independientemente de su privacidad. Además, con autoridad para **enviarles notificaciones** a los usuarios gestores con sugerencias o cambios necesarios que no respeten la política. Además, la **creación de universidades** será llevada a cabo por este administrador.

3.2. Requisitos de información

Esta serie de requisitos determinan la información obtenida y producida por el sistema para su almacenamiento.

RI-00: Requisito de información para Usuario

Usuario		
RI-00	*Datos de un usuario*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres(21)	Nick único del estudiante
Contraseña	SHA	Se almacenará en forma de hash y salteada
Tipo	Unsigned int	Rol de usuario que cumple

RI-01: Requisito de información para Estudiante Erasmus

Estudiante Erasmus		
RI-01	*Datos de un usuario estudiante*	
Atributo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres(40)	Nombre de pila del estudiante
Apellidos	Cadena de caracteres(80)	Apellidos del estudiante
Usuario	Cadena de caracteres(21)	Nick único del estudiante
Correo electrónico	Cadena de caracteres(255)	Email del estudiante
Universidad destino	Cadena de caracteres (12)	Identificador de la universidad de acogida del estudiante
Imagen de perfil	Cadena de caracteres (255)	Ruta de imagen para el perfil de usuario

RI-02: Requisito de información para Asociación Erasmus

Asociación Erasmus		
RI-02	*Datos de una asociación Erasmus*	
Atributo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres (40)	Nombre completo de la asociación
Acrónimo	Cadena de caracteres (5)	Siglas de la asociación
Código identificación	Cadena de caracteres (21)	Acrónimo+Destino
Correo electrónico	Cadena de caracteres (255)	Email del administrador
Descripción	Cadena de caracteres (5000)	Motivación y detalles presentados por la asociación
Destino	Unsigned int	Referencia a la ciudad donde el administrador es delegado
Imagen de perfil	Cadena de caracteres (255)	Ruta de la imagen para el perfil de asociación
Imagen de portada	Cadena de caracteres (255)	Ruta de la imagen para el banner de la asociación
Abierta	Bit	Por defecto 1. Requerida solicitud si valor 0.
Pública	Bit	Por defecto 1. Privada con valor 0.
Estado	Bit	Por defecto 1 (ACTIVA). Suspendida si valor 0.

RI-03: Requisito de información para Evento Erasmus

Evento Erasmus		
RI-03	*Datos de un evento Erasmus*	
Atributo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres (50)	Nombre asignado al evento en cuestión
Descripción	Cadena de caracteres (500)	Breve resumen de la realización de este
Código identificación	Unsigned int	Identificación del evento
Código de asociación	Cadena de caracteres (21)	Asociación organizadora
Imagen del evento	Cadena de caracteres (40)	Ruta de la imagen para el evento
Fecha	Date	Día y hora en el que iniciará el evento
Localizacion	Unsigned int	Referencia a la localización del evento
Tipo de evento	Cadena de caracteres (10)	Tipo de actividad a celebrar: DEPORTES, FIESTAS, NOCHES, TANDEMS, VIAJES, TOURS, OTROS
Número de participantes	Unsigned int	Por defecto 0
Coste	Decimal	Por defecto 0. La moneda será la del destino

RI-04: Requisito de información para Inscripción en asociación Erasmus

Solicitud inscripción asociación Erasmus		
RI-04	*Datos de una solicitud*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres (15)	Nick único del estudiante
Código asociación	Cadena de caracteres (21)	Identificación de la asociación
Fecha	Date	Fecha de solicitud
Aceptada	Bit	Por defecto 0. Aceptada valor 1

RI-05: Requisito de información para Administrador del sistema

Usuario administrador del sistema		
RI-05	*Datos de un usuario administrador del sistema*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres (15)	Nick único del admin
Destino	Unsigned int	Referencia a la ciudad donde el administrador es delegado

RI-06: Requisito de información para Notificación a asociación Erasmus

Notificación asociación Erasmus		
RI-06	*Datos de una notificación de un administrador*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres (15)	Nick único del administrador
Código asociación	Cadena de caracteres (21)	Identificación de la asociación
Fecha	Date	Fecha de la notificación
Estado	Bit	Por defecto 0. Aceptada valor 1
Descripción	Cadena de caracteres (500)	Comentario de los cambios o sugerencias
Desactivación	Bit	Desactivación de asociación necesaria

RI-07: Requisito de información para Partipación a evento Erasmus

Partipación a evento Erasmus		
RI-07	*Datos de una participación de un usuario en un evento*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres (15)	Nick único del estudiante
Código asociación	Cadena de caracteres (21)	Identificación de la asociación
Código evento	Unsigned int	Identificación numérica del evento

RI-08: Requisito de información para Destino Erasmus

Destino Erasmus		
RI-08	*Datos de un Destino Erasmus*	
Atributo	**Tipo**	**Descripción**
Id	Unsigned int	Identificador único del destino
Ciudad	Cadena de caracteres (15)	Nombre de la ciudad destino
País	Cadena de caracteres (15)	Nombre del país destino

RI-09: Requisito de información para Universidad Erasmus

Universidad Erasmus		
RI-09	*Datos de un Destino Erasmus*	
Atributo	**Tipo**	**Descripción**
Id	Cadena de caracteres (12)	Código ERASMUS único de la universidad de acogida del estudiante
Nombre	Cadena de caracteres (255)	Nombre de la universidad destino
Destino	Unsigned int	Referencia a la ciudad donde el administrador es delegado

RI-10: Requisito de información para Localización de evento

Localización		
RI-10	*Datos de una localización de un evento*	
Atributo	**Tipo**	**Descripción**
Id	Unsigned int	Identificador único de la ubicación
Nombre	Cadena de caracteres (255)	Nombre de la ubicación
Dirección	Cadena de caracteres (255)	Dirección completa de la ubicación
Longitud	Decimal	Longitud del punto en el globo de la localización
Latitud	Decimal	Latitud del punto en el globo de la localización

3.3. Requisitos funcionales

RF-01: Requisito funcional para Alta de usuario Erasmus

Requisito Funcional	*Alta de usuario Erasmus*	RF-01
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-00, RI-01	
Precondiciones		
Postcondiciones	En caso de éxito, el nuevo usuario será almacenado en la BD. En caso contrario, no ocurrirá.	
Referencias		
Propósito	Dar de alta un nuevo usuario estudiante en la aplicación	
Descripción	Un estudiante introduce los datos de registro y es dado de alta en la BD del sistema. Si la operación se realiza con éxito, se inicia sesión y aparece la página principal, en caso contrario un mensaje de rechazo con el error correspondiente	

RF-02: Requisito funcional para Identificación de usuarios

Requisito Funcional	*Identificación de usuario*	RF-02
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-00	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema.	
Postcondiciones	En caso de éxito, el usuario podrá acceder a la aplicación y será redirigido a la pantalla principal con la sesión iniciada.	
Referencias		
Propósito	Identificarse como un usuario para acceder a las funciones de la aplicación.	
Descripción	Un usuario introduce su email/usuario y su contraseña para acceder a la aplicación. Si la operación se realiza con éxito será redirigido a la pantalla principal dependiendo de su rol, en caso contrario un mensaje de rechazo aparecerá y permanecerá en la página de Inicio de sesión.	

RF-03: Requisito funcional para Cierre de sesión de usuarios

Requisito Funcional	*Cierre de sesión de usuario*	RF-03
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-00	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada.	
Postcondiciones	En caso de éxito, el usuario cerrará sesión y será redirigido a la pantalla principal sin identificar.	
Referencias		
Propósito	Cerrar sesión como un usuario.	
Descripción	Un usuario clica sobre el icono de cierre de sesión para abandonar la aplicación. Si la operación se realiza con éxito será redirigido a la pantalla principal.	

RF-04: Requisito funcional para Consulta de usuarios Erasmus

Requisito Funcional	*Consulta de usuario Erasmus*	RF-04
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-01	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada.	
Postcondiciones		
Referencias		
Propósito	Consultar los datos de un usuario estudiante.	
Descripción	Un usuario puede consultar su propio perfil o el de otros.	

RF-05: Requisito funcional para Modificar el perfil de usuarios Erasmus

Requisito Funcional	*Modificiar usuario Erasmus*	RF-05
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-01	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada.	
Postcondiciones	En caso de éxito, los datos de la BD del usuario se verán modificados.	
Referencias		
Propósito	Modificar los datos de un usuario	
Descripción	Desde el perfil de usuario, un estudiante podrá modificar sus datos, para que el sistema realice el correspondiente cambio en la BD. Si se realiza con éxito emite por pantalla un mensaje de éxito y los datos se observan modificados. En caso contrario aparecerá un mensaje con el error.	

RF-06: Requisito funcional para Dar de baja el perfil de usuarios Erasmus

Requisito Funcional	*Dar de baja usuario Erasmus*	RF-06
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-00, RI-01	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada.	
Postcondiciones	En caso de éxito, el usuario se eliminará de la BD.	
Referencias		
Propósito	Dar de baja a un usuario del sistema	
Descripción	Desde el perfil de usuario, un estudiante podrá eliminar su cuenta, para que el sistema realice la eliminación en la BD. Si se realiza con éxito emite por pantalla un mensaje de éxito y cierra sesión. En caso contrario aparecerá un mensaje con el error.	

RF-07: Requisito funcional para Inscripciones en asociaciones Erasmus

Requisito Funcional	*Inscripción en asociación Erasmus*	RF-07
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-01, RI-02, RI-04	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada. La asociación debe estar previamente almacenada en la BD y ha de consultarse si requiere de solicitud.	
Postcondiciones	En caso de requerir solicitud y no estar creada, se generará una solicitud de inscripción y será almacenada en la BD. En caso de no requerirla, se registrará al usuario automáticamente con la inscripción en estado aceptada. En caso de existir y haber sido aceptada, se modificará su estado a aceptada en la BD. En caso de existir y haber sido denegada, se finalizará el proceso.	
Referencias		
Propósito	Registrar usuario a asociación erasmuus	
Descripción	Desde el perfil de la asociación Erasmus, un estudiante podrá solicitar su inscripción a estar, para que el sistema realice la generación de la solicitud en la BD (si es necesario). Si se realiza con éxito emite por pantalla unmensaje con lo realizado (Solicitud ó inscripción automática). En caso contrario aparecerá un mensaje con el error.	

RF-08: Requisito funcional para Responder solicitudes de inscripción en asociaciones Erasmus

Requisito Funcional	*Responder solicitud inscripción en asociación Erasmus*	RF-08
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-01, RI-02, RI-04	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada. La asociación debe estar previamente almacenada en la BD.	
Postcondiciones	En caso de éxito, se responderá una solicitud de inscripción y será modificada en la BD.	
Referencias	RF-07	
Propósito	Aceptar o rechazar inscripción a asociación erasmuus	
Descripción	Desde el perfil de la asociación Erasmus, un administrador podrá observar las solicitudes de inscripción. Se modificará la BD según la decisión tomada, y se volverá al RF-06. En caso contrario aparecerá un mensaje con el error.	

RF-09: Requisito funcional para Eliminar la inscripción de un usuario Erasmus con una asociación

Requisito Funcional	*Eliminar inscripción de usuario a asociación*	RF-09
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-01, RI-02	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada. La asociación debe estar previamente almacenada en la BD y el usuario pertenecer a ella.	
Postcondiciones	En caso de éxito, el usuario dejará de pertenecer a la asociación, provocando los cambios correspondientes en la base de datos.	
Referencias		
Propósito	Dar de baja a un usuario de una asociación	
Descripción	Desde el perfil de asociación, un estudiante podrá eliminar su inscripción,para que el sistema realice la eliminación en la base de datos de la solicitud de inscripción aceptada. Si se realiza con éxito emite por pantalla un mensaje de éxito y vuelve a la página principal. En caso contrario aparecerá un mensaje con el error.	

RF-10: Requisito funcional para Inscribirse a un evento Erasmus

Requisito Funcional	*Participar en evento Erasmus*	RF-10
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-01, RI-03, RI-07	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada. El evento debe estar almacenado en la BD del sistema.	
Postcondiciones	En caso de éxito, el usuario quedará inscrito al evento seleccionado.	
Referencias		
Propósito	Registrar usuario a evento de asociación Erasmus	
Descripción	Desde el perfil de la asociación Erasmus ó el feed de la página, un estudiante podrá consultar e inscribirse a un evento. Si se realiza, emite por pantalla un mensaje de éxito. En caso contrario aparecerá un mensaje con el error.	

RF-11: Requisito funcional para Eliminar la participación de un usuario en un evento

Requisito Funcional	*Eliminar participación de usuario en evento*	RF-11
Actores	Sistema, usuario Erasmus	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-01, RI-03, RI-07	
Precondiciones	El usuario debe estar dado de alta en la BD del sistema. La sesión de usuario debe estar iniciada. El evento debe estar almacenado en la BD del sistema y el usuario participar en este.	
Postcondiciones	En caso de éxito, el usuario dejará de pertenecer al evento, provocando los cambios correspondientes en la base de datos.	
Referencias		
Propósito	Dar de baja a un usuario de un evento.	
Descripción	Desde la pantalla del evento, un estudiante podrá eliminar su participación, para que el sistema realice la eliminación en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito y vuelve a la página principal. En caso contrario aparecerá un mensaje con el error.	

RF-12: Requisito funcional para Alta de asociación Erasmus

Requisito Funcional	*Dar de alta asociación Erasmus*	RF-12
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02	
Precondiciones		
Postcondiciones	En caso de éxito, la nueva asociación será almacenada en la BD. En caso contrario, no ocurrirá.	
Referencias		
Propósito	Dar de alta una nuevo asociación en la aplicación	
Descripción	Un administrador introduce los datos de registro y es dado de alta en la base de datos del sistema. Si la operación se realiza con éxito, se inicia sesión y aparece la página principal de la asociación, en caso contrario un mensaje de rechazo con el error correspondiente.	

RF-13: Requisito funcional para Consulta de asociación Erasmus

Requisito Funcional	*Consultar asociación Erasmus*	RF-13
Actores	Sistema, usuario	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones		
Referencias		
Propósito	Consultar los datos de un asociación Erasmus.	
Descripción	Un usuario entra en el perfil de una asociación para observar sus datos y eventos.	

RF-14: Requisito funcional para Modificar asociación Erasmus

Requisito Funcional	*Modificar asociación Erasmus*	RF-14
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, los datos de la base de datos de la asociación se verán modificados.	
Referencias		
Propósito	Modificar los datos de una asociación	
Descripción	Desde el perfil de una asociación, un administrador podrá modificar sus datos, para que el sistema realice el correspondiente cambio en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito y los datos se observan modificados. En caso contrario aparecerá un mensaje con el error.	

RF-15: Requisito funcional para Crear eventos Erasmus

Requisito Funcional	*Crear evento Erasmus*	RF-15
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02, RI-03, RI-10	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, se creará un evento relacionado con la asociación	
Referencias	RF-18	
Propósito	Crear un evento Erasmus relacionado con la asociación.	
Descripción	Desde el perfil de asociación, un administrador podrá crear un evento, para que el sistema lo relacione con esta en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito y muestra el evento creado. En caso contrario aparecerá un mensaje con el error.	

RF-16: Requisito funcional para Consultar eventos Erasmus

Requisito Funcional	*Consultar evento Erasmus*	RF-16
Actores	Sistema, usuario	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02, RI-03, RI-10	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada. El evento debe estar almacenado en la BD	
Postcondiciones	En caso de éxito, se podrá consultar un evento relacionado con una asociación	
Referencias		
Propósito	Consultar los datos un evento Erasmus.	
Descripción	Desde la página de eventos de una asociación, un usuario podrá clicar sobre un evento, para que su información sea mostrada por pantalla.	

RF-17: Requisito funcional para Consultar listado de evento Erasmus

Requisito Funcional	*Consultar listado de evento Erasmus*	RF-17
Actores	Sistema, usuario	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02, RI-03	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, se podrá consultar el listado de eventos actuales relacionados con una asociación	
Referencias		
Propósito	Consultar el listado de eventos Erasmus relacionados con la asociación.	
Descripción	Un usuario puede consultar el listado completo de eventos de una asociación,desde una página dedicada para ello. En caso de no tener eventos, estará vacía.	

RF-18: Requisito funcional para Seleccionar dirección del evento mediante Google Maps

Requisito Funcional	*Seleccionar dirección del evento mediante Google Maps*	RF-18
Actores	Sistema, administrador asociación	
Tipo	Secundario, Deseable	
Requisitos de información	RI-02, RI-03, RI-10	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada. El evento debe estar almacenado en la base de datos.	
Postcondiciones	En caso de éxito, se asignará una dirección correcta al evento.	
Referencias		
Propósito	Seleccionar desde la herramienta de Google Maps una dirección precisa para el evento.	
Descripción	Desde la página de creación de eventos de la asociación, un administrador podrá seleccionar una dirección desde la herramienta de Google Maps, para que el sistema lo asigne en la base de datos	

RF-19: Requisito funcional para Editar eventos Erasmus

Requisito Funcional	*Editar evento Erasmus*	RF-19
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02, RI-03, RI-10	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada. El evento debe estar almacenado en la base de datos.	
Postcondiciones	En caso de éxito, se modificará un evento relacionado con la asociación	
Referencias	RF-18	
Propósito	Editar un evento Erasmus relacionado con la asociación.	
Descripción	Desde la página de eventos de la asociación, un administrador podrá editar un evento, para que el sistema lo modifique en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito y muestra el evento modificado. En caso contrario aparecerá un mensaje con el error.	

RF-20: Requisito funcional para Eliminar eventos Erasmus

Requisito Funcional	*Eliminar evento Erasmus*	RF-20
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02, RI-03, RI-10	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada. El evento debe estar almacenado en la base de datos.	
Postcondiciones	En caso de éxito, se eliminará un evento relacionado con la asociación	
Referencias		
Propósito	Eliminar un evento Erasmus relacionado con la asociación.	
Descripción	Desde la página de eventos de la asociación, un administrador podrá eliminar un evento, para que el sistema lo borre en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito y vuelve al perfil de asociación. En caso contrario aparecerá un mensaje con el error.	

RF-21: Requisito funcional para Dar de baja a usuario de asociación

Requisito Funcional	*Dar de baja a usuario de asociación*	RF-21
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02, RI-03	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, se eliminará la inscripción de un usuario a la asociación	
Referencias		
Propósito	Dar de baja a usuario Erasmus relacionado con la asociación.	
Descripción	Desde la página de usuarios de la asociación, un administrador podrá expulsar al usuario, para que el sistema borre su inscripción en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito y vuelve al perfil de asociación. En caso contrario aparecerá un mensaje con el error.	

RF-22: Requisito funcional para Consultar listado de usuarios pertenecientes a una asociación

Requisito Funcional	*Consultar listado de usuarios pertenecientes a una asociación*	RF-22
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02, RI-03	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, se podrá consultar el listado de usuarios actuales inscritos a una asociación	
Referencias	RF-03, RF-17	
Propósito	Consultar el listado de usuarios inscritos a la asociación.	
Descripción	Un administrador podrá consultar la lista de usuarios pertenecientes a la asociación, y tendrá la opción de consultar su información.	

RF-23: Requisito funcional para Responder notificación sobre cambios a realizar en la asociación

Requisito Funcional	*Responder notificación sobre cambios a realizar*	RF-23
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02,RI-05 ,RI-06	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, se modificará el estado de una solicitud en la base de datos dirigida a la asociación.	
Referencias		
Propósito	Resolver la notificación recibida por el sistema	
Descripción	Desde el perfil de la asociación, un administrador podrá observar sus notificaciones de cambios, donde podrá finalizarlas tras haberlas consultado y se realice el cambio en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito. En caso contrario aparecerá un mensaje con el error.	

RF-24: Requisito funcional para Búsqueda de evento ó asociación

Requisito Funcional	*Búsqueda de evento ó asociación*	RF-24
Actores	Sistema, usuario	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-03, RI-04, RI-10	
Precondiciones	El evento debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, aparecerá una lista de eventos y asociaciones según la búsqueda.	
Referencias		
Propósito	Buscar un evento ó una asociación Erasmus según diferentes filtros.	
Descripción	Desde la barra de navegación existirá una opción de búsqueda. Se introducirá un texto a buscar y se podrán aplicar distintos filtros (asociación, ubicación, fecha, tipo de evento, destino). Si se encuentran coincidencias aparecerán divididas en Asociaciones y Eventos en forma de lista. En caso contrario, estas listas estarán vacías.	

RF-25: Requisito funcional para Consultar eventos en calendario

Requisito Funcional	*Consulta de eventos en calendario*	RF-25
Actores	Sistema, usuario	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-01, RI-03	
Precondiciones		
Postcondiciones		
Referencias		
Propósito	Explorar un calendario interactivo dónde estén contenidos los eventos inscritos.	
Descripción	Desde la sección de usuario Mis Eventos, aparecerá un calendario interactivo con los eventos señalados en sus correspondientes fechas, permitiendo la interacción y redirección con ellos. En caso de no existir eventos, el calendario no tendrá fechas señaladas.	

RF-26: Requisito funcional para Eliminar asociaciones Erasmus

Requisito Funcional	*Eliminar asociación Erasmus*	RF-26
Actores	Sistema, administrador asociación	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, la asociación se eliminará de la base de datos.	
Referencias		
Propósito	Dar de baja a una asociación del sistema	
Descripción	Desde el perfil de la asociación, un administrador podrá eliminar esta, para que el sistema realice la eliminación en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito. En caso contrario aparecerá un mensaje con el error.	

RF-27: Requisito funcional para Cambiar el estado de asociaciones Erasmus

Requisito Funcional	*Cambiar estado asociación Erasmus*	RF-27
Actores	Sistema, administrador del sistema	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02, RI-05, RI-06	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, el estado de la asociación se modificará en la base de datos.	
Referencias		
Propósito	Suspender o activar una asociación en la plataforma.	
Descripción	Desde el perfil de la asociación, un administrador podrá desactivarla o activarla, para que el sistema realice la correspondiente modificación en la base de datos. Si se realiza con éxito emite por pantalla un mensaje de éxito. En caso contrario aparecerá un mensaje con el error.	

RF-28: Requisito funcional para Solicitar cambios a asociaciones

Requisito Funcional	*Solicitar cambios a asociación*	RF-28
Actores	Sistema, administrador del sistema	
Tipo	Primario, Obligatorio	
Requisitos de información	RI-02,RI-05 ,RI-06	
Precondiciones	La asociación debe estar previamente almacenada en la BD. El usuario debe estar identificado correctamente y con la sesión iniciada.	
Postcondiciones	En caso de éxito, se generará una solicitud en la base de datos dirigida a la asociación.	
Referencias		
Propósito	Generar una solicitud de cambio dirigida a una asociación.	
Descripción	Desde el perfil de la asociación, un administrador podrá enviar propuestas de cambios, para que el sistema realice la correspondiente generación en la base de datos y sea recibida por el administrador de la asociación. Si se realiza con éxito emite por pantalla un mensaje de éxito. En caso contrario aparecerá un mensaje con el error.	

3.4. Requisitos de datos

Esta serie de requisitos representan la información introducida por el usuario para el procesamiento de los datos en el sistema.

RD-00: Datos para identificación de Usuario

Usuario		
RD-00	*Datos necesarios para la identificación de un usuario*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres(21)	Identificador único del usuario
Contraseña	Cadena de caracteres (255)	Contraseña de inicio de sesión del usuario

RD-01: Datos de un Estudiante Erasmus

Estudiante Erasmus		
RD-01	*Datos necesarios para el alta de un usuario estudiante*	
Atributo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres(40)	Nombre de pila del estudiante
Apellidos	Cadena de caracteres(80)	Apellidos del estudiante
Usuario	Cadena de caracteres(21)	Nick único del estudiante
Contraseña	Cadena de caracteres(255)	Se almacenará en forma de hash y salteada
Correo electrónico	Cadena de caracteres(255)	Email del estudiante
Destino	Cadena de caracteres (15)	Nombre de la ciudad destino
Universidad	Cadena de caracteres (255)	Nombre de la universidad de acogida del estudiante
Imagen de perfil	File	Imagen para el perfil de usuario

RD-02: Datos de una Asociación Erasmus

Asociación Erasmus		
RD-02	*Datos necesarios para el alta de una asociación Erasmus*	
Atributo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres (40)	Nombre completo de la asociación
Acrónimo	Cadena de caracteres (5)	Siglas de la asociación
Contraseña	Cadena de caracteres (255)	Se almacenará en forma de hash y salteada
Correo electrónico	Cadena de caracteres (255)	Email del administrador
Descripción	Cadena de caracteres (5000)	Motivación y detalles presentados por la asociación
Destino	Cadena de caracteres (15)	Nombre de la ciudad destino
Imagen de perfil	File	Imagen para el perfil de asociación
Imagen de portada	File	Imagen para el banner de la asociación
Abierta	Bit	Por defecto 1. Requerida solicitud si valor 0.
Pública	Bit	Por defecto 1. Privada con valor 0.
Estado	Bit	Por defecto 1 (ACTIVA). Inactiva si valor 0.

RD-03: Datos de un Evento Erasmus

Evento Erasmus		
RD-03	*Datos necesarios para el alta de un evento Erasmus*	
Atributo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres (50)	Nombre asignado al evento en cuestión
Descripción	Cadena de caracteres (500)	Breve resumen de la realización de este
Imagen del evento	File	Imagen para el evento
Fecha	Date	Día y hora en el que iniciará el evento
Localizacion	Cadena de caracteres (255)	Dirección donde se celebra el evento
Tipo de evento	Cadena de caracteres (10)	Tipo de actividad a celebrar: DEPORTES, FIESTAS, NOCHES, TANDEMS, VIAJES, TOURS, OTROS
Número de participantes	Unsigned int	Por defecto 0
Coste	Decimal	Por defecto 0. La moneda será la del destino

RD-04: Datos de un Administrador del sistema

Usuario administrador del sistema		
RD-04	*Datos necesarios para el alta de un usuario administrador del sistema*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres (15)	Nick único del admin
Contraseña	Cadena de caracteres (255)	Se almacenará en forma de hash y salteada
Destino	Cadena de caracteres (15)	Nombre de la ciudad destino

RD-05: Datos de un destino Erasmus

Destino Erasmus		
RD-05	*Datos necesarios para el alta de un destino erasmus*	
Atributo	**Tipo**	**Descripción**
Ciudad	Cadena de caracteres (15)	Nombre de la ciudad destino
País	Cadena de caracteres (255)	Nombre de la ciudad destino
Administrador	RD-04	Datos del administrador delegado
Imagen	File	Imagen de perfil asociada al destino
Gif	File	Imagen GIF asociada al destino

RD-06: Datos de una Universidad Erasmus

Universidad Erasmus		
RD-06	*Datos necesarios para el alta de una universidad erasmus*	
Atributo	**Tipo**	**Descripción**
Id	Cadena de caracteres (12)	Código ERASMUS único de la universidad de acogida del estudiante
Nombre	Cadena de caracteres (255)	Nombre de la universidad destino

RD-07: Datos para Inscripción en asociación Erasmus

Datos para la solicitud de usuario para la inscripción en asociación Erasmus		
RD-07	*Datos de una solicitud*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres (15)	Nick único del estudiante
Código asociación	Cadena de caracteres (21)	Identificación de la asociación

RD-08: Datos para Notificación a asociación Erasmus

Notificación asociación Erasmus		
RD-08	*Datos para el registro de una notificación por un administrador*	
Atributo	**Tipo**	**Descripción**
Código asociación	Cadena de caracteres (21)	Identificación de la asociación
Descripción	Cadena de caracteres (500)	Comentario de los cambios o sugerencias
Desactivación	Bit	Desactivación de asociación necesaria

RD-09: Datos Partipación a evento Erasmus

Partipación a evento Erasmus		
RD-09	*Datos para el registro de una participación de un usuario en un evento*	
Atributo	**Tipo**	**Descripción**
Usuario	Cadena de caracteres (15)	Nick único del estudiante
Código evento	Unsigned int	Identificación numérica del evento

RD-10: Datos de modificación de Estudiante Erasmus

Estudiante Erasmus		
RD-10	*Datos necesarios para la modificación de un usuario estudiante*	
Atributo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres(40)	Nombre de pila del estudiante
Apellidos	Cadena de caracteres(80)	Apellidos del estudiante
Imagen de perfil	File	Imagen para el perfil de usuario

RD-11: Datos de modificación de una Asociación Erasmus

Asociación Erasmus		
RD-11	*Datos necesarios para la modificación de una asociación Erasmus*	
Atributo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres (40)	Nombre completo de la asociación
Descripción	Cadena de caracteres (5000)	Motivación y detalles presentados por la asociación
Imagen de perfil	File	Imagen para el perfil de asociación
Imagen de portada	File	Imagen para el banner de la asociación
Abierta	Bit	Por defecto 1. Requerida solicitud si valor 0.
Pública	Bit	Por defecto 1. Privada con valor 0.
Estado	Bit	Por defecto 1 (ACTIVA). Inactiva si valor 0.

RD-12: Datos de modificación de un destino Erasmus

Destino Erasmus		
RD-12	*Datos necesarios para la modificación de un destino erasmus*	
Atributo	**Tipo**	**Descripción**
Ciudad	Cadena de caracteres (15)	Nombre de la ciudad destino
País	Cadena de caracteres (255)	Nombre de la ciudad destino
Imagen	File	Imagen de perfil asociada al destino
Gif	File	Imagen GIF asociada al destino

3.5. Requisitos no funcionales

RNF1: Requisito no funcional de Integridad

RNF1	Integridad
Descripción	La información del sistema será completa, sólida y exacta evitando datos duplicados, datos faltantes, alterados o incorrectos.

RNF2: Requisito no funcional de Usabilidad

RNF2	Usabilidad
Descripción	El sistema será fácil de usar, para que los usuarios puedan adaptarse lo más rápido posible. La información se encuentra fácilmente.

RNF3: Requisito no funcional de Eficacia

RNF3	Eficacia
Descripción	Las funcionalidades del sistema llevarán a cabo su realización de forma adecuada.

RNF4: Requisito no funcional de Accesibilidad

RNF4	Accessibilidad
Descripción	Ofrecerá una interfaz clara, sencilla y llamativa.

RNF5: Requisito no funcional de Seguridad

RNF5	Seguro
Descripción	Los datos personales de los administradores y usuarios se asegurarán mediante métodos de encriptado.

3.6. Casos de uso

CU-01: Caso de uso para Alta de usuarios Erasmus (Diagrama 1)

Caso de uso	*Alta de usuario Erasmus*	CU-01
Actores	Usuario Erasmus	
Tipo	Primario y esencial	
Referencias	RF-01	
Precondición		
Postcondición	El usuario es dado de alta en el sistema	

Propósito
Conseguir que el usuario se dé de alta en el sistema.

Resumen
El usuario suministra su información personal de acuerdo a lo requerido por el sistema (Nombre, Apellidos, Nick, Email, Contraseña, Destino Erasmus, Universidad de destino e Imagen de Perfil)

Curso normal (básico)			
1.	Usuario se registra en la web.	2.	Recepción de la solicitud y salvado.
		3.	Inicio de sesión y redirección a página principal.

Cursos Alternos	
2.a	El sistema detecta error en los datos e informa al usuario para que lo solucione.

CU-02: Caso de uso para Identificación de usuarios (Diagrama 2)

Caso de uso	*Identificación de usuario*	CU-02
Actores	Usuario	
Tipo	Primario y esencial	
Referencias	RF-02	
Precondición	El usuario debe estar dado de alta en el sistema	
Postcondición	Acceso a la aplicación	

Propósito
Identificarse como un usuario para acceder al sistema

Resumen
El usuario introduce su información de inicio de sesión (email/usuario y contraseña). Si estos datos son correctos la sesión será iniciada.

Curso normal (básico)			
1.	Usuario introduce sus datos en la web	2.	Verificación
		3.	Inicio de sesión y redirección a página principal.

Cursos Alternos	
3.a	El sistema no encuentra un usuario con estos credenciales e informa del error.

CU-03: Caso de uso para Cierre de sesión de usuarios (Diagrama 3)

Caso de uso	*Cierre de sesión de usuario*	CU-03
Actores	Usuario	
Tipo	Primario y esencial	
Referencias	RF-03	CU-02
Precondición	El usuario debe estar dado de alta en el sistema y con la sesión iniciada.	
Postcondición	La sesión queda terminada	

Propósito
Terminar la sesión actual y redirigir a pantalla de inicio

Resumen
El usuario que se encuentra identificado en la aplicación, tiene la opción de cerrar su sesión pulsando un botón destinado para ello.

Curso normal (básico)			
1.	Usuario pulsa en Cerrar Sesión	2.	Termino de sesión y redirección a inicio.

CU-04: Caso de uso para Consulta de usuarios (Diagrama 4)

Caso de uso	*Consulta de usuario*	CU-04
Actores	Usuario Erasmus	
Tipo	Primario y esencial	
Referencias	RF-04	CU-02
Precondición	El usuario debe estar dado de alta en el sistema y con la sesión iniciada.	
Postcondición	Conocer la información de sí mismo u otro usuario.	

Propósito
Consultar los datos de un usuario Erasmus.

Resumen
El usuario tiene la opción de acceder a su información en su propio perfil, donde aparecerán sus datos, sus eventos y asociaciones. Además puede consultar otros perfiles de usuarios (de la misma asociación o evento), los cuales no mostrarán toda la información sino una vista reducida

Curso normal (básico)			
1.	Usuario pulsa en Mi Perfil	2.	Redirección y recolección de los datos de usuario.

Cursos Alternos	
1.a	Usuario: El usuario selecciona el perfil de otro usuario. Sistema: Redirección a la página del usuario.

CU-05: Caso de uso para Modificación de usuarios (Diagrama 5)

Caso de uso	*Modificar usuario*	CU-05
Actores	Usuario Erasmus	
Tipo	Primario y esencial	
Referencias	RF-05	CU-04
Precondición	El usuario debe estar dado de alta en el sistema y con la sesión iniciada.	
Postcondición	Modificación de los datos de usuario en el sistema.	

Propósito
Posibilitar la modificación de los datos de los distintos usuarios del sistema.

Resumen
El usuario tiene la opción de modificar su información desde su perfil.

Curso normal (básico)			
1.	Usuario pulsa en Editar Perfil	2.	Redirección y recolección de los datos de usuario.
3.	Modificación de los datos	4.	Verificación, salvado y redirección al perfil.

Cursos Alternos	
4.a	Sistema: Detecta error en el formato de los datos. Informa del error y permanece en la página.

CU-06: Caso de uso para Dar de baja usuarios (Diagrama 6)

Caso de uso	*Dar de baja usuario Erasmus*	CU-06
Actores	Usuario Erasmus	
Tipo	Primario y esencial	
Referencias	RF-06	CU-04
Precondición	El usuario debe estar dado de alta en el sistema y con la sesión iniciada.	
Postcondición	El usuario es eliminado del sistema.	

Propósito
Posibilitar la baja voluntaria de los datos de los distintos usuarios del sistema.

Resumen
El usuario tiene la opción de darse de baja del sistema desde su perfil.

Curso normal (básico)			
1.	Usuario pulsa en Eliminar Perfil	2.	Doble verificación.
3.	Aceptación	4.	Eliminado de los datos de usuario de la BD.

Cursos Alternos	
3.a	Usuario: No acepta una de las verificaciones. Sistema: Termina los pop-ups y muestra de nuevo el perfil.

CU-07: Caso de uso para Inscripción de usuarios en asociaciones Erasmus (Diagrama 7)

Caso de uso	*Inscripción de usuario en asociación Erasmus*	CU-07
Actores	Usuario Erasmus, Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-07, RF-08	CU-02, CU-12
Precondición	El usuario debe estar dado de alta en el sistema y con la sesión iniciada. La asociación debe estar registrada en el sistema.	
Postcondición	El usuario es inscrito en una asociación.	

Propósito
Posibilitar la inscripción de usuarios en asociaciones Erasmus.

Resumen
Desde el perfil de una asociación, un estudiante podrá solicitar su inscripción. Si la asociación requiere de verificación, se creará y deberá ser resuelta. En caso contrario la inscripción será automática.

Curso normal (básico)			
1.	Usuario pulsa en Inscribirse	2.	Una solicitud es creada y enviada al administrador de la asociación.
3.	Administrador recibe notificación y acepta la solicitud.	4.	Modificación del estado de la solicitud y usuario queda registrado.

Cursos Alternos	
2.a	Sistema: La asociación no requiere de solicitud. Automáticamente pasa a ser aceptada.
3.a	Admin Asociación: Rechaza la solicitud. Sistema: La solicitud es eliminada de la BD.

CU-08: Caso de uso para Eliminar la inscripción de usuarios en asociaciones Erasmus (Diagrama 8)

Caso de uso	*Eliminar inscripción de usuario a asociación Erasmus*	CU-08
Actores	Usuario Erasmus, Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-09	CU-04, CU-12
Precondición	El usuario debe estar dado de alta en el sistema y pertenecer a la asociación. La asociación debe estar registrada en el sistema y el administrador debe tener la sesión iniciada.	
Postcondición	La inscripción del usuario es eliminada.	

Propósito
Posibilitar la eliminación de usuarios de asociaciones Erasmus por parte del administrador

Resumen
Desde el perfil de una asociación, un administrador podrá dar de baja la inscripción de uno de sus usuarios.

Curso normal (básico)			
1.	Administrador consulta el perfil de un usuario de la asociación.		
2.	Pulsa sobre Dar de baja inscripción	3.	Eliminación de la BD de la solicitud de inscripción del usuario con la asociación.

CU-09: Caso de uso para Inscripción de usuarios en eventos Erasmus (Diagrama 9)

Caso de uso	*Inscripción en evento Erasmus*	CU-09
Actores	Usuario Erasmus	
Tipo	Primario y esencial	
Referencias	RF-10	CU-02, CU-15
Precondición	El usuario debe estar dado de alta en el sistema y pertenecer a la asociación. El evento debe estar almacenado en la base de datos.	
Postcondición	El usuario queda relacionado en la BD con el evento	

Propósito
Posibilitar la inscripción de usuarios a eventos.

Resumen
Desde la consulta de un evento, un usuario perteneciente a la asociación podrá apuntarse al evento.

Curso normal (básico)			
1.	Usuario consulta el evento de una asociación.		
2.	Pulsa sobre Inscribirse	3.	El sistema genera la asociación en la base de datos.

CU-10: Caso de uso para Eliminar participación de usuarios en eventos Erasmus (Diagrama 10)

Caso de uso	*Eliminar participación en evento Erasmus*	CU-10
Actores	Usuario Erasmus	
Tipo	Primario y esencial	
Referencias	RF-11	CU-02, CU-15
Precondición	El usuario debe estar dado de alta en el sistema y pertenecer a la asociación. El evento debe estar almacenado en la base de datos.	
Postcondición	El usuario deja de estar relacionado en la BD con el evento	

Propósito
Posibilitar la cancelación de usuarios en su participación en eventos.

Resumen
Desde la consulta de un evento, un usuario perteneciente a la asociación podrá cancelar su participación al evento.

Curso normal (básico)			
1.	Usuario consulta el evento de una asociación.		
2.	Pulsa sobre Cancelar participación	3.	El sistema elimina la asociación en la BD.

CU-11: Caso de uso para Alta de asociaciones Erasmus (Diagrama 11)

Caso de uso	*Alta de asociación Erasmus*	CU-11
Actores	Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-12	
Precondición		
Postcondición	La asociación es dada de alta en el sistema	

Propósito
Conseguir publicar una nueva asociación alta en el sistema

Resumen
El administrador suministra toda la información de una asociación de acuerdo a lo requerido por el sistema (Nombre, Acrónimo, Email, Contraseña, Destino Erasmus, Descripción, Imagen de Perfil, Imagen de portada, Solicitud de inscripción)

Curso normal (básico)			
1.	Administrador registra los datos en la web.	2.	Recepción de la solicitud y salvado.
		3.	Inicio de sesión y redirección a página principal de la asociación.

Cursos Alternos	
2.a	El sistema detecta error en los datos e informa al usuario para que lo solucione.

CU-12: Caso de uso para Consulta de asociaciones Erasmus (Diagrama 12)

Caso de uso	*Consulta de asociación*	CU-12
Actores	Usuario	
Tipo	Primario y esencial	
Referencias	RF-13	CU-02, CU-11
Precondición	El usuario debe estar dado de alta en el sistema y con la sesión iniciada. La asociación debe estar registrada en el sistema.	
Postcondición	Conocer la información de una asociación	

Propósito
Consultar los datos de una asociación Erasmus.

Resumen
El usuario tiene la opción de acceder a la información de una asociación, donde aparecerá sus información, sus eventos y usuarios. En caso de ser privada sólo podrá acceder a esta información si está inscrito en ella, en caso contrario la vista será reducida. Además, el usuario administrador de esta asociación tendrá el acceso completo a toda la información.

Curso normal (básico)			
1.	Usuario pulsa en perfil de asociación	2.	Redirección y recolección de los datos de la asociación.

Cursos Alternos	
1.a	Administrador asociación: Pulsa en su asociación. Sistema: Redirección a la página de la asociación con la información completa y las opciones de modificación oportunas.
1.b	Usuario: Usuario pulsa en perfil de asociación pero esta es privada y no pertenece a ella. Sistema: Redirecciona y recoge los datos mínimos de la asociación.

CU-13: Caso de uso para Modificación de asociaciones Erasmus (Diagrama 13)

Caso de uso	*Modificar asociación Erasmus*	CU-13
Actores	Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-14	CU-04
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada.	
Postcondición	Modificación de los datos de usuario en el sistema.	

Propósito
Posibilitar la modificación de los datos de las distintas asociaciones del sistema.

Resumen
El administrador de una asociación tiene la opción de modificar la información de esta desde su perfil.

Curso normal (básico)			
1.	Administrador pulsa en Editar asociación	2.	Redirección y recolección de los datos de la asociación.
3.	Modificación de los datos	4.	Verificación, salvado y redirección al perfil

Cursos Alternos	
4.a	Sistema: Detecta error en el formato de los datos. Informa del error y permanece en la página.

CU-14: Caso de uso para Creación de eventos Erasmus (Diagrama 14)

Caso de uso	*Crear evento Erasmus*	CU-14
Actores	Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-15	CU-02, CU-12, CU-17
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada.	
Postcondición	Dada de alta de evento el sistema.	

Propósito
Posibilitar la agregación de eventos a las distintas asociaciones del sistema.

Resumen
El administrador de una asociación crea un evento Erasmus desde el perfil de la asociación.

Curso normal (básico)			
1.	Administrador pulsa en Crear evento	2.	Redirección a página creación.
3.	Introducción de los datos	4.	Verificación, salvado y redirección al evento creado

Cursos Alternos	
4.a	Sistema: Detecta error en el formato de los datos. Informa del error y permanece en la página.

CU-15: Caso de uso para Consulta de eventos Erasmus (Diagrama 15)

Caso de uso	*Consulta de evento Erasmus*	CU-15
Actores	Usuario	
Tipo	Primario y esencial	
Referencias	RF-16	CU-02, CU-14
Precondición	El usuario debe estar dado de alta en el sistema y con la sesión iniciada. La asociación debe estar registrada en el sistema. El evento debe estar dado de alta en el sistema.	
Postcondición	Conocer la información de un evento.	

Propósito
Consultar los datos de un evento Erasmus.

Resumen
El usuario puede acceder a la información de los eventos de los que es participante o aún no, donde aparecerá su información. Además, el usuario administrador de esta asociación tendrá el acceso completo a toda la información

Curso normal (básico)			
1.	Usuario pulsa en el evento	2.	Redirección y recolección de los datos del evento.

Cursos Alternos	
1.a	Administrador asociación: Pulsa en el evento.. Sistema: Redirección a la página del evento con la información completa y las opciones de modificación oportunas.

CU-16: Caso de uso para Consulta listado de eventos Erasmus (Diagrama 16)

Caso de uso	*Consulta listado de evento Erasmus*	CU-16
Actores	Usuario	
Tipo	Primario y esencial	
Referencias	RF-17	CU-02, CU-12
Precondición	El usuario debe estar dado de alta en el sistema y con la sesión iniciada. La asociación debe estar registrada en el sistema.	
Postcondición	Conocer los eventos relacionados con una asociación.	

Propósito
Consultar los eventos de una asociación Erasmus.

Resumen
Un usuario puede consultar el listado completo de eventos de una asociación, desde una página dedicada para ello.

Curso normal (básico)			
1.	Usuario pulsa en Eventos de asociación	2.	Redirección y recolección de todos los eventos de la asociación.

Cursos Alternos	
2.a	Sistema: Si no existen eventos, la página se encontrará vacía y lo comunicará al usuario.

CU-17: Caso de uso para Gestión de dirección de evento con Google Maps (Diagrama 17)

Caso de uso	*Gestión de dirección de evento con Google Maps*	CU-17
Actores	Administrador asociación	
Tipo	Secundario y deseable	
Referencias	RF-18	CU-02
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada.	
Postcondición	Asignar una dirección a un evento	

Propósito
Asignar una dirección a un evento de manera sencilla utilizando el motor de Google Maps

Resumen
Desde la página de creación de eventos de la asociación, se podrá seleccionar una dirección mediante la herramienta de Google Maps, guardando en la base de datos la dirección seleccionada.

Curso normal (básico)			
1.	Usuario selecciona dirección en Google Maps	2.	Recolección de la dirección completa y almacenamiento

CU-18: Caso de uso para Modificación de eventos Erasmus (Diagrama 18)

Caso de uso	*Modificación de evento*	CU-18
Actores	Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-19	CU-02, CU-15, CU-17
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada.	
Postcondición	Modificar un evento	

Propósito
Modificar los datos de un evento Erasmus

Resumen
Desde la página de consulta de evento existirá la opción de edición, con la que se abrirá una página con la posibilidad de modificar todos los datos.

Curso normal (básico)			
1.	Usuario selecciona editar evento	2.	Recolecciońde los datos y redirección
3.	Modificación de los datos deseados	4.	Salvado y almacenamiento de la modificación.

Cursos Alternos	
3.a	Los datos introducidos no son correctos. No se produce la modificación
3.b	Cancelación de modificación. No se produce la modificación

CU-19: Caso de uso para Eliminación de eventos Erasmus (Diagrama 19)

Caso de uso	*Eliminar evento Erasmus*	CU-19
Actores	Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-20	CU-02, CU-15
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada.	
Postcondición	Eliminar un evento del sistema	

Propósito
Eliminar los datos de un evento Erasmus

Resumen
Desde la página de consulta de evento existirá la opción de eliminar, con la que se borrarán los datos almacenados sobre el evento en cuestión.

Curso normal (básico)			
1.	Usuario selecciona eliminar evento	2.	Apertura de ventana de confirmación
3.	Acepta la acción	4.	Eliminación de la información sobre el evento

Cursos Alternos	
3.a	Cancela acción. No se produce el borrado

CU-20: Caso de uso para Dar de baja a usuarios de asociaciones (Diagrama 20)

Caso de uso	*Dar de baja a usuario de asociación*	CU-20
Actores	Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-21	CU-02, CU-04
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada. El usuario debe de pertenecer a la asociación.	
Postcondición	Usuario deja de estar inscrito en la asociación	

Propósito
Eliminar la inscripción de un usuario a una asociación

Resumen
Desde la página de consulta de usuario existirá la opción de darlo de baja de la asociación, con la que se borrarán los datos almacenados sobre su inscripción.

Curso normal (básico)			
1.	Administrador selecciona usuario a borrar	2.	Redirección a consulta de usuario
3.	Selecciona dar de baja de asociación	4.	Apertura de ventana de confirmación
5.	Acepta acción	6.	Eliminación de la información sobre la inscripción

Cursos Alternos	
5.a	Cancela acción. No se produce el borrado

CU-21: Caso de uso para Consultar listado de usuarios pertenecientes a una asociación (Diagrama 21)

Caso de uso	*Consulta listado de usuarios pertenecientes a una asociación*	CU-21
Actores	Administrador asociación	
Tipo	Primario y esencial	
Referencias	RF-22	CU-02
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada.	
Postcondición	Conocer los usuarios relacionados con una asociación.	

Propósito
Un administrador podrá consultar la lista de usuarios pertenecientes a la asociación, y tendrá la opción de consultar su información.

Resumen
Un administrador puede consultar el listado completo de usuarios de una asociación, desde una página dedicada para ello.

Curso normal (básico)			
1.	Usuario pulsa en Usuarios de asociación	2.	Redirección y recolección de todos los usuarios de la asociación.

Cursos Alternos	
2.a	Sistema: Si no existen usuarios, la página se encontrará vacía y lo comunicará al administrador.

CU-22: Caso de uso para Responder notificaciones sobre cambios a realizar (Diagrama 22)

Caso de uso	*Responder notificación sobre cambios a realizar*	CU-22
Actores	Administrador asociación, administrador del sistema	
Tipo	Primario y esencial	
Referencias	RF-23	CU-27, CU-26
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada. El usuario debe de pertenecer a la asociación.	
Postcondición	Modificación del estado de la solicitud en la base de datos	

Propósito
Resolver la notificación recibida

Resumen
Desde el perfil de la asociación, un administrador podrá ver sus notificaciones, donde podrá finalizarlas tras haberlas consultado

Curso normal (básico)			
		1.	CU-27.
2.	Administrador de asociación pincha en notificaciones		
3.	Lee la propuesta y la acepta	4.	Eliminación de la información sobre la notifcación
		5.	CU-26

CU-23: Caso de uso para Búsqueda de eventos ó asociaciones (Diagrama 23)

Caso de uso	*Búsqueda de evento ó asociación*	CU-23
Actores	Usuario	
Tipo	Primario y esencial	
Referencias	RF-24	CU-02
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada.	
Postcondición	Listado de eventos y asociaciones que concuerden con la búsqueda	

Propósito
Búsqueda de eventos o asociaciones según diferentes filtros

Resumen
Desde la barra de navegación, siempre existirá una opción de búsqueda. Se introducirá un texto y se aplicarán diferentes filtros.

Curso normal (básico)			
1.	Usuario escribe en la barra y selecciona los filtros deseados	2.	Recolección de búsqueda y selección de los datos
		3.	Muestra eventos y asociaciones de manera listada

Cursos Alternos	
3.a	No hay coincidencias. La lista se muestra vacía y se informa al usuario.

CU-24: Caso de uso para Consulta de eventos en calendario (Diagrama 24)

Caso de uso	*Consulta de eventos en calendario*	CU-24
Actores	Usuario	
Tipo	Primario y esencial	
Referencias	RF-25	CU-02, CU-15
Precondición	La asociación debe estar dada de alta en el sistema y con la sesión iniciada.	
Postcondición	Redirección a evento desde calendario	

Propósito
Búsqueda de eventos por fecha en calendario interactivo. El calendario denotará los días que contengan eventos

Resumen
Desde la página de la asociación, aparecerá un calendario interactivo que permitirá listará los eventos por fecha y permitirá su consulta.

Curso normal (básico)			
1.	Usuario interactúa con el calendario y selecciona una fecha.	2.	Listado de eventos
3.	Selecciona un evento	4.	Redirección a consulta de evento

Cursos Alternos	
2.a	La fecha no tiene eventos. No ocurre nada.

CU-25: Caso de uso para Eliminar asociaciones Erasmus (Diagrama 25)

Caso de uso	*Eliminar asociación Erasmus*	CU-25
Actores	Administrador del sistema	
Tipo	Primario y esencial	
Referencias	RF-26	CU-02
Precondición	La asociación debe estar almacenada en la base de datos. El administrador debe tener la sesión iniciada	
Postcondición	La asociación queda eliminada del sistema	

Propósito
Dar de baja a una asociación del sistema

Resumen
Desde el perfil de una asociación, un administrador podrá eliminar esta.

Curso normal (básico)			
1.	Usuario selecciona eliminar asociación.	2.	Apertura de ventana de confirmación
3.	Acepta la acción	4.	Eliminación de la información sobre la asociación

Cursos Alternos	
3.a	Cancela acción. No se produce el borrado

CU-26: Caso de uso para Cambiar el estado de asociaciones Erasmus (Diagrama 26)

Caso de uso	*Cambiar estado asociación Erasmus*	CU-26
Actores	Administrador del sistema	
Tipo	Primario y esencial	
Referencias	RF-27	CU-02
Precondición	La asociación debe estar almacenada en la base de datos. El administrador debe tener la sesión iniciada	
Postcondición	El estado de la asociación cambia en el sistema	

Propósito
Activar ó suspender el estado de una asociación en la plataforma.

Resumen
Desde el perfil de una asociación, un administrador podrá cambiar el estado de esta.

Curso normal (básico)			
1.	Usuario selecciona cambiar estado asociación	2.	Apertura de ventana de confirmación
3.	Acepta la acción	4.	Activación o suspensión de la asociación en la base de datos.

Cursos Alternos	
3.a	Cancela acción. No se produce el cambio

CU-27: Caso de uso para Solicitar cambios a asociaciones (Diagrama 27)

Caso de uso	*Solicitar cambios a asociación*	CU-27
Actores	Administrador del sistema	
Tipo	Primario y esencial	
Referencias	RF-28	CU-02, CU-26
Precondición	La asociación debe estar almacenada en la base de datos. El administrador debe tener la sesión iniciada	
Postcondición	La asociación recibe notificación sobre cambio	

Propósito
Dar a conocer a la asociación de cambios que debe realizar, generando una notificación con las especificaciones.

Resumen
Desde el perfil de una asociación, un administrador podrá enviar un reporte a su administrador. Mientras se encuentra sin resolver, la asociación queda suspendida (CU-26)

Curso normal (básico)			
1.	Usuario selecciona Solicitar cambios	2.	Apertura de ventana de texto
3.	Rellena el reporte y acepta su envío	4.	CU-26
		5.	Creación de la notificación en el sistema y redirección a página principal.

Cursos Alternos	
3.a	Cancela acción. No se produce el envío.

3.7. Diagramas de Caso de Uso

Para la realización de los diagramas de Caso de Uso, estos se han dividido en susbsistemas que agrupen conceptos en común: Usuarios, Asociaciones, Inscripciones y Eventos.

3.7.1. Gestión de usuarios

Los usuarios se dividen según sus funciones y servicios disponibles: (I) estudiante Erasmus, (II) administrador de asociación, (III) administrador del sistema.

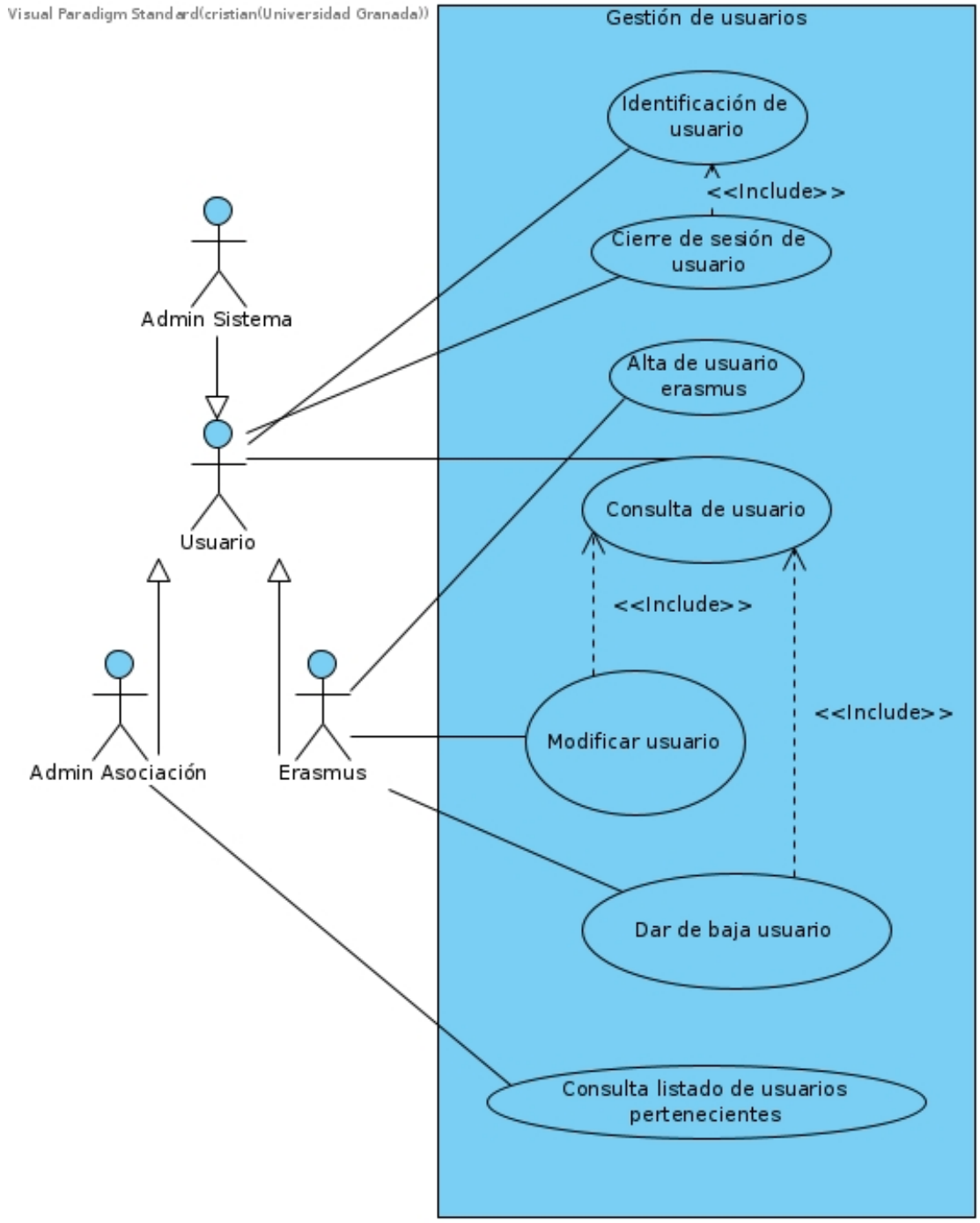


Figura 3.1: Diagrama de caso de uso de gestión de usuarios

3.7.2. Gestión de asociaciones

El administrador de asociación se encarga de todas las funcionalidades. La consulta de asociación es común para todos los usuarios. El administrador del sistema puede solicitar cambios o cambiar el estado de la asociación.

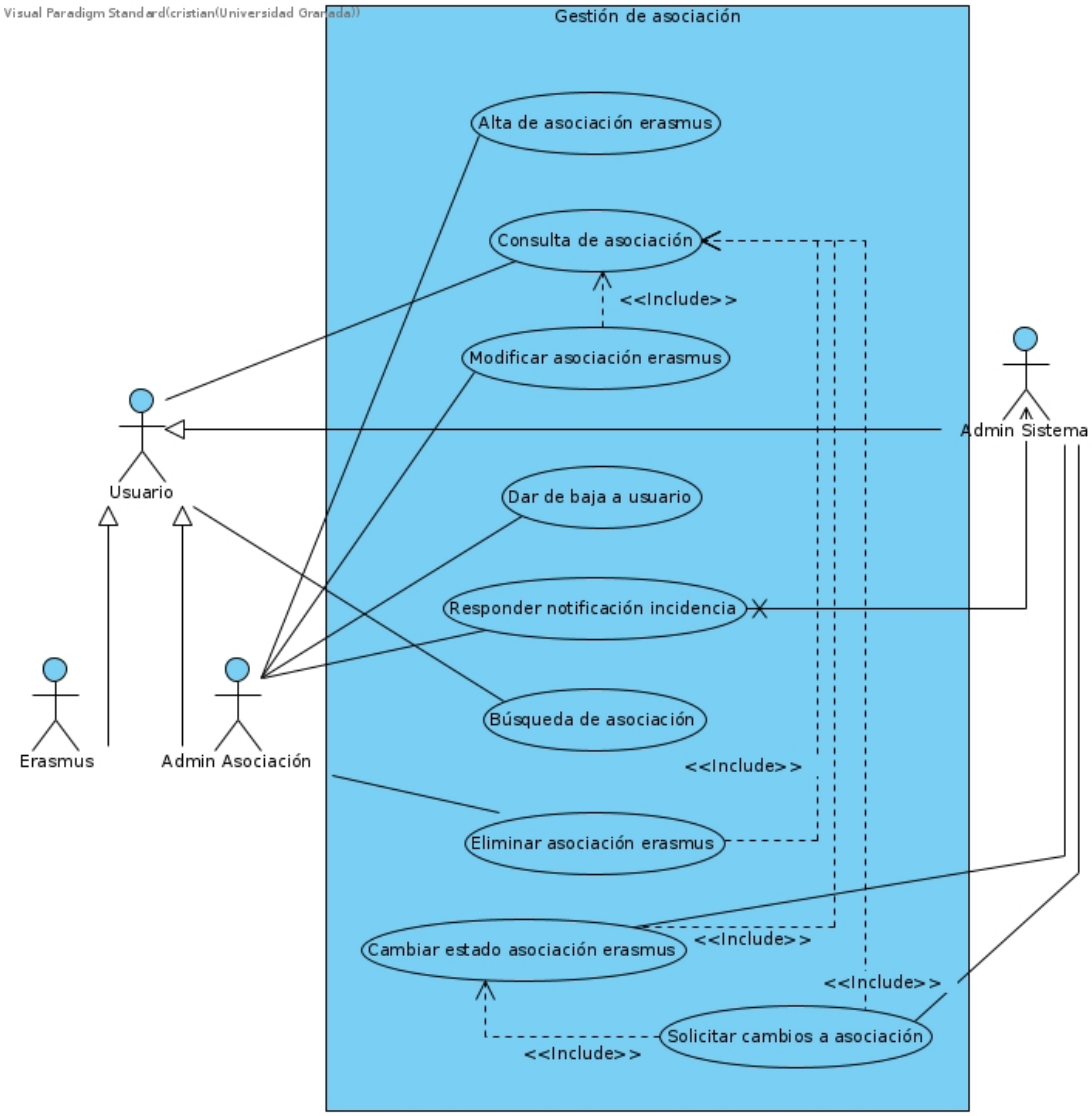


Figura 3.2: Diagrama de caso de uso de gestión de asociaciones

3.7.3. Gestión de inscripciones

Las inscripciones se realizan por parte del estudiante, salvo aquella dirigida a la asociación, que puede requerir de la confirmación por parte de esta.

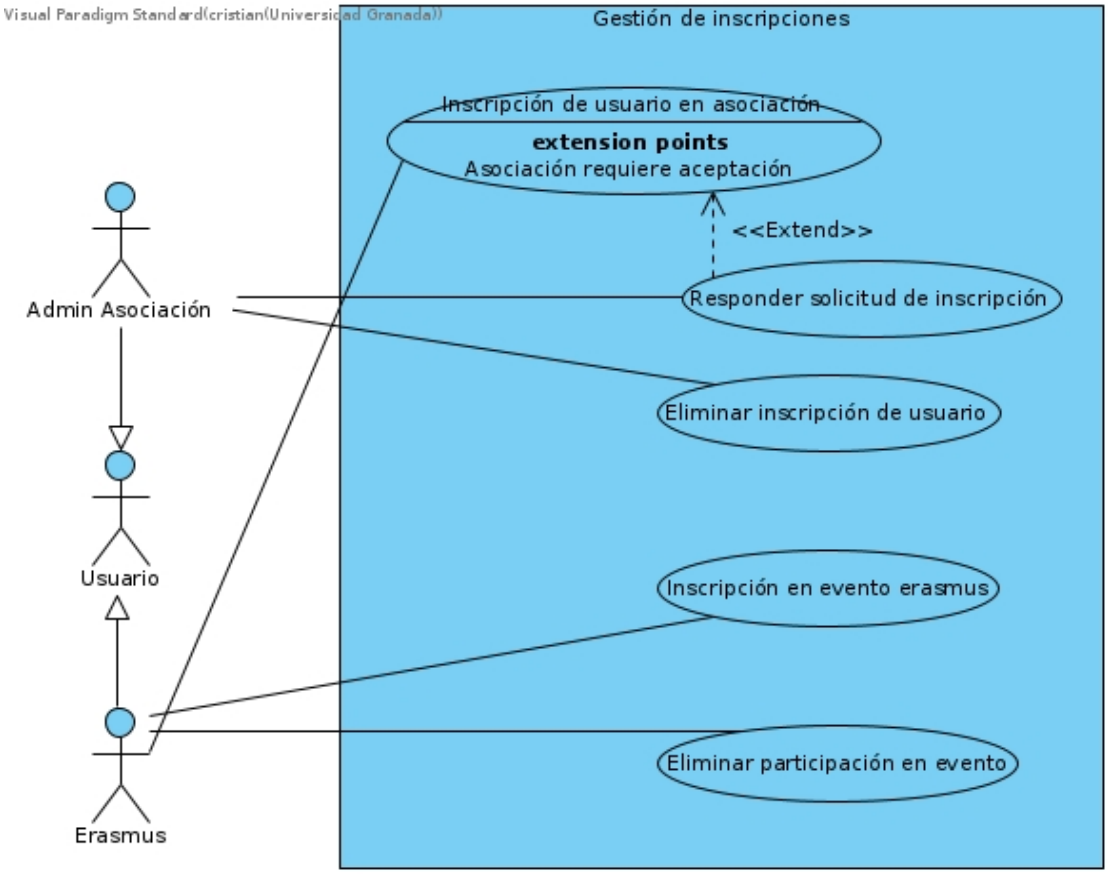


Figura 3.3: Diagrama de caso de uso de gestión de inscripciones

3.7.4. Gestión de eventos

La consulta y búsqueda de eventos será común a todos los usuarios. En cuanto a la gestión de estos la realizarán los administradores de asociación.

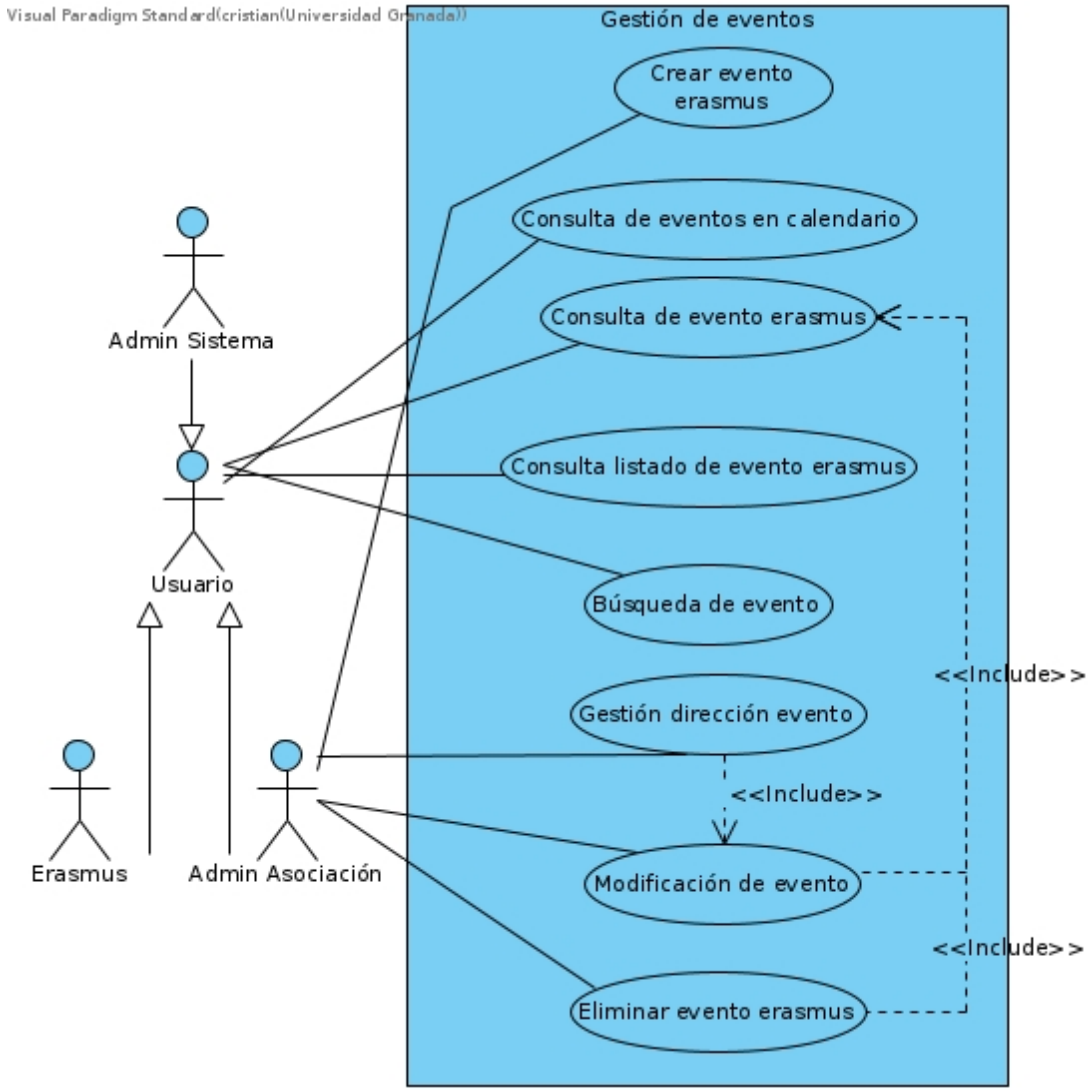


Figura 3.4: Diagrama de caso de uso de gestión de eventos

3.8. Diagramas de secuencia

Diagrama 1: Diagrama de secuencia de Alta de usuario Erasmus (CU-01)

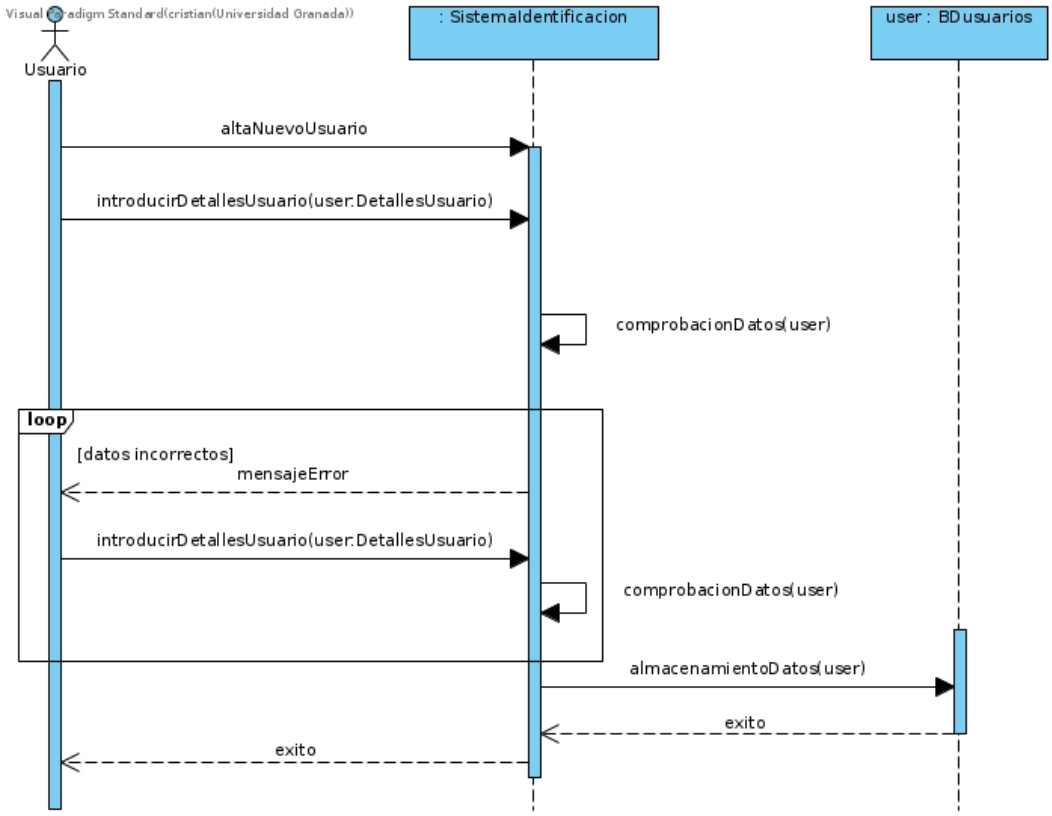


Diagrama 2: Diagrama de secuencia de Identificación de usuarios (CU-02)

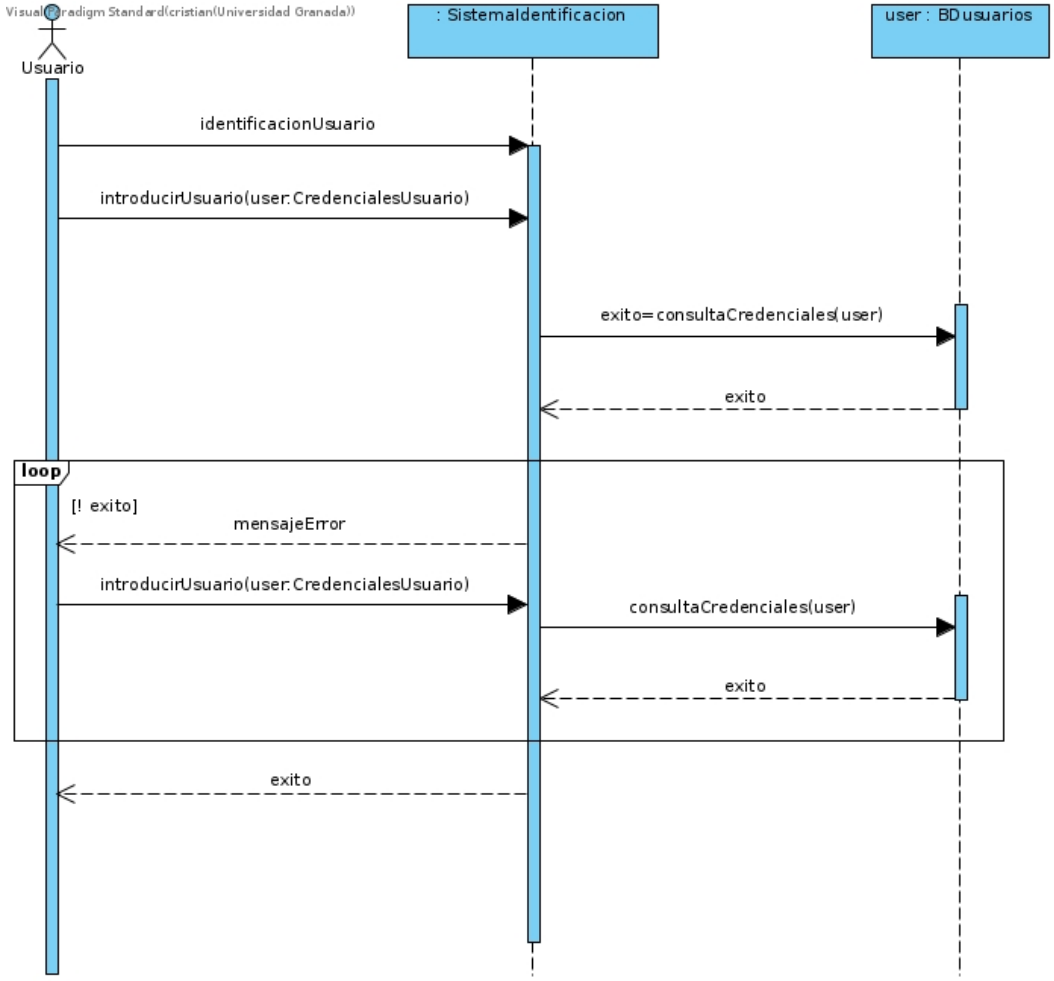


Diagrama 3: Diagrama de secuencia de Cierre de sesión de usuario (CU-03)

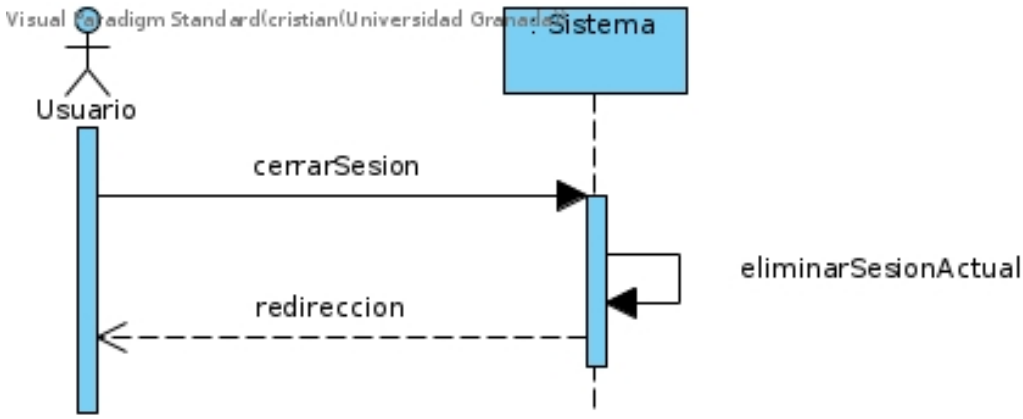


Diagrama 4: Diagrama de secuencia de Consulta de usuarios (CU-04)

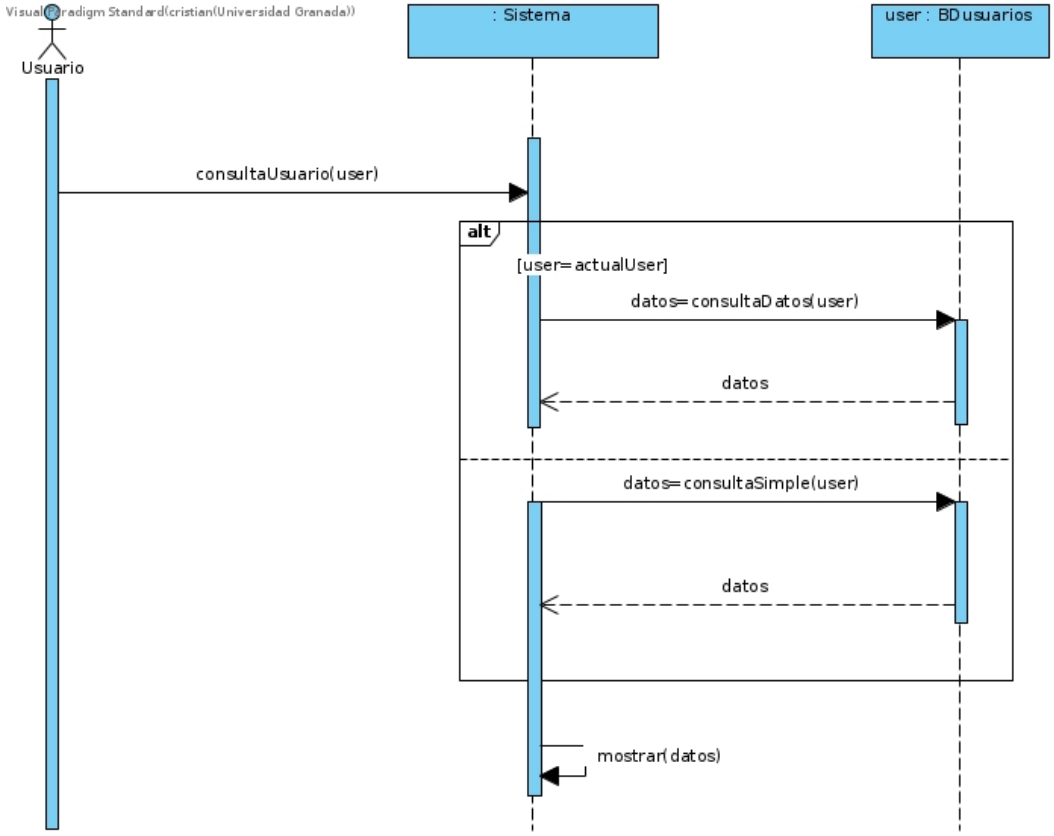


Diagrama 5: Diagrama de secuencia de Modificación de perfiles de usuario (CU-05)

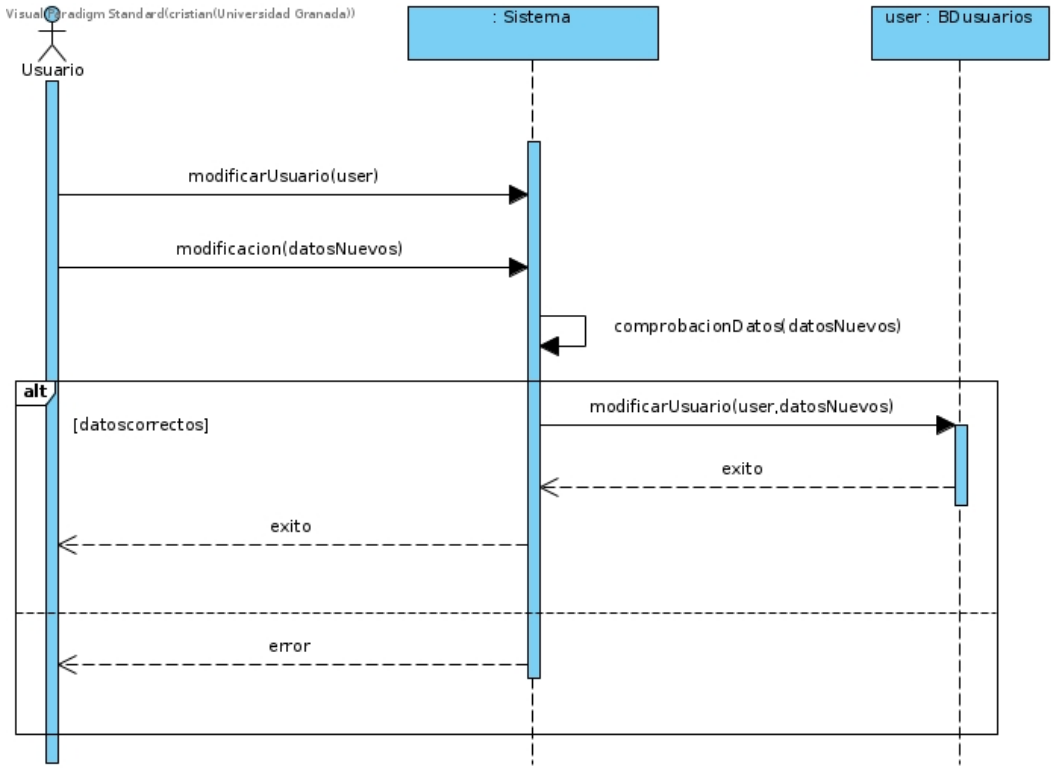


Diagrama 6: Diagrama de secuencia de Baja de usuarios (CU-06)

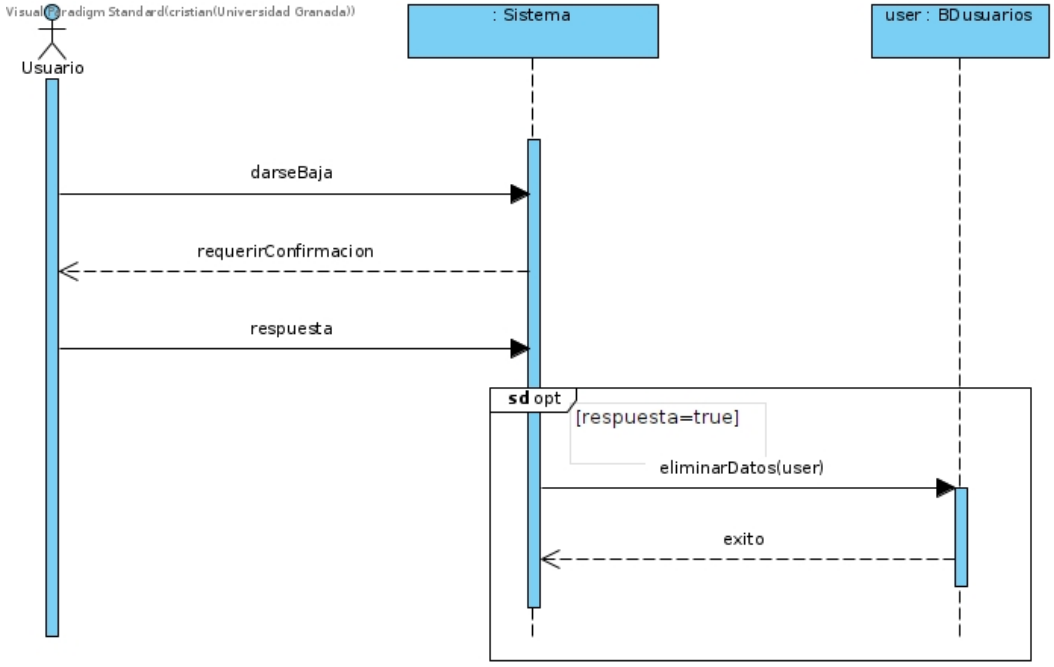


Diagrama 7: Diagrama de secuencia de Inscripción de usuarios en asociaciones Erasmus (CU-07)

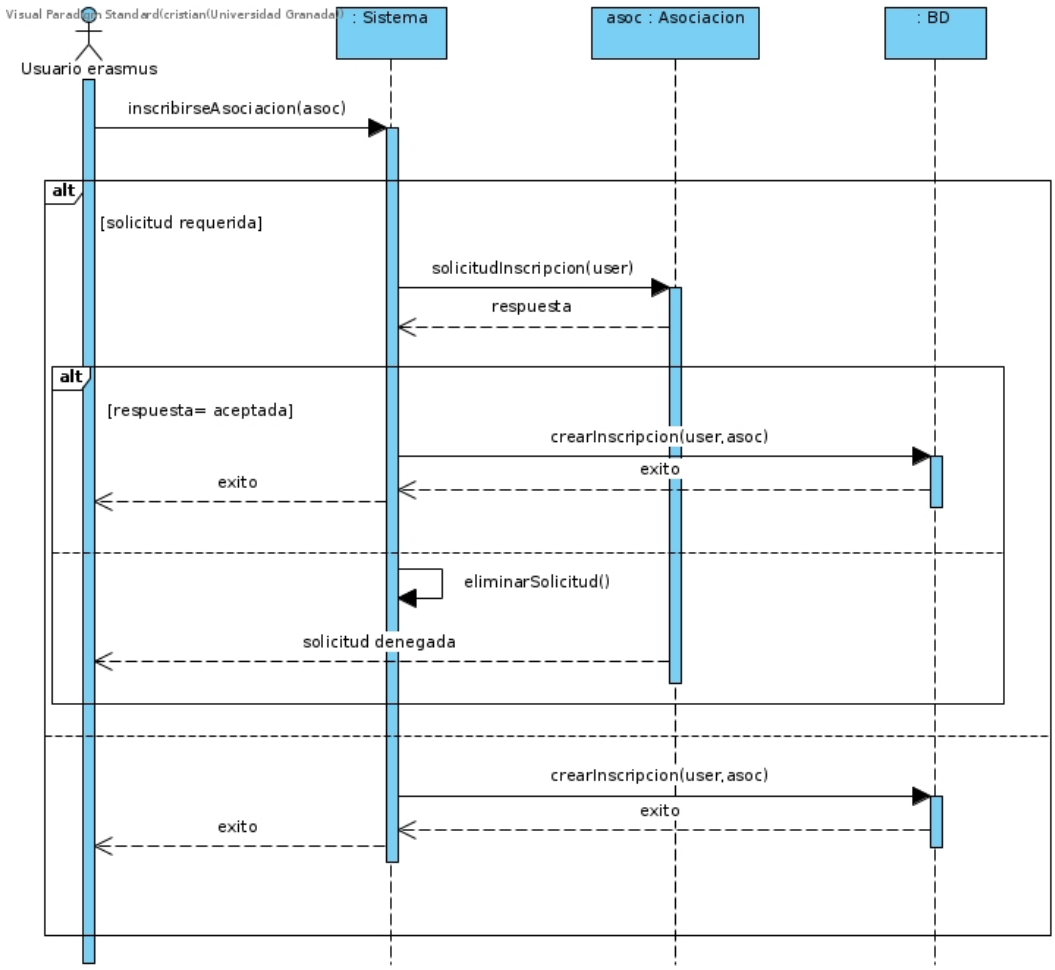


Diagrama 8: Diagrama de secuencia de Eliminar la inscripción de usuarios en asociaciones Erasmus (CU-08)

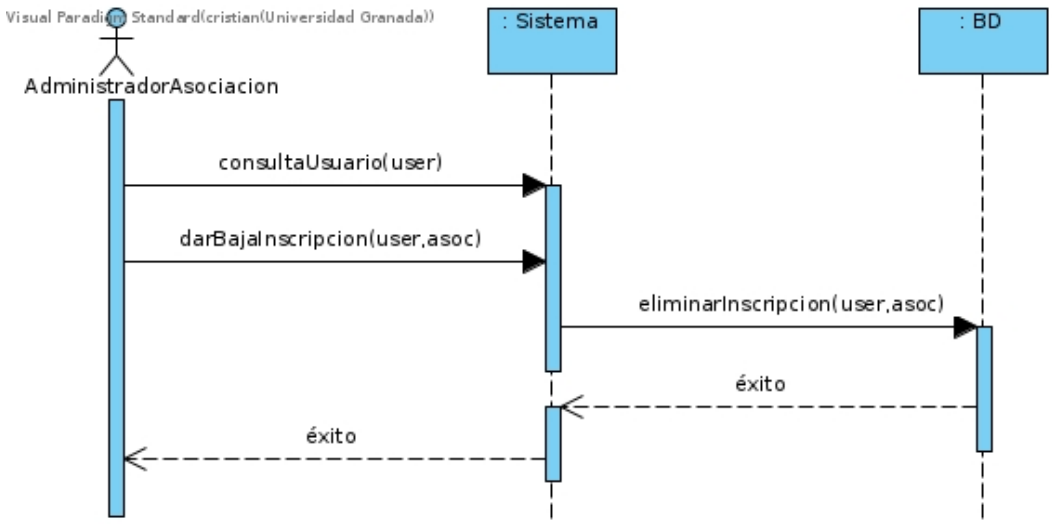


Diagrama 9: Diagrama de secuencia de Inscripción de usuarios en eventos Erasmus (CU-09)

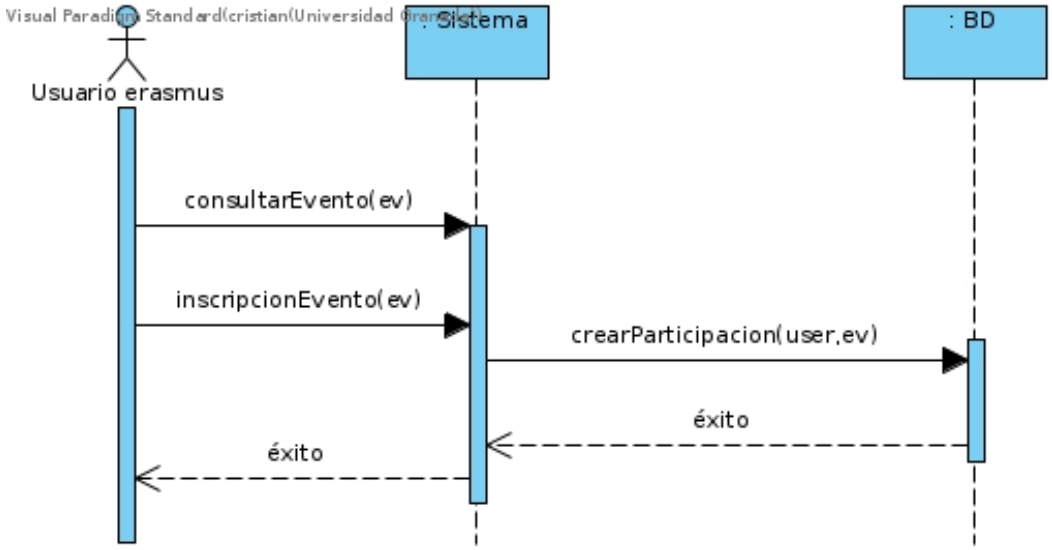


Diagrama 10: Diagrama de secuencia de Eliminar participación de usuarios en eventos Erasmus (CU-10)

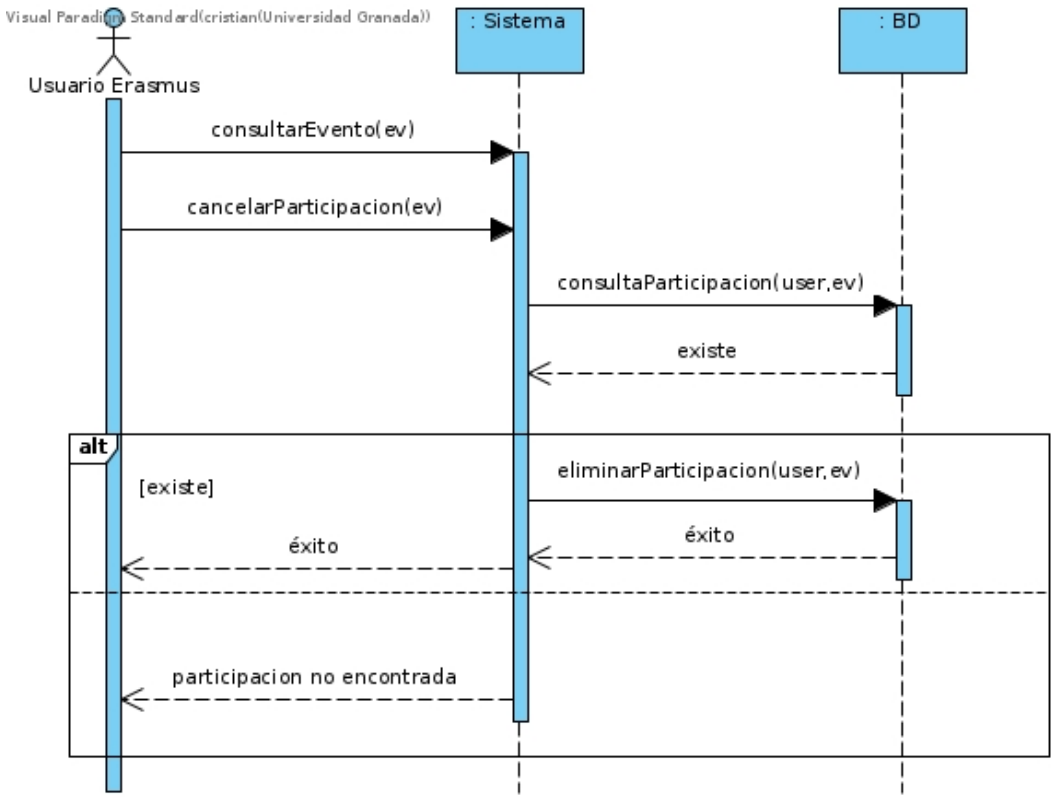


Diagrama 11: Diagrama de secuencia de Alta de asociaciones Erasmus (CU-11)

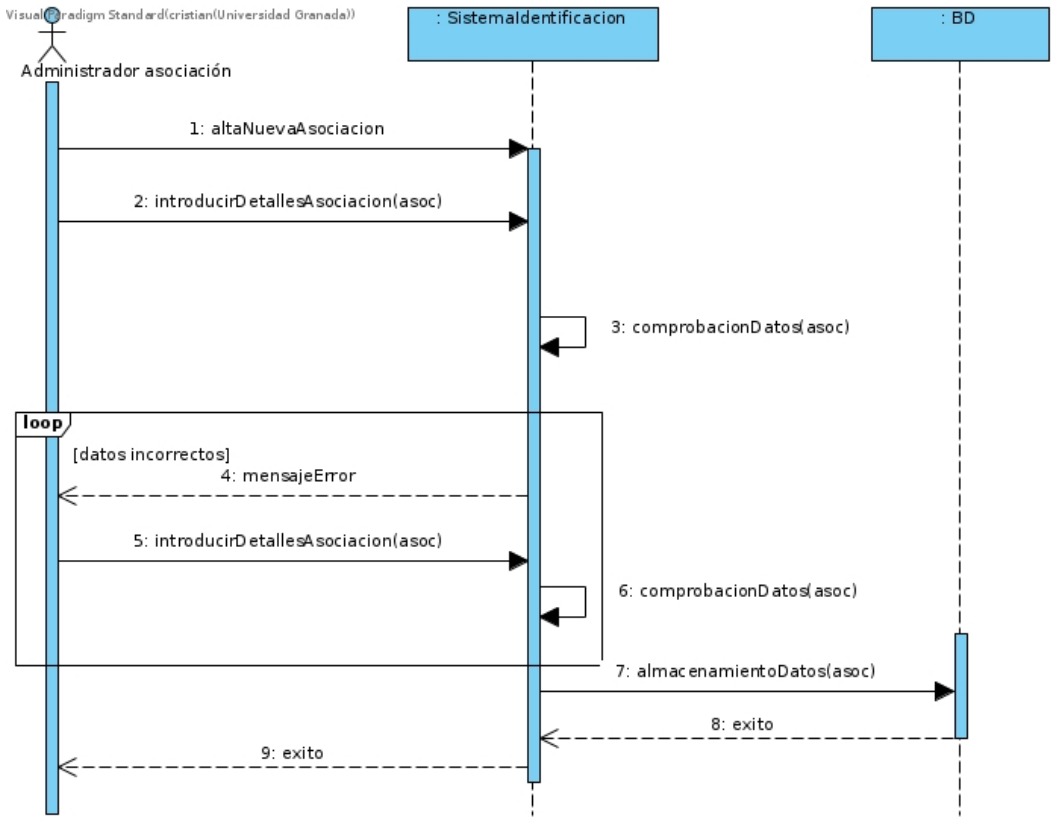


Diagrama 12: Diagrama de secuencia de Consulta de asociaciones Erasmus (CU-12)

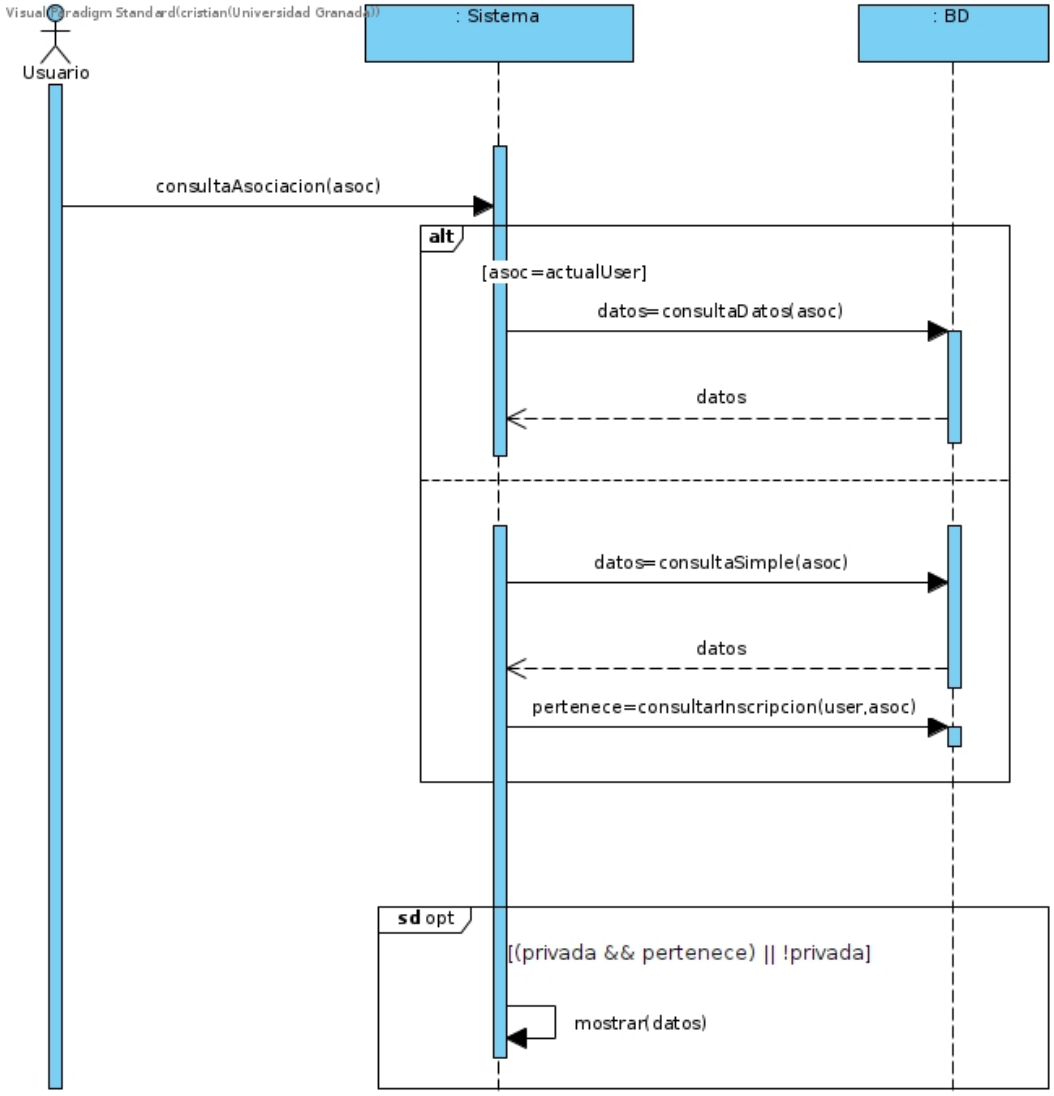


Diagrama 13: Diagrama de secuencia de Modificación de asociaciones Erasmus (CU-13)

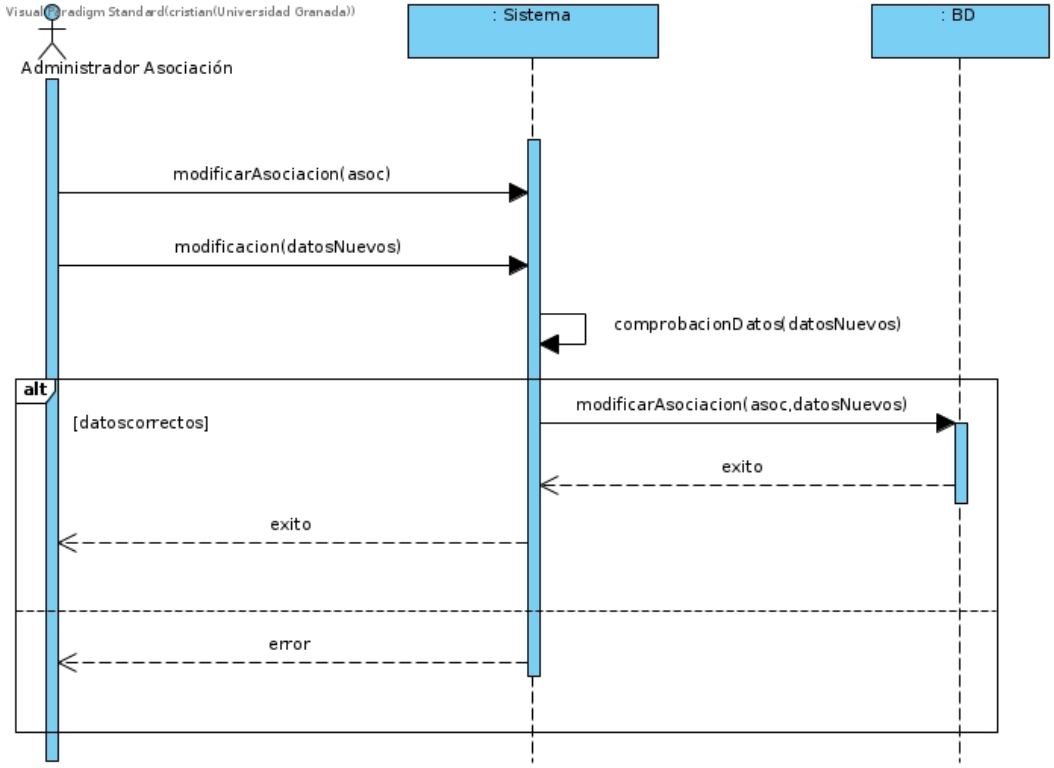


Diagrama 14: Diagrama de secuencia de Creación de eventos Erasmus (CU-14)

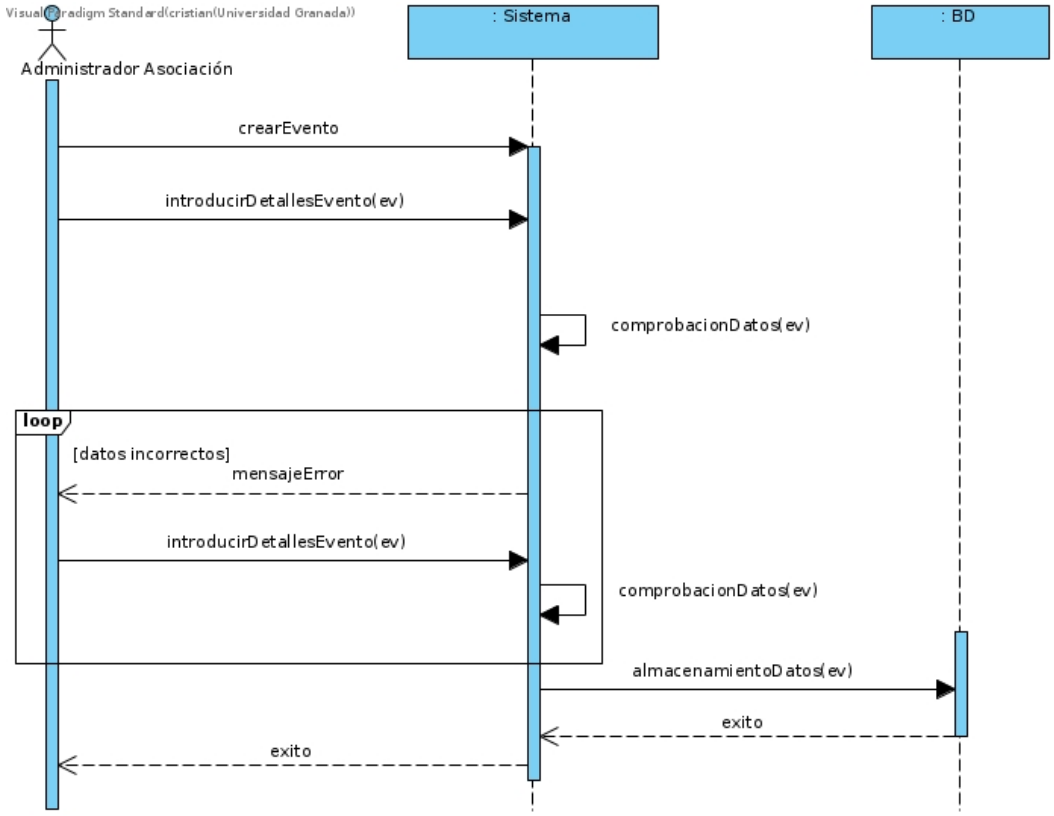


Diagrama 15: Diagrama de secuencia de Consulta de eventos Erasmus (CU-15)

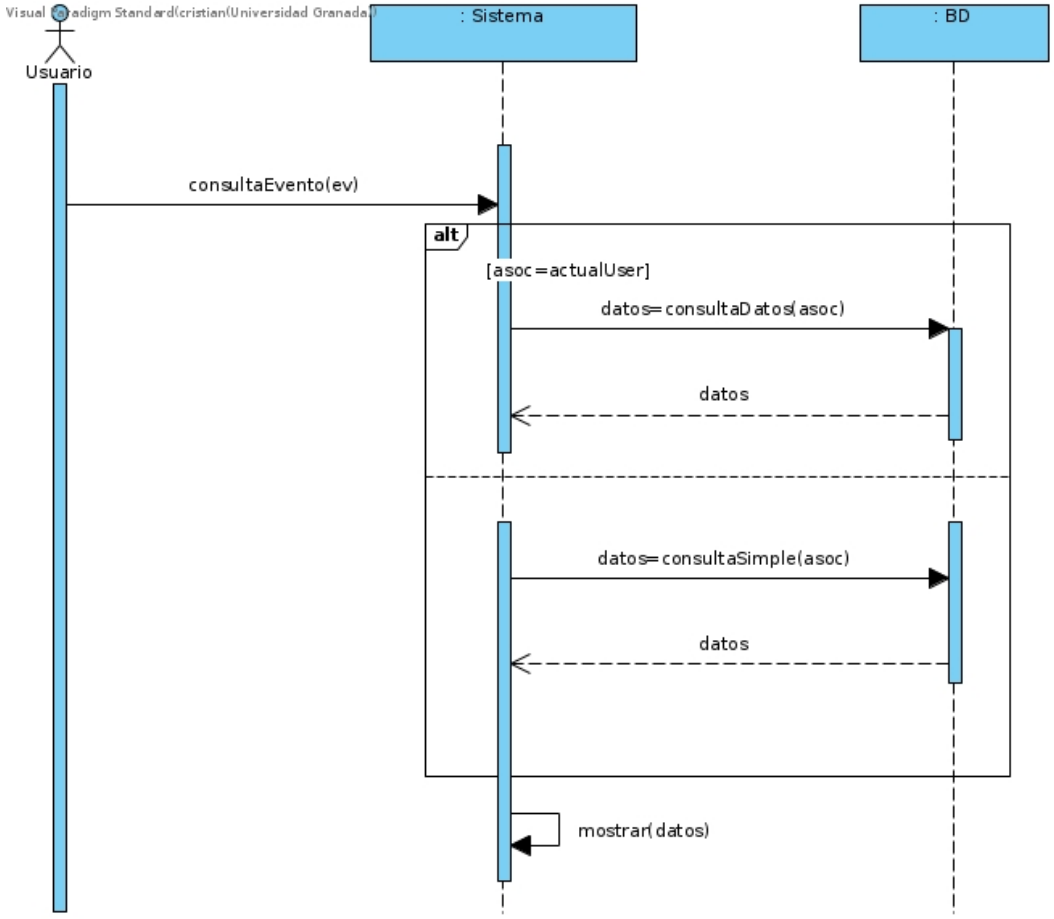


Diagrama 16: Diagrama de secuencia de Consulta listado de eventos Erasmus (CU-16)

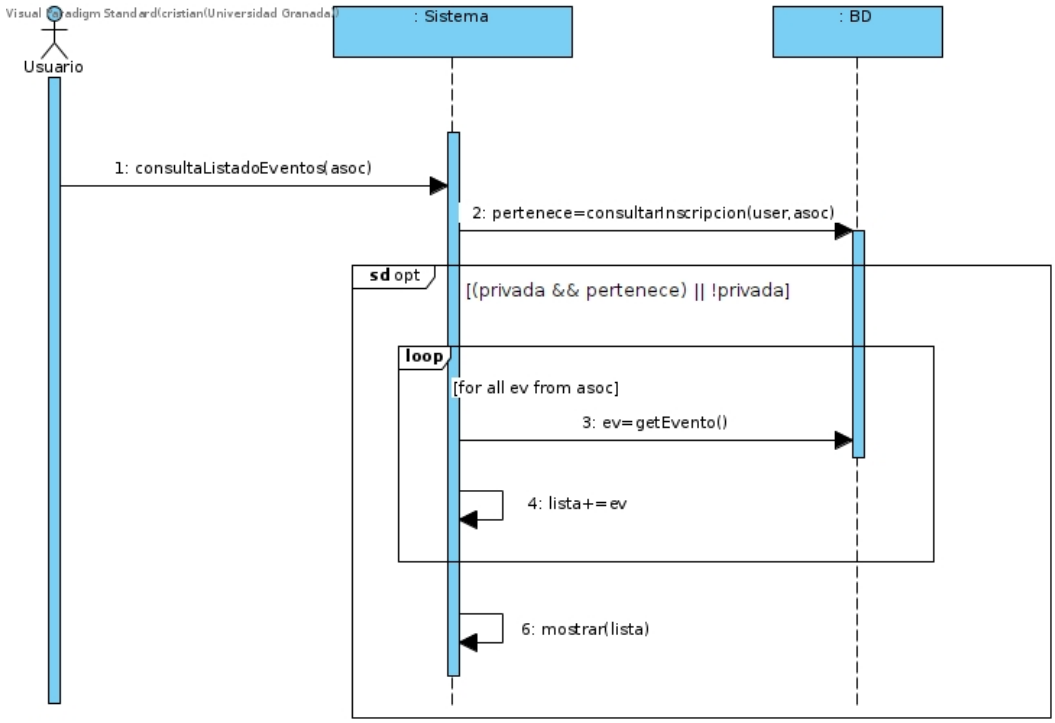


Diagrama 17: Diagrama de secuencia de Gestión de dirección de evento con Google Maps (CU-17)

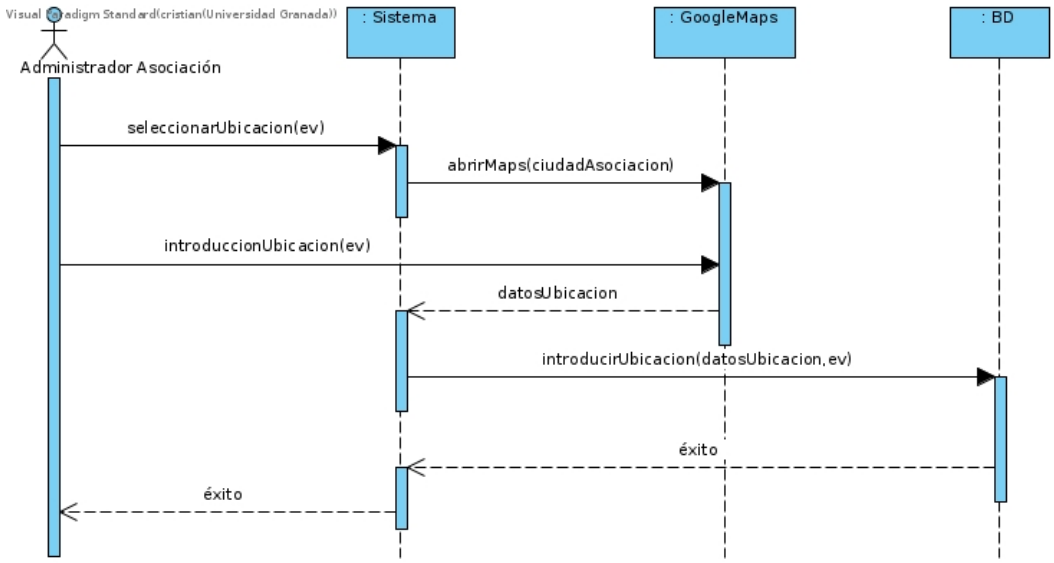


Diagrama 18: Diagrama de secuencia de Modificación de eventos Erasmus (CU-18)

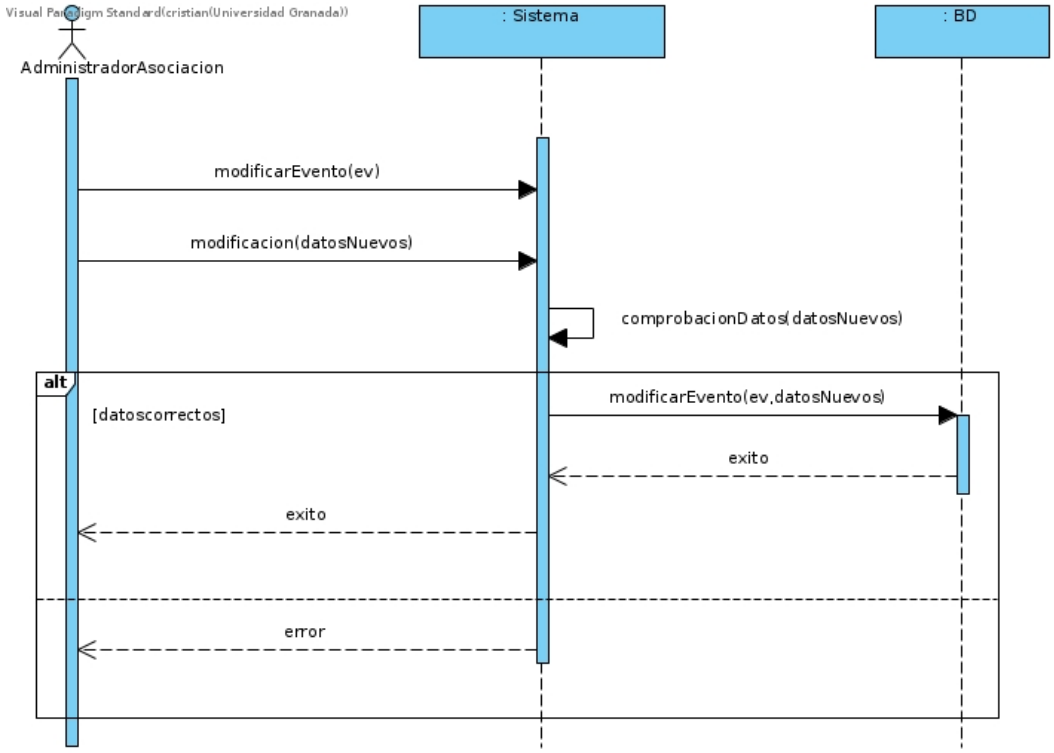


Diagrama 19: Diagrama de secuencia de Eliminación de eventos Erasmus (CU-19)

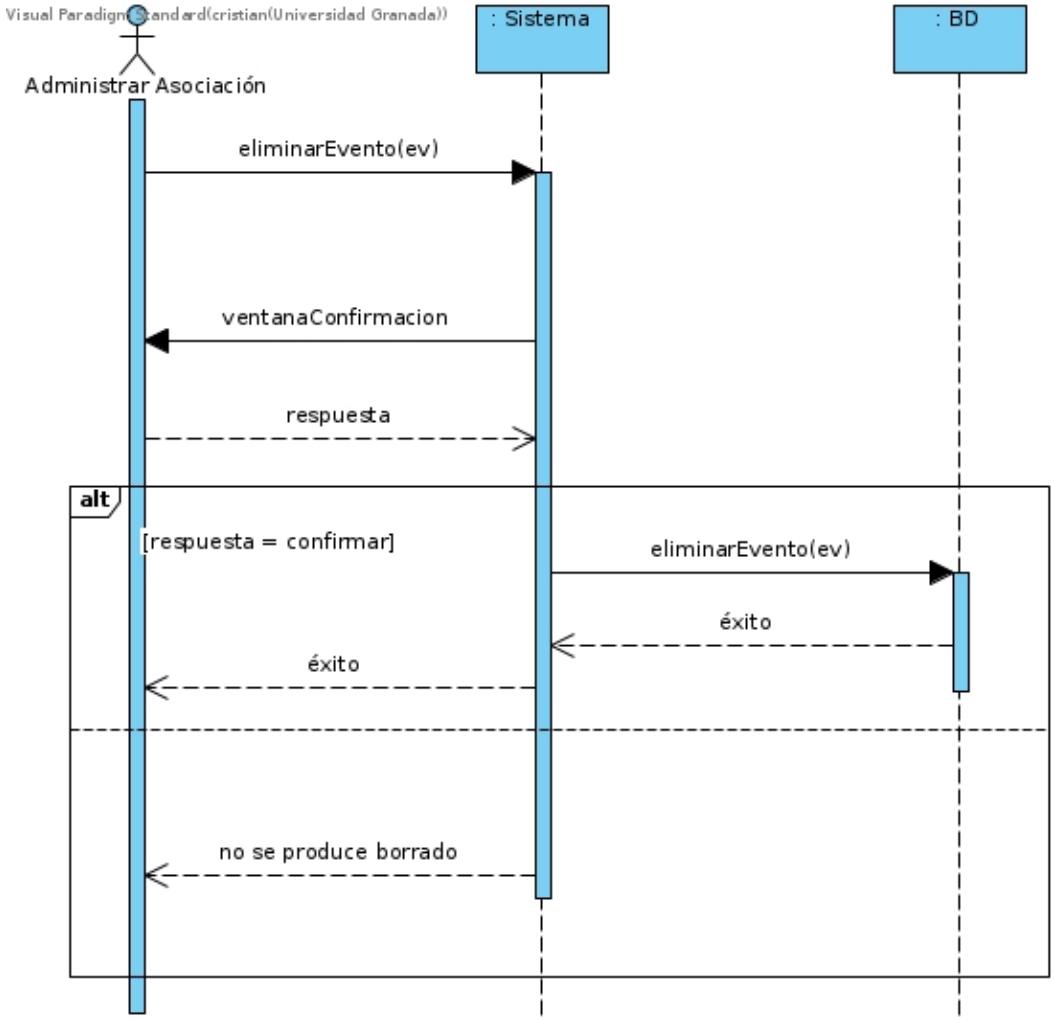


Diagrama 20: Diagrama de secuencia de Dar de baja a usuarios de asociaciones (CU-20)

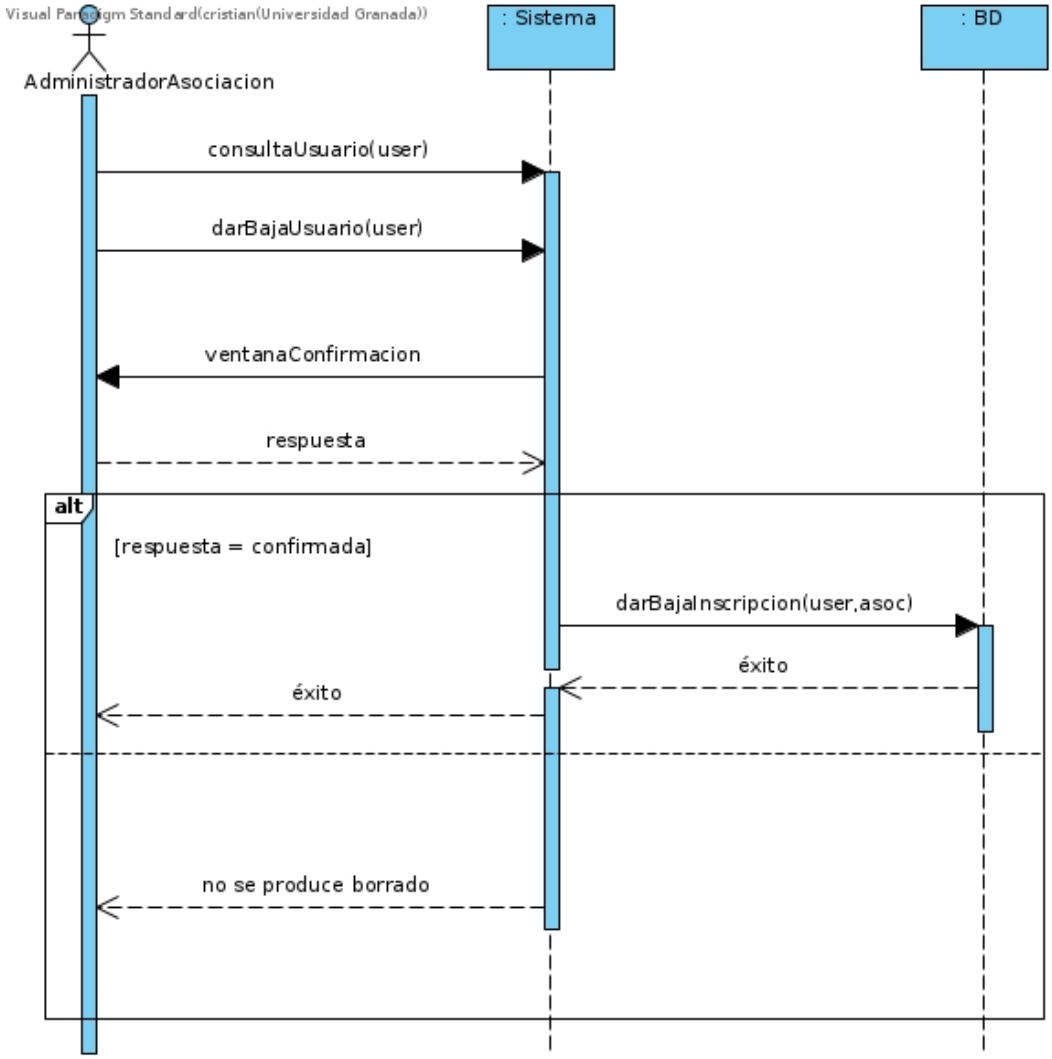


Diagrama 21: Diagrama de secuencia de Consulta listado de usuarios pertenecientes a una asociación (CU-21)

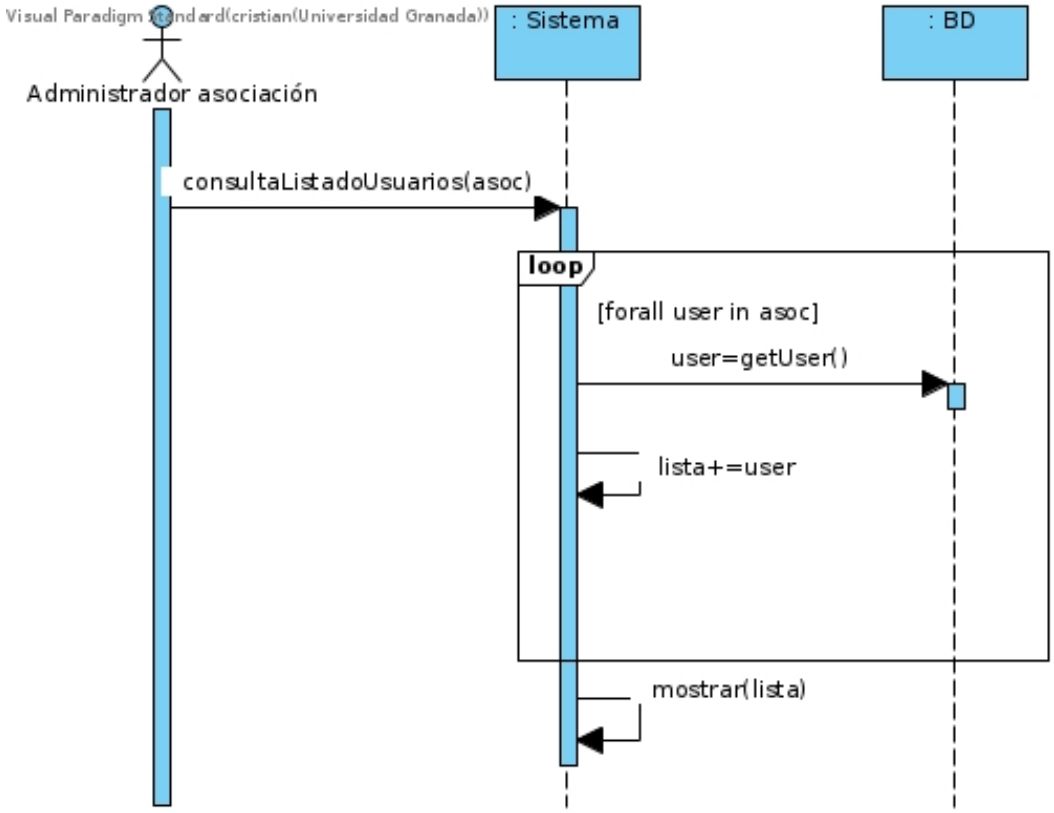


Diagrama 22: Diagrama de secuencia de Responder notificación sobre cambios a realizar (CU-22)

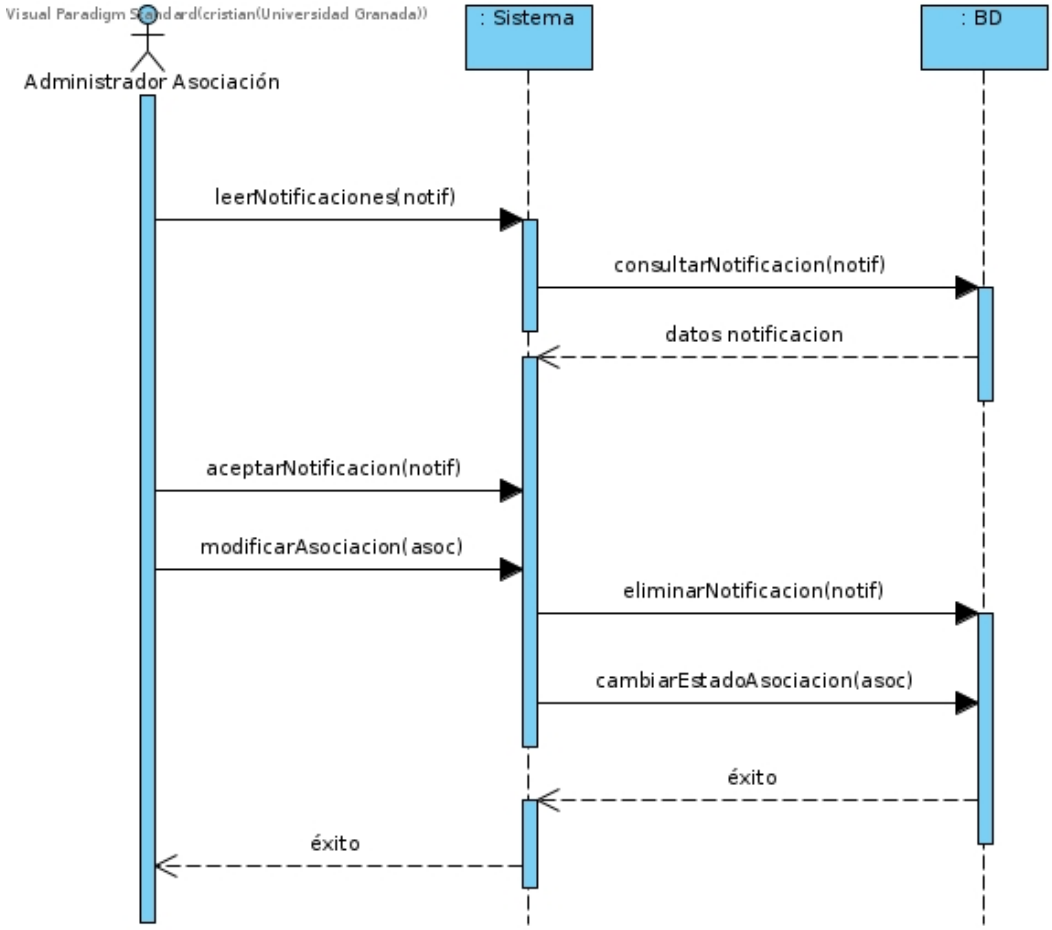


Diagrama 23: Diagrama de secuencia de Búsqueda de eventos ó asociaciones (CU-23)

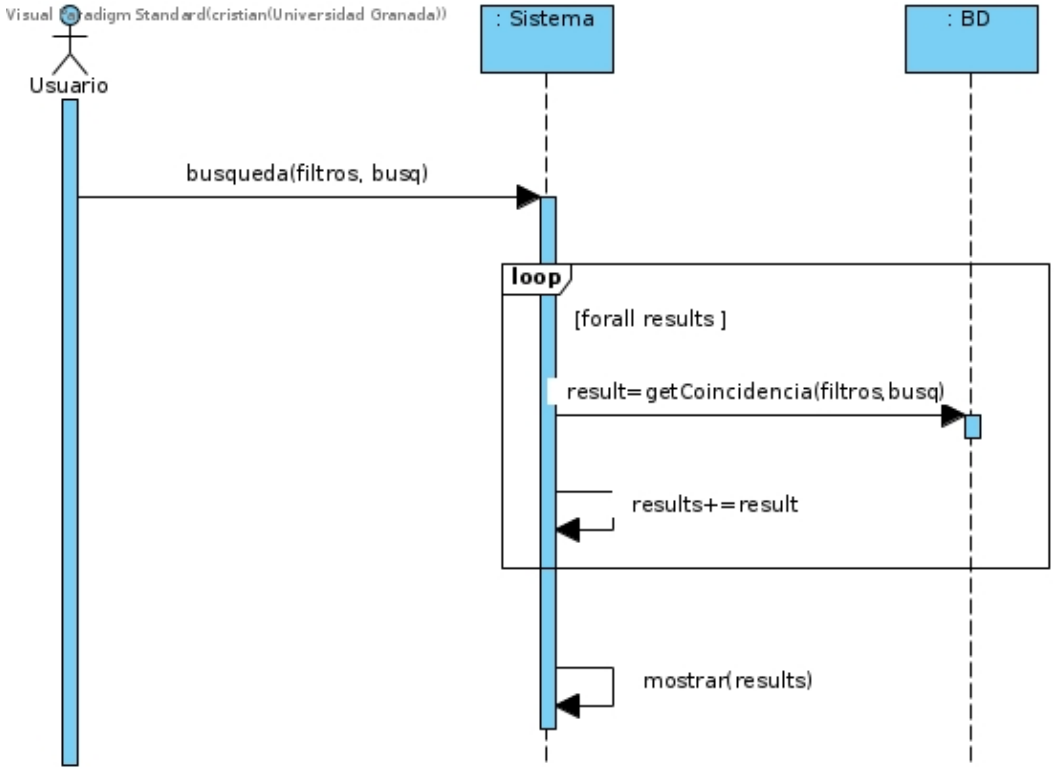


Diagrama 24: Diagrama de secuencia de Consulta de eventos en calendario (CU-24)

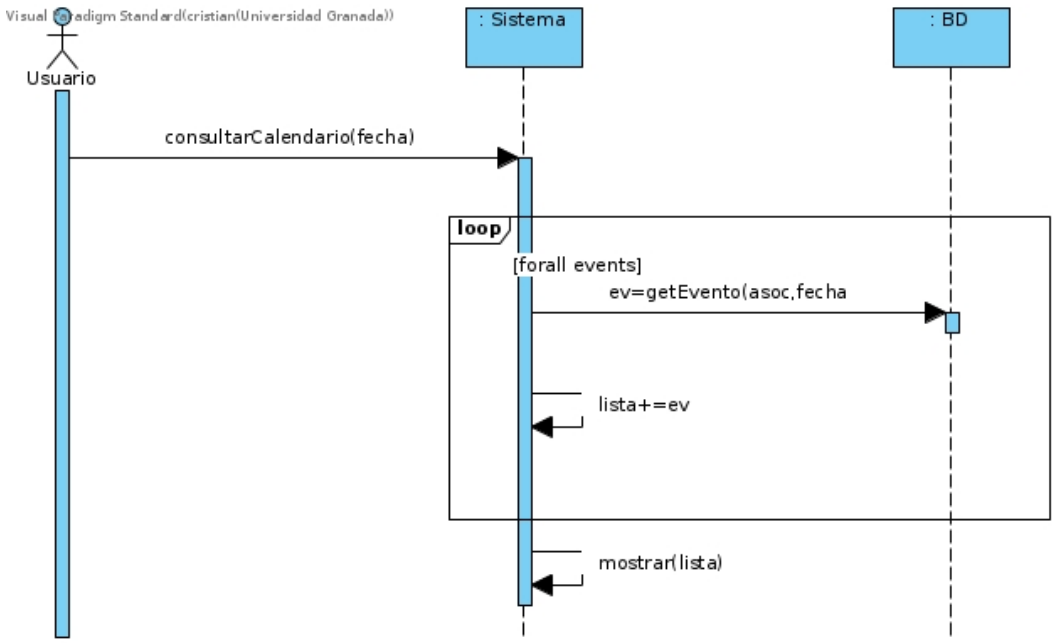


Diagrama 25: Diagrama de secuencia de Eliminar asociaciones Erasmus (CU-25)

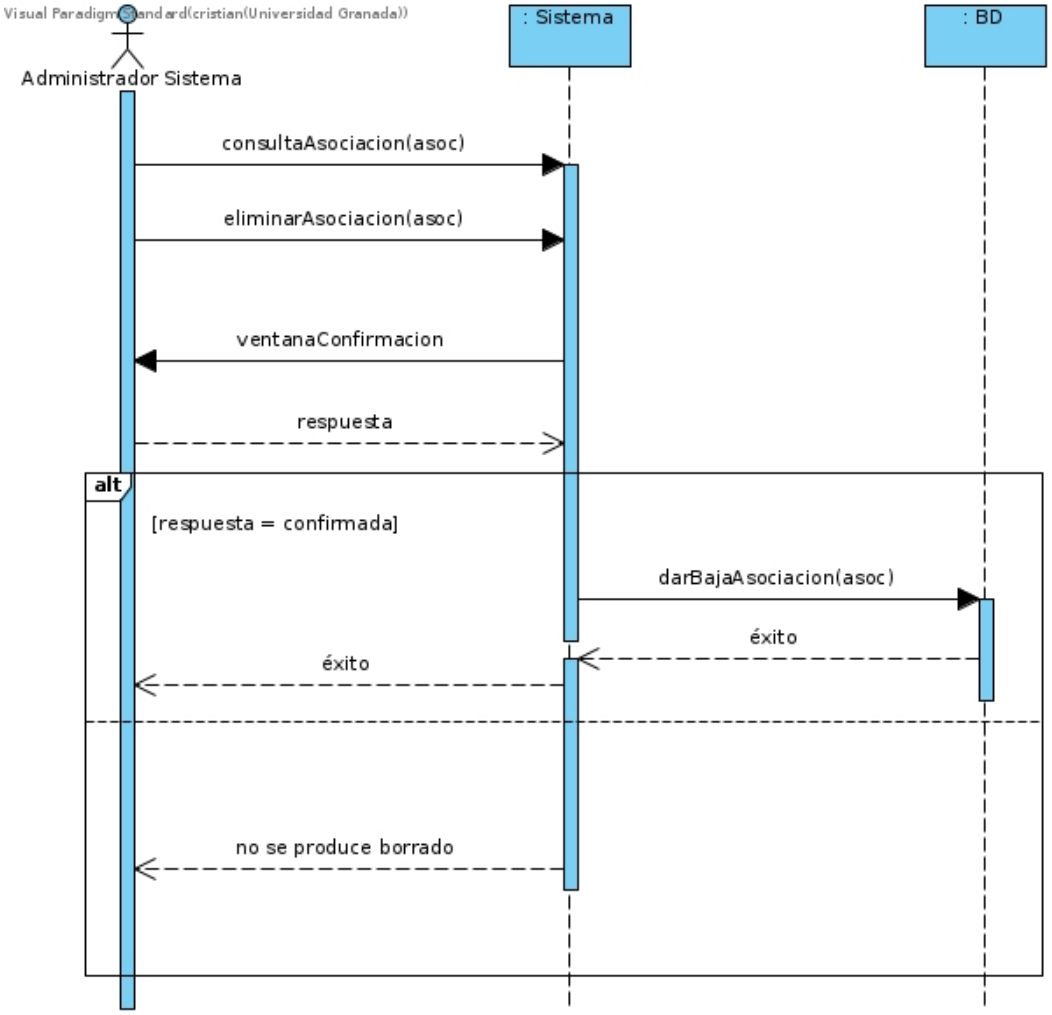


Diagrama 26: Diagrama de secuencia de Cambiar el estado de asociaciones Erasmus (CU-26)

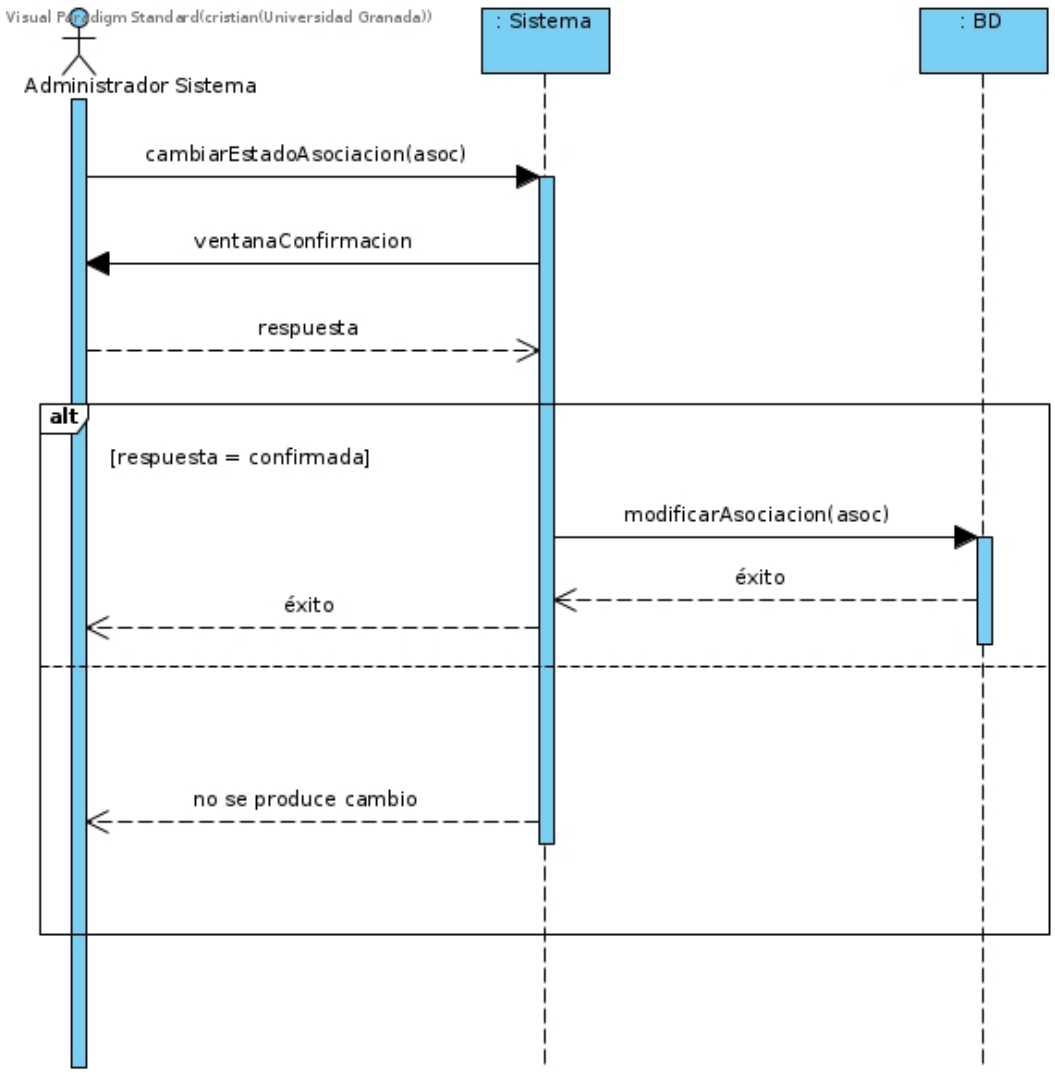
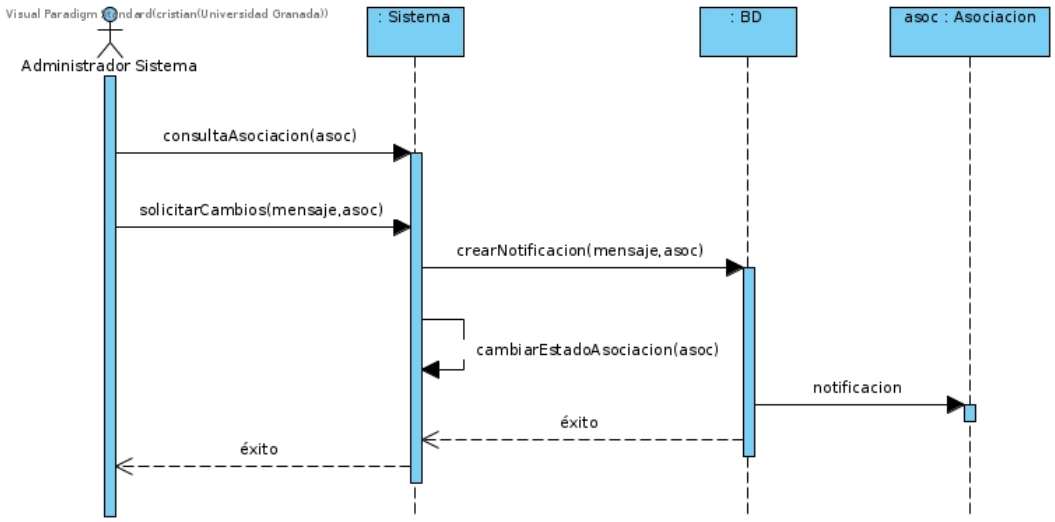


Diagrama 27: Diagrama de secuencia de Solicitar cambios a asociaciones Erasmus (CU-27)



Capítulo 4

Diseño

4.1. Diseño de la base de datos

4.1.1. Diagrama Entidad-Relación

El sistema debe integrarse y delimitarse correctamente. Surge así el modelo entidad-relación, donde los elementos que conforman un sistema son llamados entidades (rectángulo), y las interacciones entre ellos relaciones (rombos). Todas ellas cuentan con una serie de atributos, que serán traducidos en tablas y columnas de la base de datos. (Kendall & Kendall, 2005)

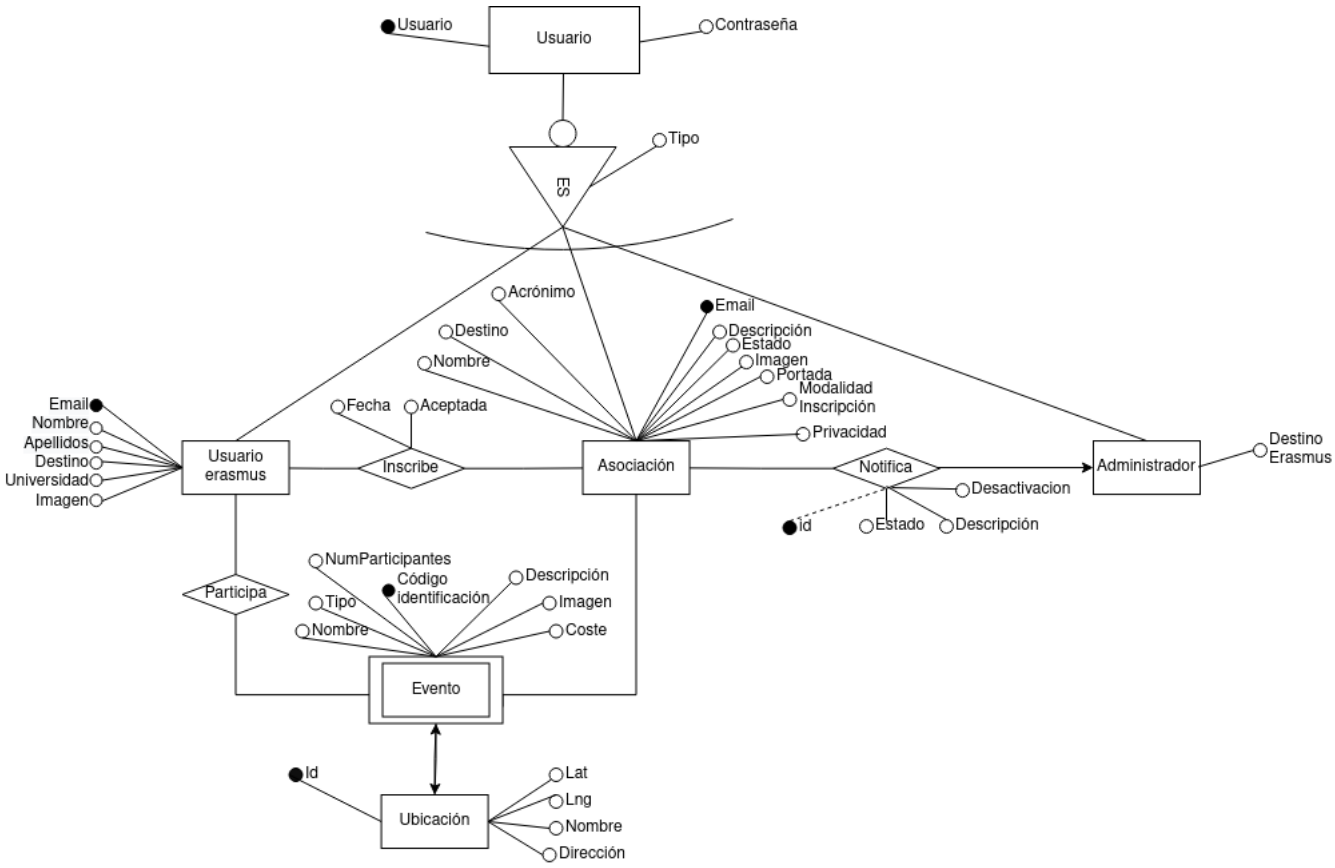


Figura 4.1: Diagrama Entidad-Relación del sistema

Como entidades del sistema representaremos a los diferentes actores mencionados durante todo el proceso de diseño:(I) Usuario Erasmus, (II) Asociación, (III) Administrador, y sin olvidar los (IV) Eventos, que organizará cada asociación.

Los usuarios podrán identificarse con nickname ó email y contarán con unos atributos mínimos de información para el perfil.

Las asociaciones se identificarán con código de asociación ó email. Se almacenará a su vez su estado (activa ó suspendida), la modalidad de inscripción (con solicitud ó libre) y además la privacidad para consultar sus eventos e información.

Por último, los eventos se identifican como entidad débil, ya que formarán parte siempre de una asociación. Para su identificación inequívoca requerirán pues del código de asociación y el código de evento.

En cuanto a relaciones, existirán tres: (I) Inscripción, (II) Notificación, (III) Participación. Los usuarios Erasmus podrán solicitar su inscripción a varias asociaciones, así como participar en numerosos eventos. Las asociaciones podrán inscribir a los estudiantes que lo soliciten, a su vez, podrán organizar una cantidad ilimitada de eventos. Finalmente, un administrador del sistema que encuentre algún inconveniente o error en alguna asociación, podrá hacérselo saber notificándola, con la descripción y fecha del problema.

4.1.2. Paso a tablas

Tablas normalizadas

Las presentes tablas definen la estructura de la base de datos del sistema. Han pasado por un proceso de normalización a **Forma Normal de Boyce-Codd** donde, tal y como señala Javier Quiroz (Quiroz, 2003), *"todos los atributos no llave de una relación dependen sólo y exclusivamente de la llave"*.

1. Usuario: Username (CP), Contraseña, Tipo

2. Destino Erasmus: Código destino (CP), País, Ciudad (CC), Imagen, Gif

3. Universidad: Código universidad (CP), Código destino (CE: Destino Erasmus), Nombre universidad

4. Administrador: Username (CE: Usuario) (CP), Destino (CE: Destino Erasmus) (CC)

5. Usuario Erasmus: Username (CE: Usuario), Nombre, Apellidos, Email (CC), Universidad destino (CE: Universidad), Imagen

6. Asociación: codAsociacion (CE: Usuario), Nombre, Email (CC) , Destino (CE: Destino Erasmus), Acrónimo (CC), Descripción, Imagen, Portada, Modalidad Inscripcion, Es-

tado, Privacidad

7. Ubicación: Id (CP), Nombre, Dirección, Lat, Lng

8. Evento: codEvento (CP) codAsociacion (CE: Asociación), Nombre, Tipo, Localización (CE: Ubicación), Num Participantes, Descripción, Imagen, Coste

9. Inscribe: Usuario (CE: Usuario Erasmus), codAsociacion (CE: Asociación) (CP), Fecha, Aceptada

10. Notifica: Id(CP), codAsociacion (CE: Asociación), admin (CE: Usuario), Fecha, Estado, Descripción, Desactivación

11. Participa: Usuario (CE: Usuario Erasmus), codEvento (CE: Evento) (CP)

Proceso de normalización

"Las formas normales son conjuntos de criterios que utilizamos para «normalizar» (es decir, mejorar la estructura) de las bases de datos." (Jaque, 2015) Estas han sido aplicadas en cada una de las entidades y relaciones generadas desde el Diagrama Entidad-Relación del sistema siguiendo el proceso a continuación:

1. Usuario: Username (CP), Contraseña, Tipo

1FN - Se cumple, todos los atributos son dependientes de la clave primaria. 2FN - Se cumple, no existen dependencias parciales, todos los atributos dependen de las claves candidatas.
3FN - Se cumple, no existen dependencias transitivas, las columnas dependen solo de las claves.
FNBC - Se cumple, pues no existen claves compuestas.

2. Usuario Erasmus: Username (CE: Usuario), Nombre, Apellidos, Email (CC), Destino Erasmus, Universidad, Imagen

1FN - No se cumple pues con los campos Destino Erasmus y Universidad estaríamos repitiendo información, hemos de generar otra tabla para almacenar estos. De manera que Usuario sólo haga referencia a la universidad.

2. Usuario Erasmus: Username (CE: Usuario), Nombre, Apellidos, Email (CC), Universidad destino (CE: Universidad), Imagen

2FN - Se cumple, no existen dependencias parciales, todos los atributos dependen

de las claves candidatas.
3FN - Se cumple, no existen dependencias transitivas, las columnas dependen solo de las claves.
FNBC - Se cumple, pues no existen claves compuestas.

3. Asociación: codAsociacion (CE: Usuario), Nombre, Email (CC), Destino (CE: Destino Erasmus), Acronimo, Contraseña, Descripción, Imagen, Portada, Modalidad Inscripcion, Estado, Privacidad

2FN - No existen dependencias parciales, ya que la clave primaria es solo un atributo.
3FN - No se cumple. Destino y Acrónimo tienen dependencias transitivas, pues: Destino Erasmus, Acrónimo → cualquier atributo.

Para solucionar esto hacemos de estos atributos una clave candidata.

3. Asociación: codAsociacion (CE: Usuario), Nombre, Email (CC), Destino (CE: Destino Erasmus), Acrónimo (CC), Descripción, Imagen, Portada, Modalidad Inscripcion, Estado, Privacidad

FNBC - Se cumple, pues existen claves compuestas pero no comparten atributos.

4. Administrador: Username (CE: Usuario) (CP), Destino (CE: Destino Erasmus) (CC)

1FN - Se cumple, todos los atributos son dependientes de la clave primaria.
2FN - Se cumple, no existen dependencias parciales, pues ambas son claves
3FN - Se cumple, no existen dependencias transitivas pues no existen atributos no clave.
FNBC - Se cumple, puesto que ambos son claves

5. Ubicación: Id (CP), Nombre, Dirección, Lat, Lng

1FN - Se cumple, todos los atributos son dependientes de la clave primaria.
2FN - Se cumple, no existen dependencias parciales, todos los atributos dependen de las claves candidatas.
3FN - Se cumple, no existen dependencias transitivas entre los atributos no clave.
FNBC - Se cumple, el único atributo determinante es la clave candidata, no existen dependencias del tipo Atributo no primo → Atributo primo.

6. Evento: codEvento (CP), codAsociacion (CE: Asociación), Nombre, Tipo, Localización (CE: Ubicación), Num Participantes, Descripción, Imagen, Coste, Fecha

1FN - Se cumple, todos los atributos son dependientes de la clave primaria.
2FN - Se cumple, no existen dependencias parciales, todos los atributos dependen de las claves candidatas.

3FN - Se cumple, no existen dependencias transitivas entre los atributos no clave.
FNBC - Se cumple, el único atributo determinante es la clave candidata, no existen dependencias del tipo Atributo no primo → Atributo primo.

7. Inscribe: Usuario (CE: Usuario Erasmus), codAsociacion (CE: Asociación) (CP), Fecha, Aceptada

1FN - Se cumple, todos los atributos son dependientes de la clave primaria.
2FN - Se cumple, no existen dependencias parciales, todos los atributos dependen de las claves candidatas.
3FN - Se cumple, no existen dependencias transitivas entre los atributos no clave.
FNBC - Se cumple, los únicos atributos determinantes son los de la clave candidata, no existen dependencias del tipo Atributo no primo → Atributo primo.

8. Notifica: Id(CP), codAsociacion (CE: Asociación), admin (CE: Usuario), Fecha, Estado, Descripción, Desactivación

1FN - Se cumple, todos los atributos son dependientes de la clave primaria.
2FN - Se cumple, no existen dependencias parciales, todos los atributos dependen de las claves candidatas.
3FN - Se cumple, no existen dependencias transitivas entre los atributos no clave.
FNBC - Se cumple, el único atributo determinante es la clave candidata, no existen dependencias del tipo Atributo no primo → Atributo primo.

9. Participa: Usuario (CE: Usuario Erasmus), codEvento (CE: Evento) (CP)

Todos los atributos de la tabla Participa conforman la clave primaria. No existen dependencias parciales, por lo que ya la encontramos en FNBC.

10. Destino Erasmus: Código destino (CP), País, Ciudad, Universidad, Imagen, Gif

1FN - No se cumple, pues el campo Universidad causaría la repetición de todo lo demás ante cambios. Dividimos la tabla con otra de Universidades.

10. Destino Erasmus: Código destino (CP), Pais, Ciudad, Imagen, Gif

2FN - No se cumple, pues se podría identificar un destino por su ciudad y país.

10. Destino Erasmus: Código destino (CP), País, Ciudad (CC), Imagen, Gif

3FN - Se cumple, pues no existen dependencias transitivas. Las columnas dependen solo de la clave primaria.
FNBC - Se cumple, pues no existen claves compuestas.

11. Universidad: Código universidad (CP), Código destino (CE: Destino Erasmus), Nombre universidad

1FN - Se cumple, todos los atributos son dependientes de la clave primaria.
2FN - Se cumple, no existen dependencias parciales, todos los atributos dependen de las claves candidatas.
3FN - Se cumple, no existen dependencias transitivas entre los atributos no clave.
FNBC - Se cumple, el único atributo determinante es la clave candidata, no existen dependencias del tipo Atributo no primo $\rightarrow$ Atributo primo.

4.2. Diseño visual del sistema

Lo que serán las interfaces finales del sistema a desarrollar vendrán seguidas por las siguientes direcciones y sketches. Algunas de las mejores prácticas a tener en cuenta, según Developer Apple (s.f.), son las siguientes:

- *Aprovechar las pantallas grandes para presentar más contenido, mientras se mantiene una densidad de información cómoda que no hace que el usuario se esfuerce por ver el contenido que desea.*
- *Permitir que el usuario redimensione, oculte, muestre y mueva sus ventanas para que se ajusten a su estilo de trabajo y configuración del dispositivo, y admita el modo de pantalla completa para ayudar al usuario a concentrarse en un contexto sin distracciones.*
- *Usar la barra de menú para brindar a los usuarios fácil acceso a todas las funciones que necesitan para realizar acciones en la aplicación.*

4.2.1. Guidelines

Fuentes

La fuente principal considerada es **Sansita**. Se trata de un diseño sencillo y agradable, que utilizaremos en negrita e italic para resaltar la información más importante.

Regular 400

Almost before we knew it, we had left the ground.

Regular 400 italic

Almost before we knew it, we had left the ground.

Bold 700

Almost before we knew it, we had left the ground.

Bold 700 italic

Almost before we knew it, we had left the ground.

Además, una fuente secundaria será **Noto Sans**. Un diseño simple y más relajado, que utilizaremos para el cuerpo e información menos destacada.

Regular 400

Whereas recognition of the inherent dignity.

Regular 400 Italic

Whereas recognition of the inherent dignity.

Bold 700

Whereas recognition of the inherent dignity.

Bold 700 Italic

Whereas recognition of the inherent dignity.

Iconos

Los iconos serán simples y muy fácil de reconocer, para posibilitar el fácil uso de la aplicación y agrado. Con colores blanco y negro, no muy detallados y su reconocimiento será obvio.

Paleta de colores

#ffffff	#d1c7c3	#8b8b8b	#3c3c3c	#000000

La paleta de colores se trata de una gama de grises, simple y elegante, que garantiza un confort visual único.

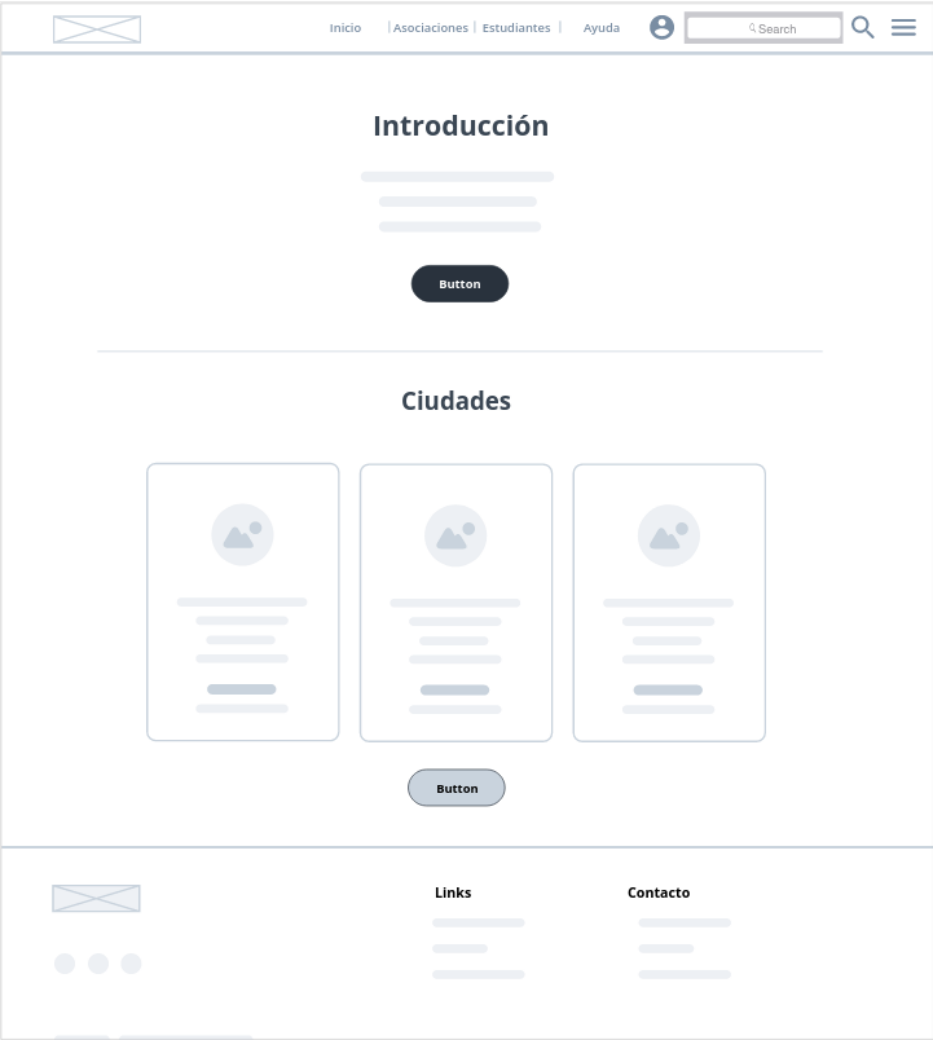
4.2.2. Prototipado

Un prototipo es un diseño a escala del producto usado para su evaluación Las herramientas de prototipado de IU permiten crear bocetos para analizar propiedades sobre la organización de información, navegación y diseño. (Cao, Ellis & Khachatryan, s.f.)

Mockups Low-fidelity

Los siguientes Mockups LoFi definirán las estructuras de las distintas interfaces con las que contará la aplicación. Son simples bocetos en los que se observará sobre todo el diseño y el flujo de control, sin entrar en detalles.

Interfaz LoFi 1: Landing page



En la primera interfaz de la web se mostrará información general sobre la utilización de esta, junto a destinos que ya están usando la aplicación. La barra de navegación inicial cuenta con información para Asociaciones o Estudiantes que quieran registrarse (RF-01, RF-12). Además de la opción de iniciar sesión (RF-02) y la barra de búsqueda. Esta vista a su vez será mostrada al cerrar sesión (RF-03).

Interfaz LoFi 2: Página principal - Sesión iniciada

Cuando iniciamos sesión (RF-02), la página de inicio pasa a ser específica según destino. Mostrará eventos relevantes además en la ciudad y permitirá además mostrar lo que está ocurriendo hoy, en el fin de semana, durante la semana o durante el mes.

La barra de navegación pasa a ser común con la sesión iniciada. El menú de Asociaciones redirige a una página específica de asociaciones según el destino (RF-13), al igual que Eventos con los respectivos de la ciudad (RF-25). La barra de búsqueda permitirá encontrar asociaciones o eventos según lo introducido (RF-24). El menú desplegable contará con más opciones del usuario, como notificaciones (RF-23), eventos en los que se participa (RF-17), asociaciones a las que se pertenece, miembros de la asociación (RF-22), etc...

Interfaz LoFi 3: Página de Asociaciones en destino

Las asociaciones de la ciudad serán mostradas en la siguiente página, desplegadas de forma que se muestre información de ellas junto a algunos de sus eventos próximos. Se permitirá la redirección a sus perfiles (RF-13) o directamente a los eventos en cuestión (RF-16).

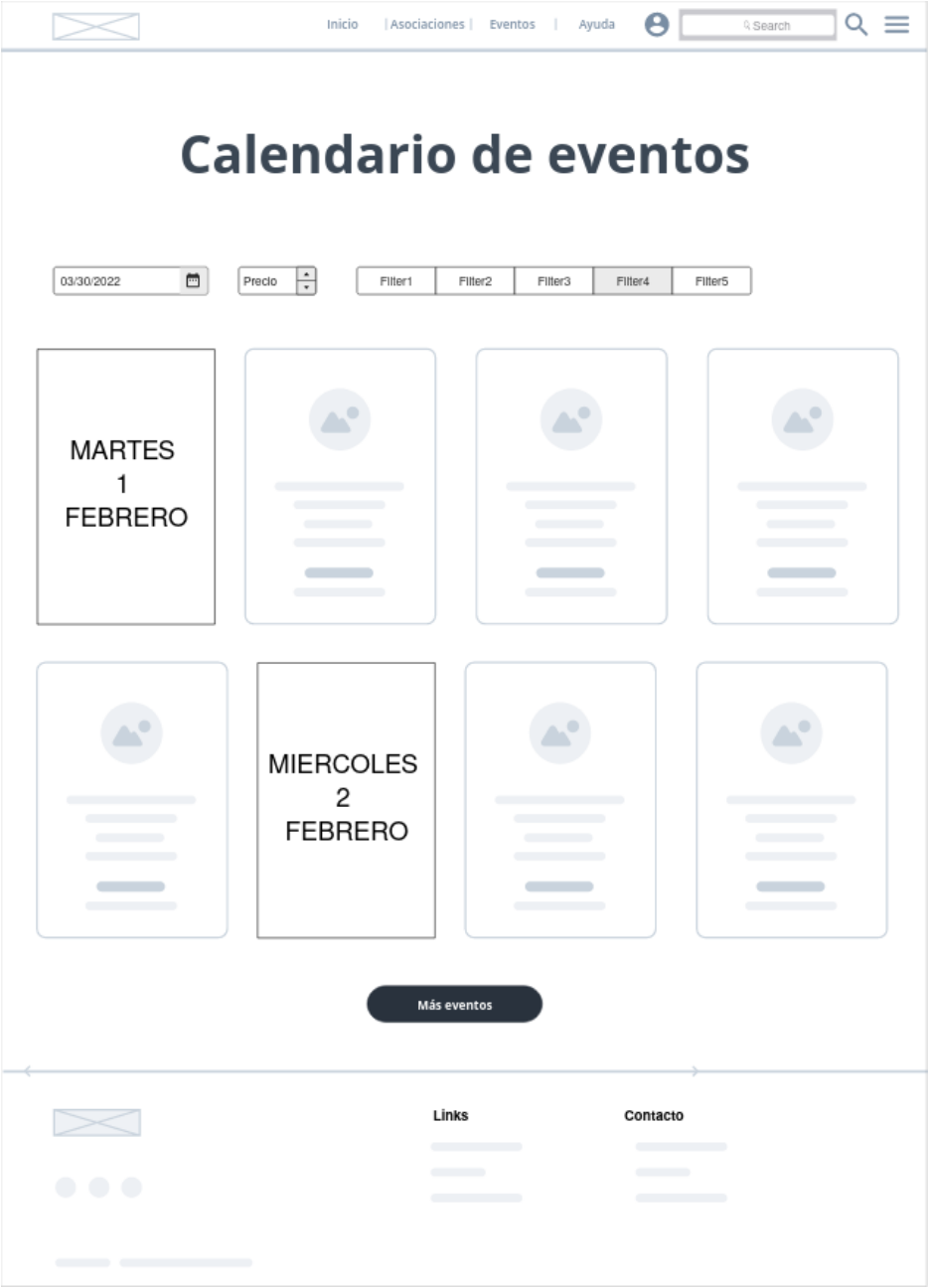
Interfaz LoFi 4: Perfil de Asociación (Vista de estudiante)

La asociación contará con un perfil donde incluirá su información e imágenes representativas. Además de un calendario que mostrará las fechas marcadas por eventos (RF-25). Estos últimos serán detallados más abajo, en una zona de eventos próximos (RF-16). Asimismo, una lista de los participantes actuales de la asociación será desplegada (RF-22).

Interfaz LoFi 5: Perfil de Asociación (Vista de administrador)

La vista del perfil de asociación desde la cuenta de administrador será distinta, pues contará con diferentes opciones de edición (RF-19) y agregación de eventos (RF-15), modificación de la asociación (RF-14), borrado (RF-26), además de la posibilidad de dar de baja a miembros de la asociación (RF-21).

Interfaz LoFi 6: Página de Calendario de Eventos



Los eventos en la ciudad se mostrarán en forma de calendario, con varios filtros a aplicar y ordenados por fecha (RF-25).

Interfaz LoFi 7: Página de Evento (Vista de estudiante)

Para cada evento se cuenta con su página específica (RF-16), con toda su información. Además, para completar el espacio, otros eventos relacionados y de la misma asociación serán ofrecidos (RF-17).

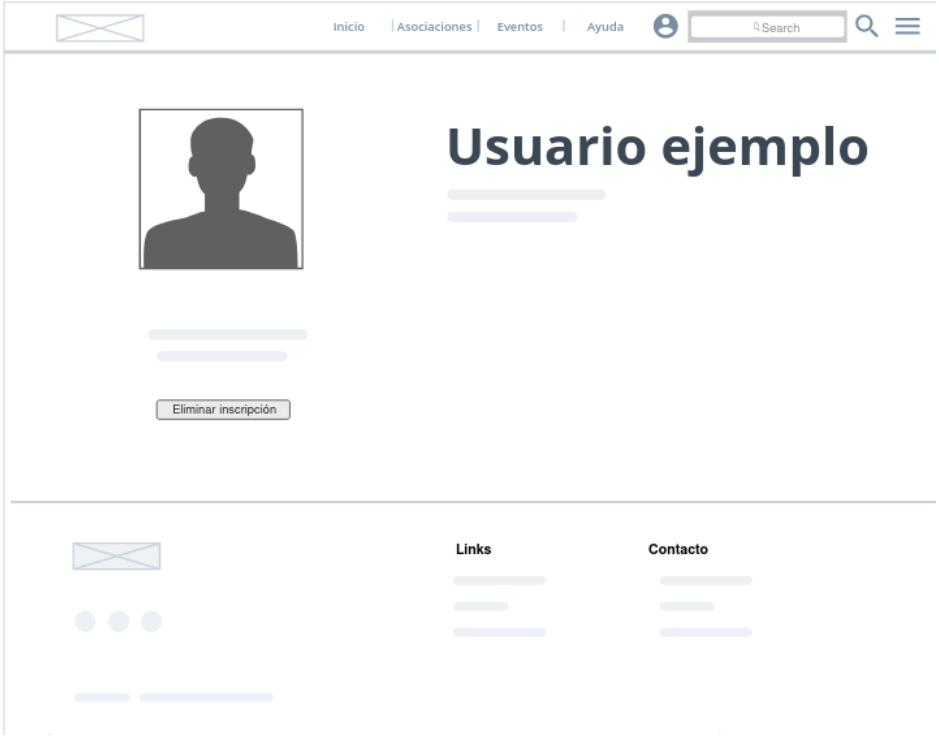
Interfaz LoFi 8: Página de Evento (Vista de administrador)

El administrador de la asociación contará con opciones de edición del evento seleccionado (RF-19), además de la opción de cancelarlo (RF-20).

Interfaz LoFi 9: Perfil de usuario (Vista de estudiante)

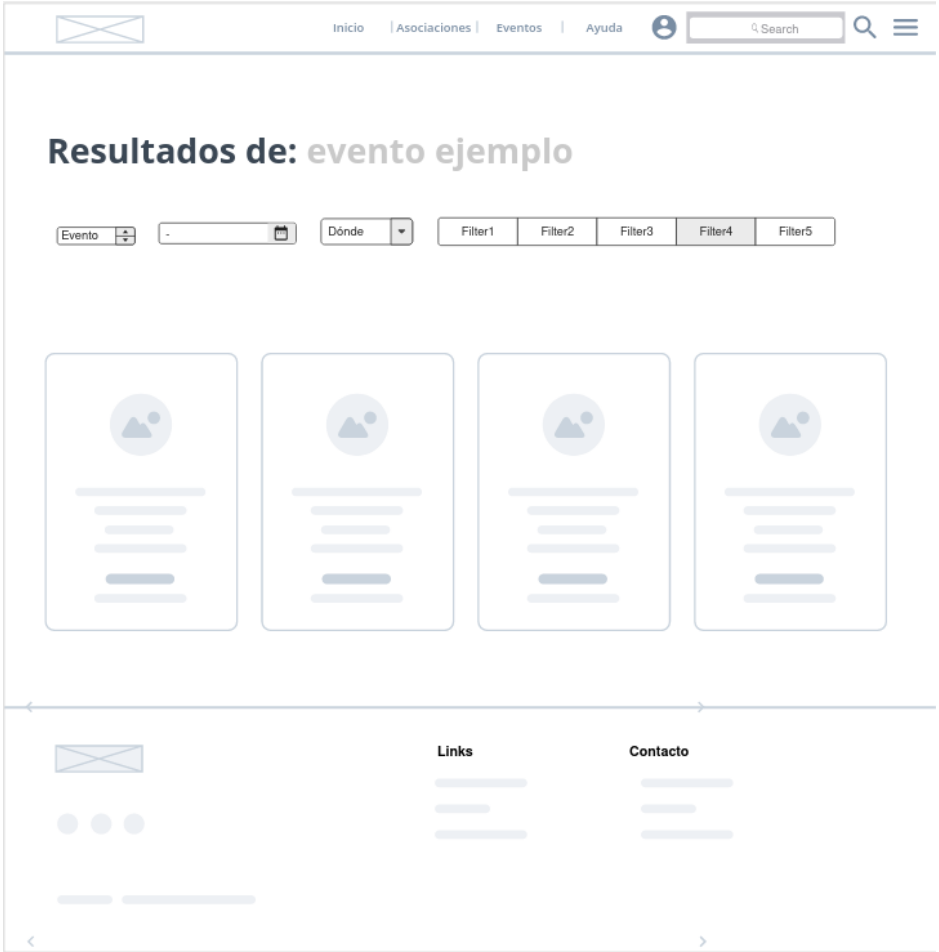
La información de cada usuario se desplegará en una página para ello, junto con las asociaciones a las que pertenece (RF-13) y los eventos a los que asistirá (RF-17).

Interfaz LoFi 10: Perfil de usuario (Vista pública y de administrador)



El perfil público de usuario no contendrá información sobre asociaciones o eventos. Si es visitado por el administrador de una asociación a la que pertenece, este podrá eliminar su inscripción (RF-21).

Interfaz LoFi 11: Menú de búsqueda

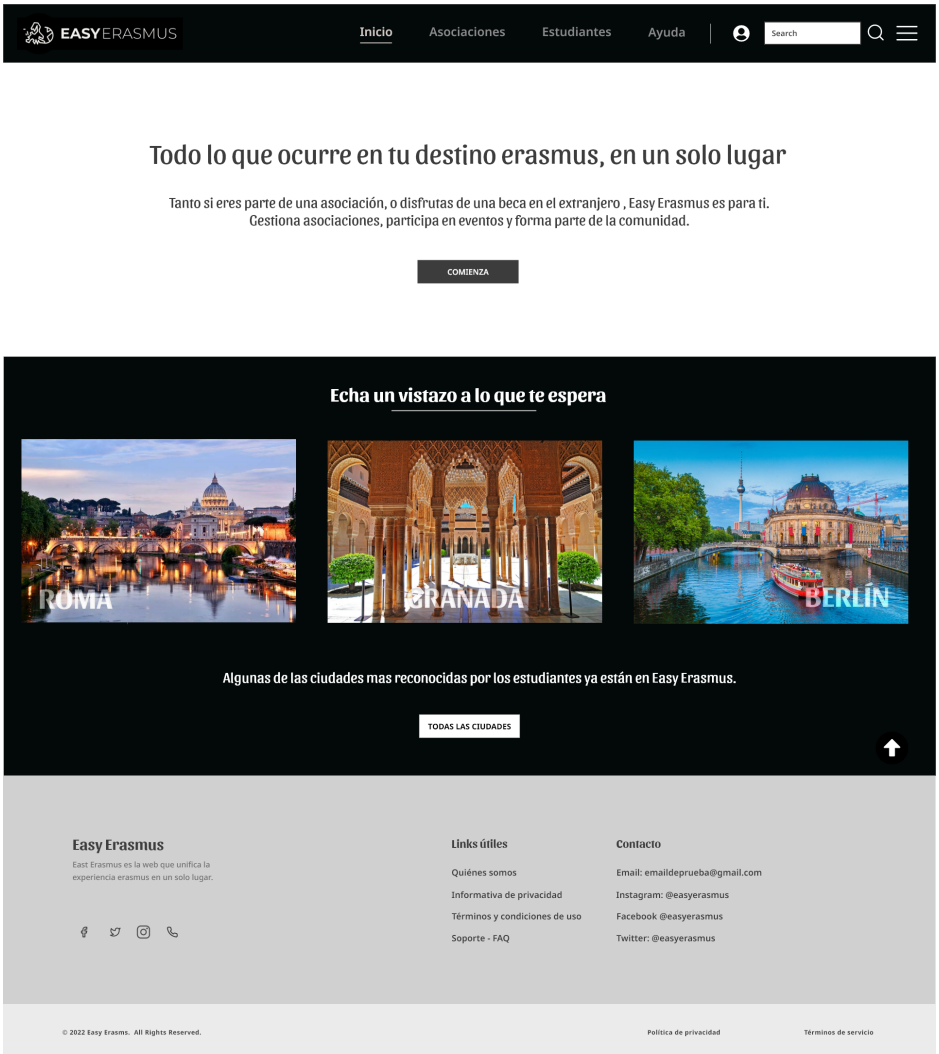


El menú de búsqueda se desplegará tras realizar una consulta desde la barra de navegación. Contendrá varios filtros entre cuales destacan el tipo de búsqueda (evento, asociación o ambos), fecha y lugar. Los resultados se mostrarán con su imagen e información relacionada (RF-24).

Mockups High-fidelity

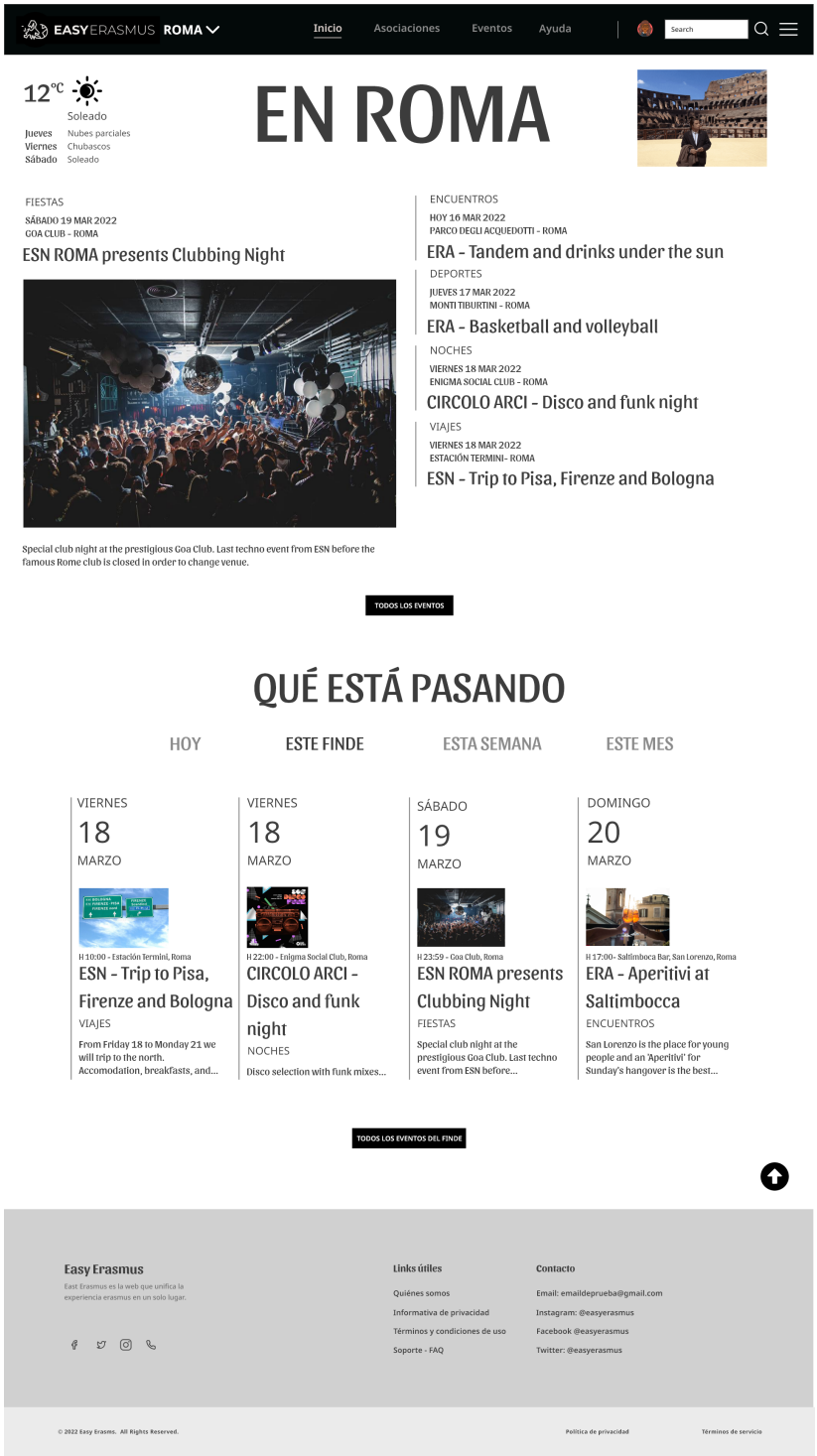
Los siguientes Mockups HiFi definirán una visión más realista de la aplicación desde la perspectiva del usuario Erasmus. Son diseños detallados en los que se observará cómo debería lucir el resultado final con datos de ejemplo.

Interfaz HiFi 1: Landing Page



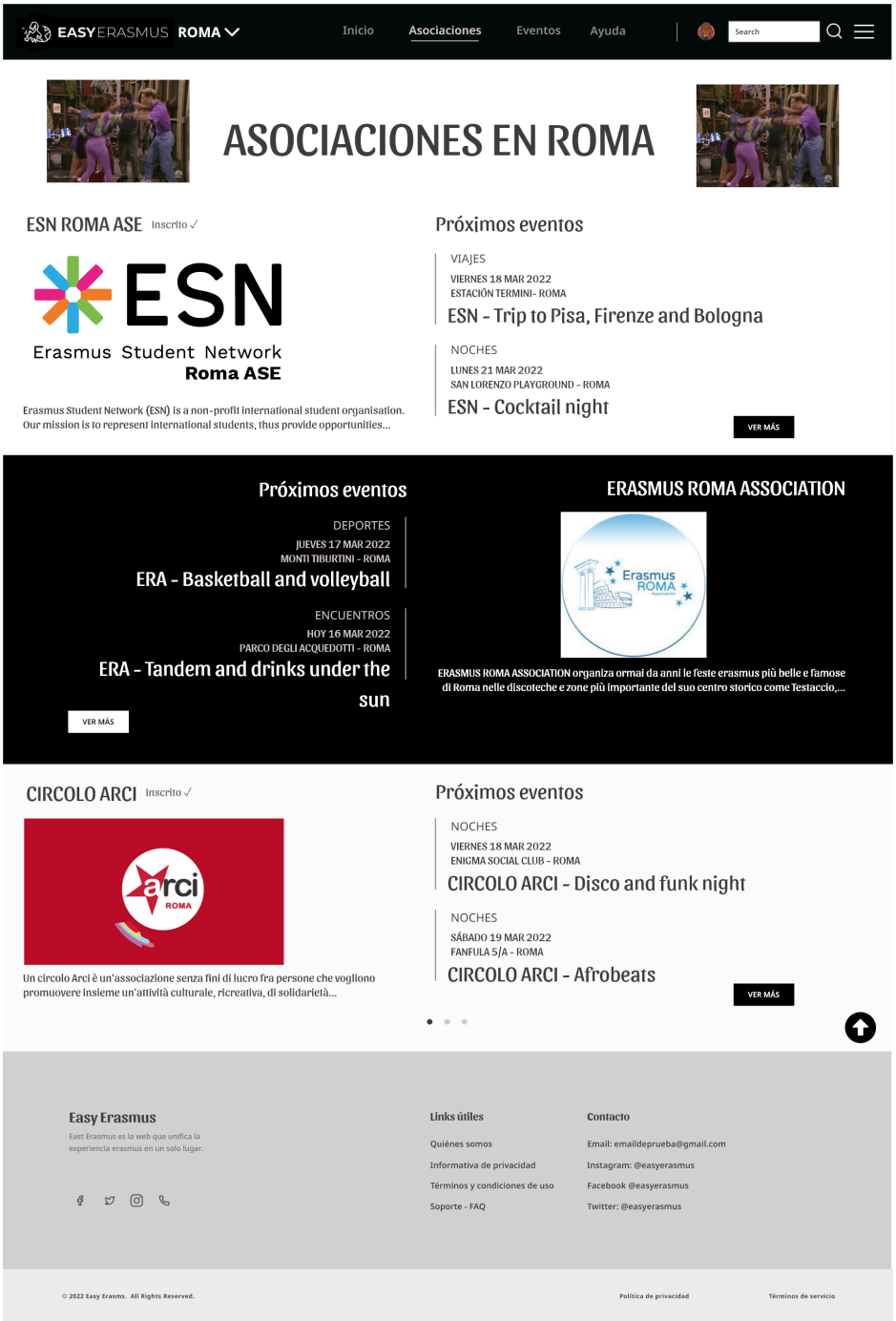
La página de inicio, con sesión cerrada (RF-03), mostrará algunas de las ciudades ya disponibles en la aplicación. Además de vínculos que lleven al registro en la web (RF-01, RF-12), para hacer uso de las demás funcionalidades.

Interfaz HiFi 2: Página de inicio por destino



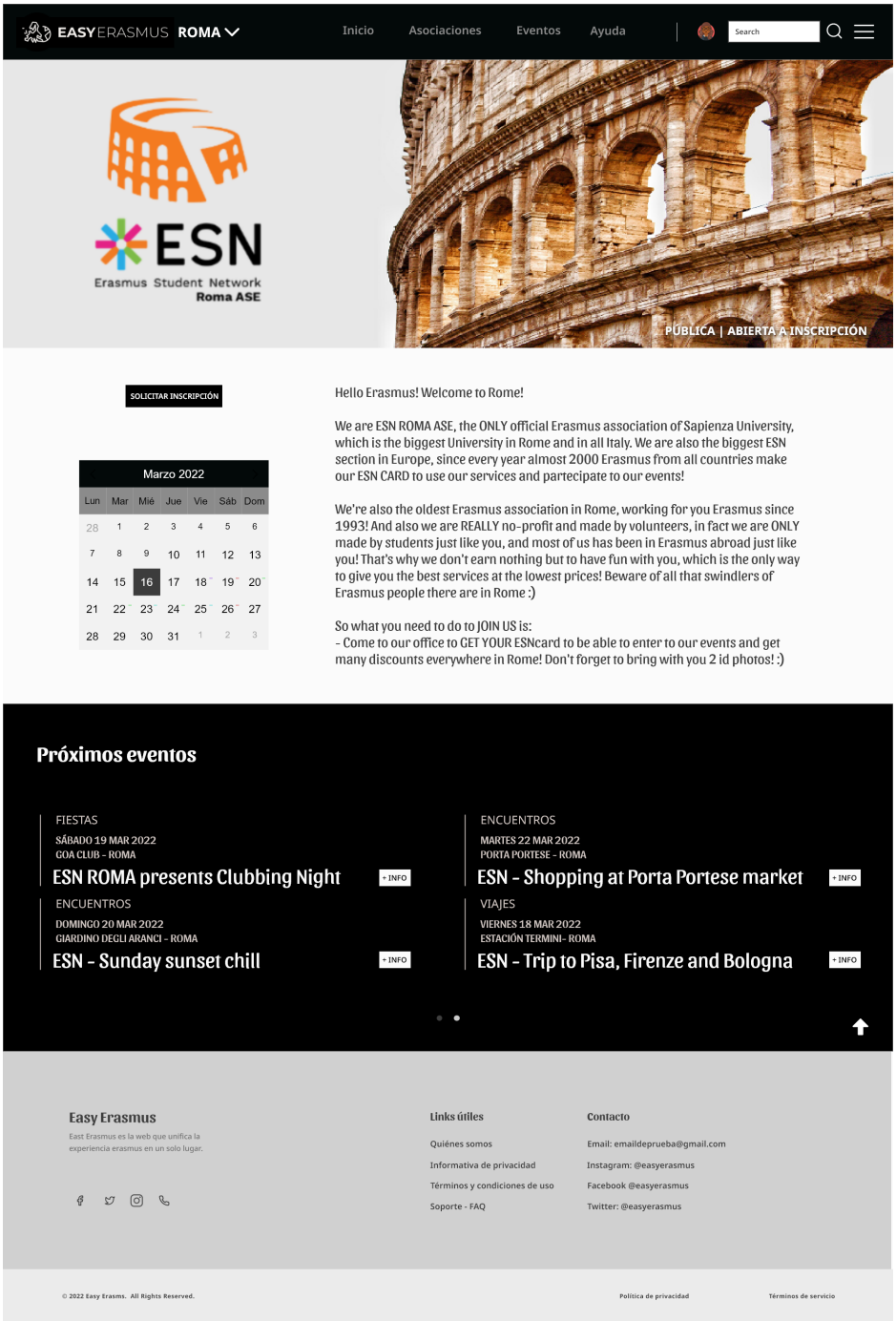
La página de inicio tras iniciar sesión (RF-02) será personalizada según el destino, mostrará los eventos destacados que ocurrirán en los próximos días, además de contar con otro despliegue por día, fin de semana, semana o mes (RF-17).

Interfaz HiFi 3: Asociaciones por destino



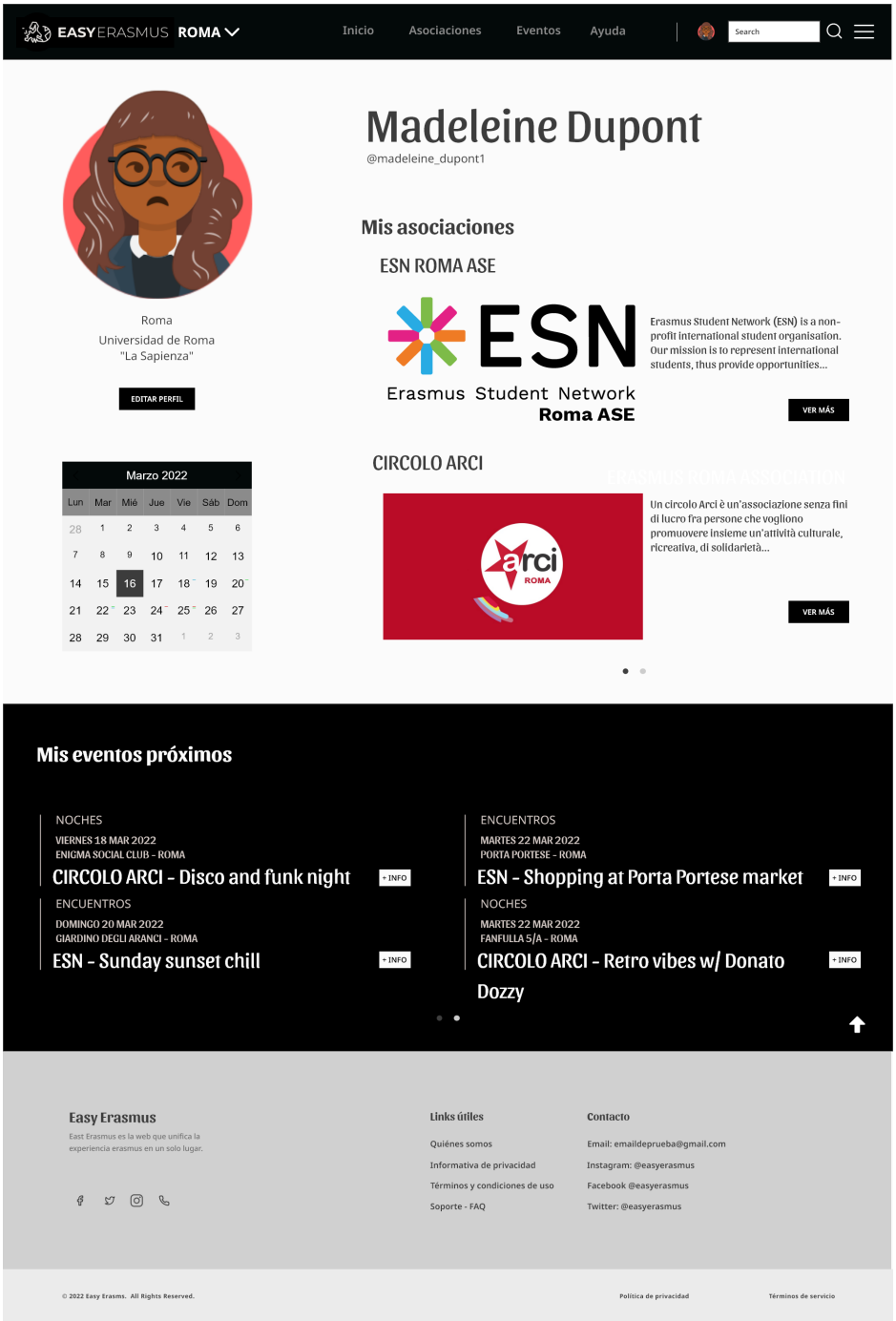
Las asociaciones por destino se desplegarán una tras otra, con una pequeña descripción y algunos de sus eventos próximos. Cada una de ellas cuenta con la redirección a sus propios perfiles (RF-13).

Interfaz HiFi 4: Perfil de asociación



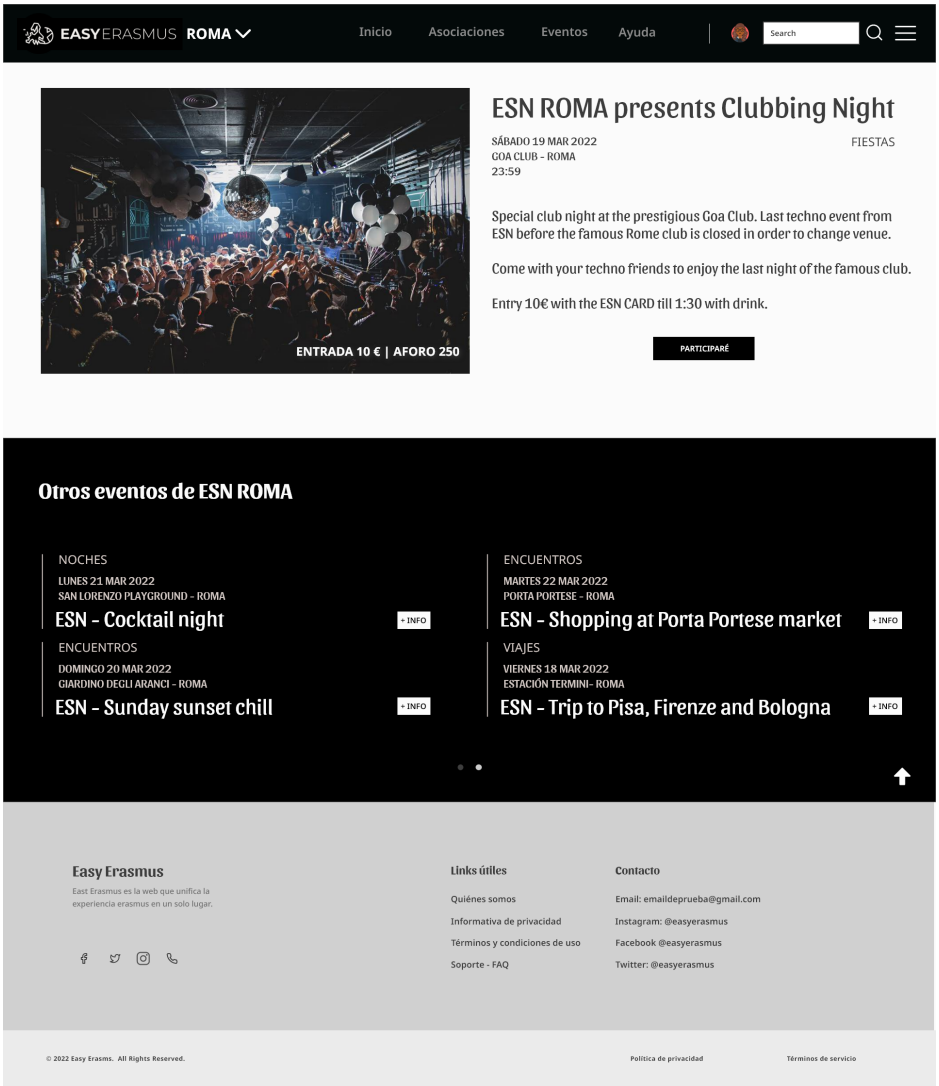
Cada asociación dispondrá de su detallado perfil (RF-13), que mostrará su información, sus eventos próximos (RF-17) y un calendario personalizado (RF-25). Los usuarios estudiantes tendrán la opción de inscribirse desde aquí (RF-07) ó eliminar su subscripción, si ya pertenecen a ella (RF-09).

Interfaz HiFi 5: Perfil de usuario



Los usuarios estudiantes tendrán un perfil que les ayude a ver tanto sus asociaciones como eventos en los que participan (RF-13, RF-17). Además de contar con la opción de cambiar sus detalles principales (RF-05, RF-06).

Interfaz HiFi 6: Información de evento



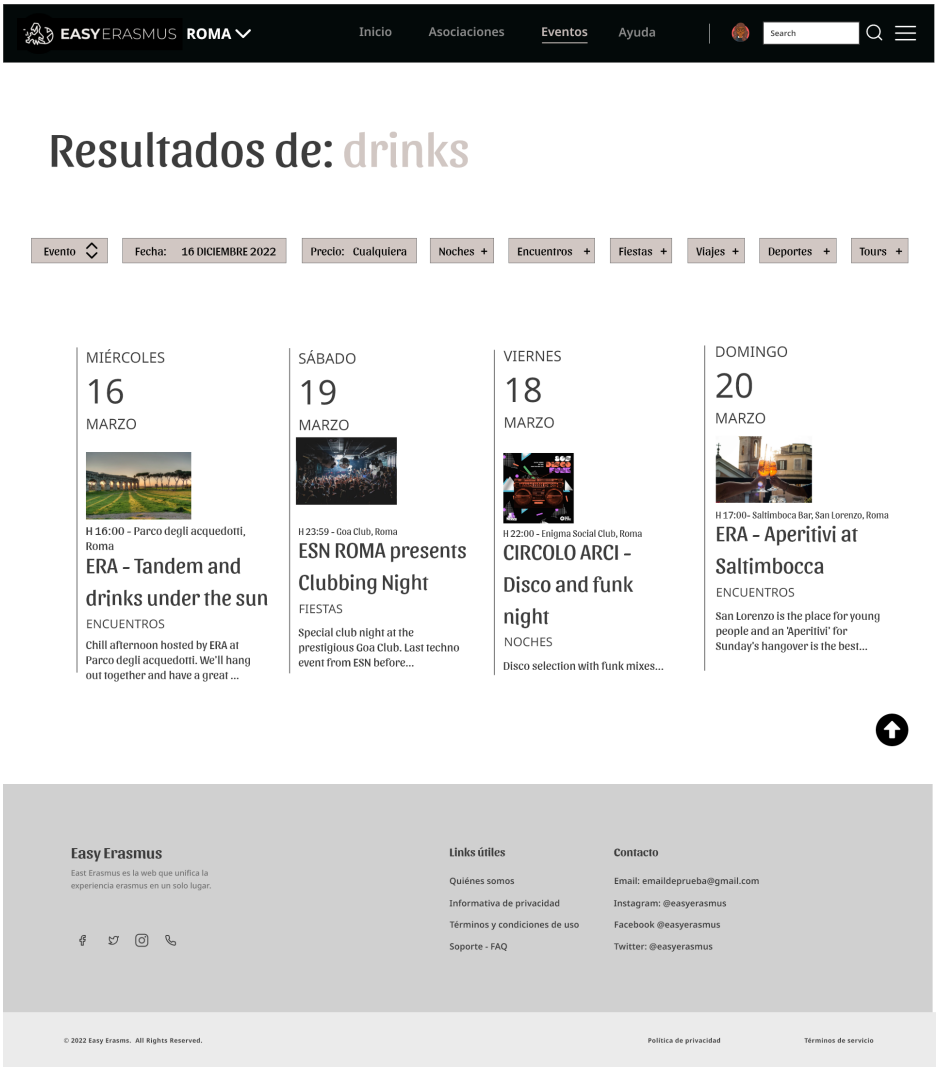
Cada evento publicado por una asociación desplegará una página propia de información y ofreciendo la posibilidad al usuario de participar (RF-16, RF-10, RF-11). También otros eventos de la misma asociación serán recomendados en esta misma página (RF-17).

Interfaz HiFi 7: Calendario de eventos por destino

Uno de los aspectos más llamativos de la aplicación será su calendario (RF-25), donde se

publicarán de manera atractiva lo que ocurre en la ciudad durante los próximos días. Además de contar con la aplicación de filtros para personalizar el despliegue según los intereses del usuario.

Interfaz HiFi 8: Búsqueda de eventos por destino



La búsqueda desde la barra de navegación redireccionará a la siguiente página. En ella la aplicación de filtros será posible para filtrar lo que se desea encontrar, y diferentes eventos o asociaciones que coincidan con la búsqueda serán mostrados RF-24.

4.3. Diseño arquitectónico del sistema

La organización del sistema web a desarrollar seguirá el patrón arquitectónico de **Modelo, Vista y Controlador**. Como es mencionado en la Unidad de Diseño Arquitectónico de la UPIICSA (Salas, Gómez & Mota, s.f.), *"el patrón MVC separa elementos de un sistema, permitiendo su modificación independiente entre sí."* Se fundamenta en la separación del código en tres capas con diferentes responsabilidades, los llamados (I) Modelos, (II) Vistas y (III) Controladores :

Modelos

En la siguiente capa se trabajará con los datos de la aplicación, contando con mecanismos para localizar la información y modificar su estado. La información se hará persistente con el uso de una base de datos y esta capa realizará las funciones para acceder a las tablas correspondientes.

Vistas

La capa con cara al público la proporcionan las vistas, conteniendo el código para la renderización de las interfaces de usuario en el navegador. Desde las vistas se trabajará con la información pero sin realizar un acceso directo a ella, sino mediante los modelos.

Controladores

Representan el enlace entre las vistas y los modelos, encargándose de la interacción del usuario. Esto es recogiendo las peticiones realizadas desde el navegador y requiriendo de la información desde los modelos para finalmente desplegar la vista adecuada. Consiste pues en el código para la interacción de ambas partes, conteniendo la mayoría de la lógica de la aplicación.

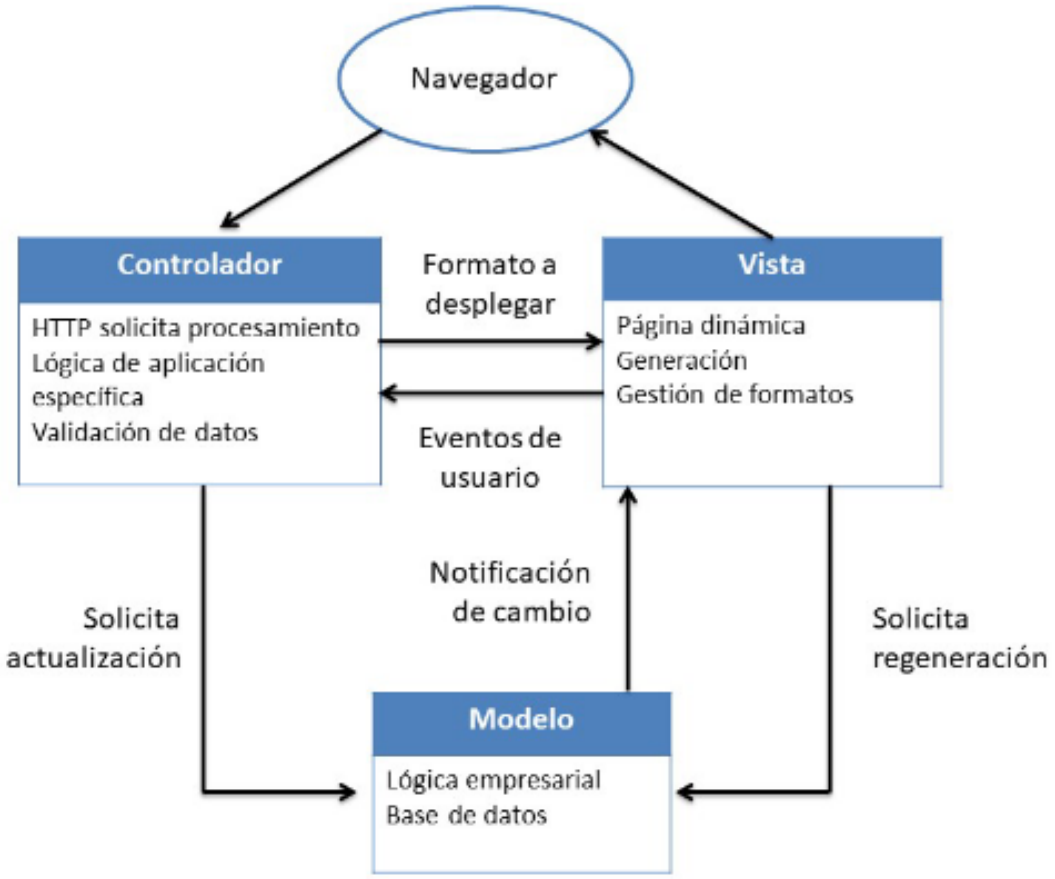


Figura 4.3: Arquitectura de aplicación web con el patrón MVC. *Extraído de Salas, Gómez & Mota (s.f.).*

Capítulo 5

Implementación

5.1. Tecnologías y herramientas a usar

En el proyecto se dará uso de las tecnologías y herramientas que están obteniendo los mejores resultados para desarrolladores y empresas en el ámbito web.
En la fase pronta de desarrollo usaremos herramientas que afecten al Front-End, y nos ayudarán a formatear y dar cara al sitio web. Seguidamente durante el desarrollo del Back-End usaremos tecnologías que definan el modelo y el control de la aplicación.
Para continuar, será necesario desplegar la aplicación online e importar lo desarrollado localmente.
Finalmente, el criterio responsive será comprometido por diferentes herramientas que pongan a prueba el sitio web implementado.

Desarrollo

Para el desarrollo de las labores del Front-End se usarán tecnologías como HTML5, CSS3, JavaScript, AJAX y Bootstrap.

- **HTML5** permite formatear el layout de la página, de manera que el navegador conozca el modo de mostrar la información y situar los elementos. (Desarrollo Web, 2001)

- **CSS3** definirá las reglas y estilos de representación para los elementos de la página. (B.,Gustavo, 2022)

- **Sass** es un extensión de CSS que reduce la repetición y ahorra tiempo. Permite funcionalidades no existentes previamente, como variables, reglas anidadas, imports, herencia, funciones, y mucho más. (Sass, s.f.)

- **JavaScript** dará el dinamismo a la web. Incorporando efectos, animaciones o acciones que ayudaran a la fluidez en la experiencia del usuario. (Fundamentos De JavaScript, 2022)

- **Ajax** permitirá que la aplicación funcione de forma asíncrona, con la finalidad de recuperar datos del servidor sin la necesidad de volver a cargar la página. (B.,Gustavo, 2022)

- **Bootstrap** es un entorno de trabajo que facilitará el diseño y desarrollo del sitio web, formado por plantillas de diseño con tipografía, como formularios, barras de navegación o menús prefabricados. (Rockcontent Blog, 2020)

Por otra parte, el desarrollo del Back-End se basará en tecnologías como PHP, Apache, MySQL y Laravel.

- **PHP** permitirá inyectar código HTML de manera sencilla, limpia y rápida. (Quality Devs, 2021)

- **Laravel** es un framework de PHP del tipo MVC que nos permitirá ahorrar tiempo de desarrollo. Cuenta con un motor de plantillas, llamado Blade. Además, Eloquent ORM nos facilitará la escritura de consultas. (Altube, 2021)

- **MySQL** será el sistema de gestión de bases de datos relacional (SQL) que usará la aplicación. Esto ayudará a mantener de forma ordenada los datos relacionados con las diferentes clases y elementos del sistema. Su uso será indirecto desde Laravel. (Robledano, 2019)

- **Apache** nos permitirá desplegar un servidor local en el que hacer pruebas en tiempo de desarrollo. Su uso será indirecto desde Laravel. (Ubuntu, s.f.)

Pruebas

Para evaluar lo implementado trataremos de poner a prueba el sistema mediante técnicas de testing software. Haciendo uso de **pruebas de caja blanca** (Ebooks Online, s.f.), escritas directamente desde el framework de Laravel, y **pruebas de caja negra** (La Oficina De Proyectos De Informática, 2017), desde el lado del cliente.

Finalmente, tras el despliegue, realizaremos un estudio de usabilidad, comprobando el sitio web en distintos navegadores web y utilizando herramientas que pongan a prueba el criterio responsive del mismo. Existen múltiples herramientas para ello como **Site-Shot** (Site-Shot, s.f.) que captura un sitio web desde distintos navegadores..

Despliegue

En la última fase, se ha de conseguir un hosting del sitio, que incluirá herramientas como cPanel y phpMyAdmin, las cuales nos ayudarán a gestionar la web y sus datos e importar lo desarrollado anteriormente.

- **cPanel** es un panel de control que nos permite administrar los servicios de un sitio en un solo lugar, ya sea monitorizando recursos, instalando software o gestionando usuarios FTP o direcciones de correo. (A., Deyimar, 2022)

- **phpMyAdmin** es una aplicación web que nos sirve para gestionar bases de datos MySQL de forma sencilla e intuitiva. Nos ayudará a subir el conjunto de archivos PHP que componen la aplicación. (García, 2022)

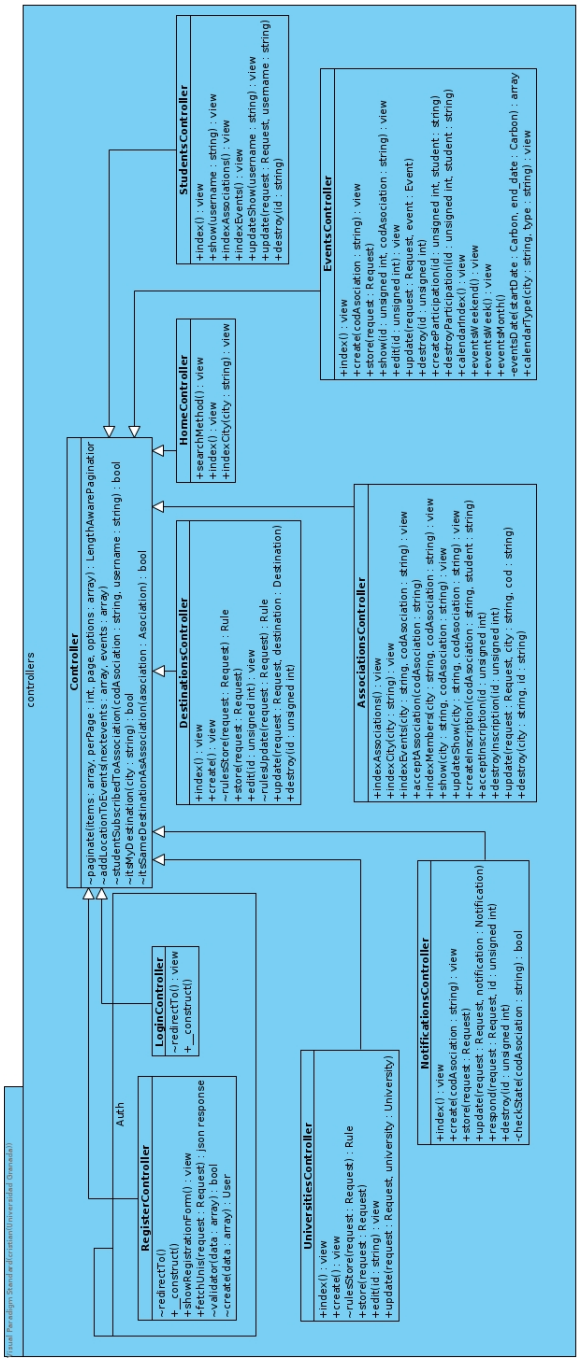
5.2. Diagrama de clases UML

En la presente sección vendrán expuestos los diagramas de clases de diseño, que hacen referencia a la arquitectura del sistema y recalcan el diseño arquitectónico de **Modelo, Vista y Controlador**.

5.2.1. Paquete Controllers

En este paquete de detallarán todos los controladores que gestionan las peticiones del sistema. Todos ellos heredan de la clase Controller de Laravel.

Diagrama de clases: Controladores del sistema



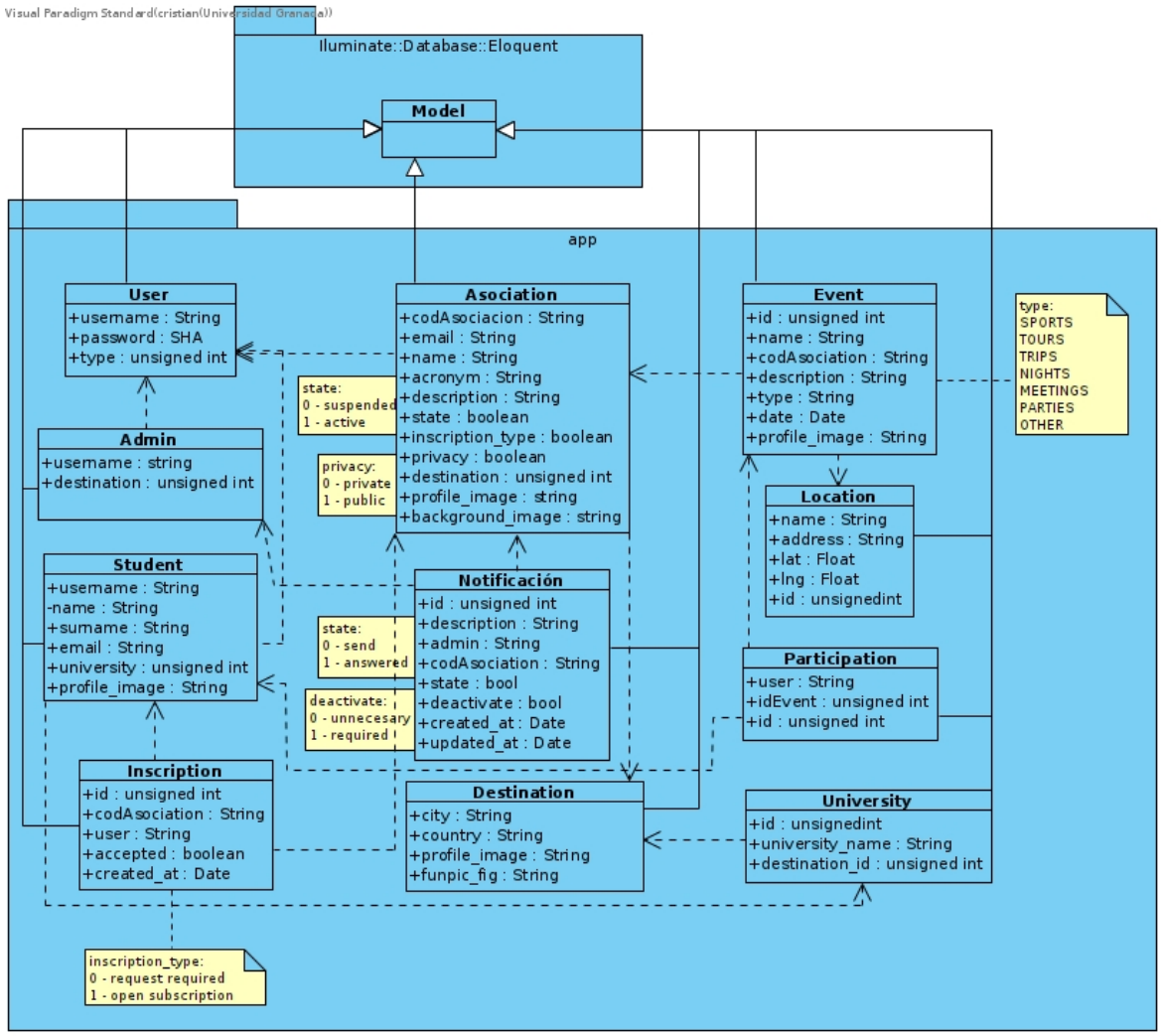
5.2.2. Paquete App

El siguiente paquete está formado por los modelos del sistema, subclases de Model de Laravel, que implementan el patrón Active Record usado por Eloquent para el acceso a la base de datos. Es por ello que cada una de las clases representa una tabla y cada instancia una fila de esta.

No deberían encontrarse relacionadas entre sí, pues solo garantizan el acceso a los datos de cada tabla. Las relaciones son implementadas por métodos en los controladores. Se han mantenido las relaciones de dependencia para entender las interacciones entre los elementos.

A su vez, todas las clases heredan de Model.

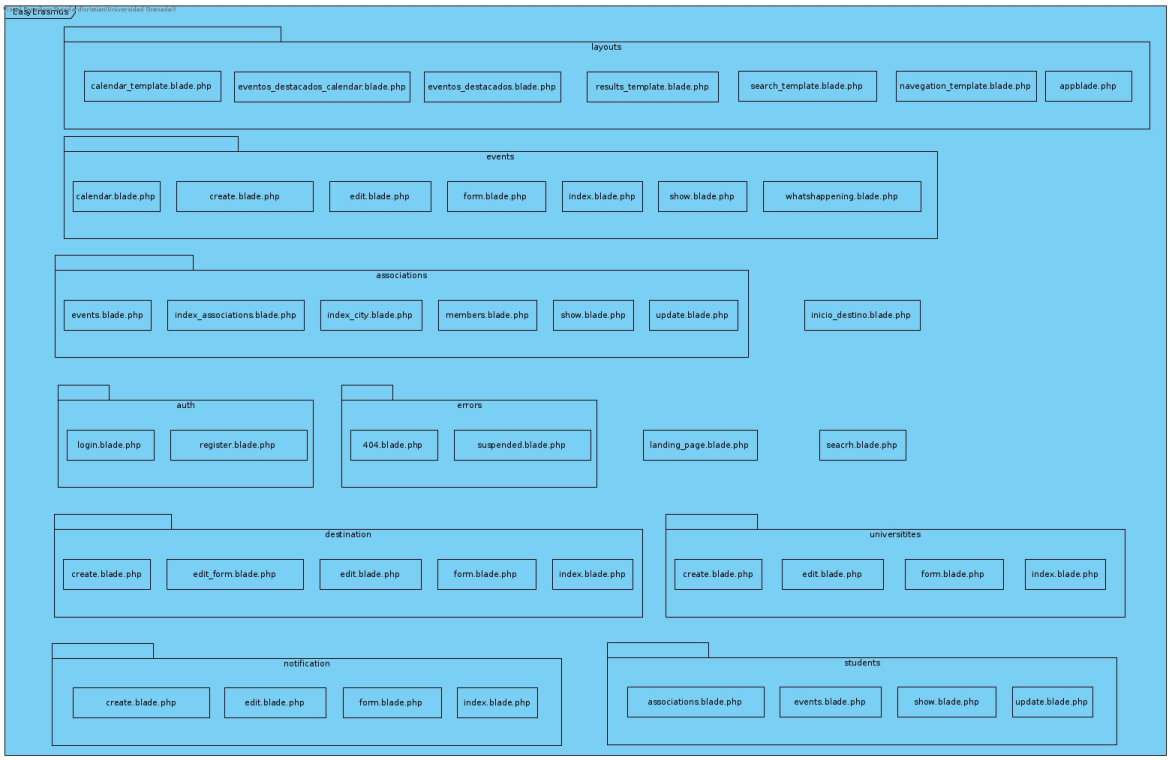
Diagrama de clases: Modelos del sistema



5.2.3. Paquete Views

A continuación, se presenta un diagrama UML aplicando la convención Web Application Extensions para mostrar cómo se comunican los diferentes controladores del servidor al generar las páginas (Orozco, 2008). El presente diagrama está formado por todas las vistas que representan el sistema, sin embargo sus interacciones no se han detallado pues sería insosteniblemente complejo. Se detallará más adelante la interacción de los controladores por separado.

Diagrama de WAE: Vistas del sistema

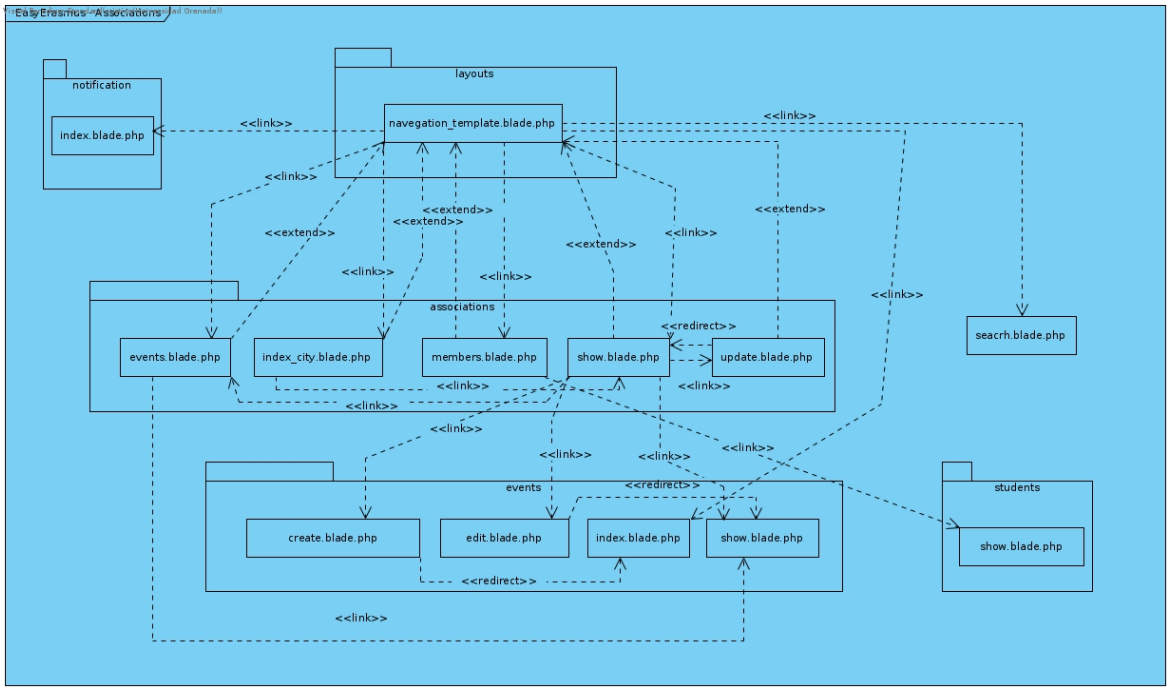


Lo que recibe el usuario como vista son archivos HTML (con su contenido audivisual y scripts asociados). Estos archivos son generados por **Blade** (Palacios, 2020). No son más que plantillas, llamadas por los controladores al finalizar sus procesamientos.

Es por ello que la mayoría de las vistas se encuentran empaquetadas por controladores. Además, el paquete superior *layouts* está formado por plantillas, que serán extendidas por los demás archivos.

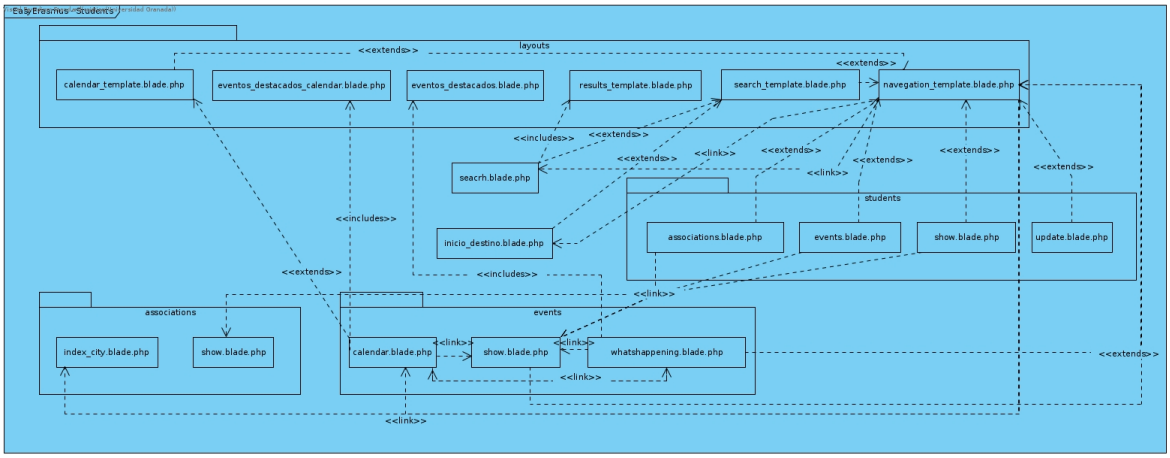
Para continuar y observar la interacción entre ellas, se ha realizado un diagrama reducido, que representa las vistas asociadas con el controlador de Asociaciones:

Diagrama de WAE: Vistas del Controlador de Asociaciones



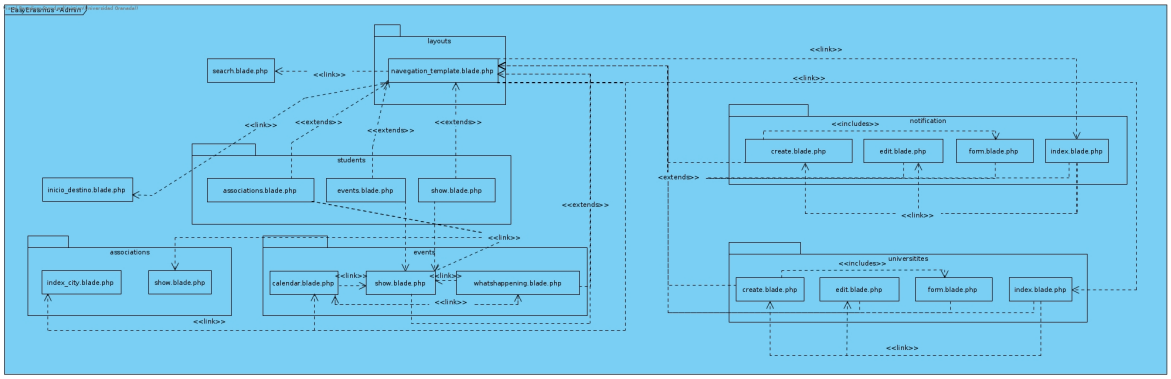
Todas las vistas de asociaciones extienden la plantilla de navegación. A su vez, desde sus varias páginas se redireccionan ó enlazan páginas de otros controladores como notificaciones, eventos o estudiantes.

Diagrama de WAE: Vistas del Controlador de Estudiantes



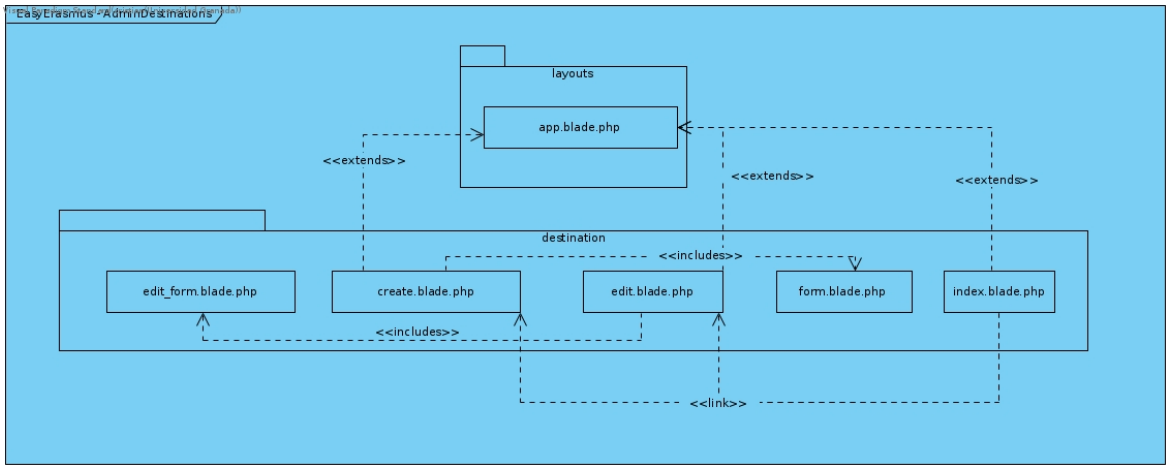
Los usuarios estudiantes pueden acceder también al índice del destino, junto al calendario interactivo, los eventos por periodo de tiempo (día, semana, fin de semana y mes), a su información de perfil y al método de búsqueda.

Diagrama de WAE: Vistas del Administrador por destino



La administración de un destino concreto además tiene acceso al CRUD de notificaciones y universidades. Sin olvidar todo la vista de usuario normal de la aplicación.

Diagrama de WAE: Vistas del Administrador de destinos



Por último, el usuario encargado de añadir destinos al sistema sólo dispondrá del CRUD de estos y sus administradores asignados, sin ningún otro acceso a la aplicación.

5.3. Descripción de clases y módulos

La tecnología que usaremos y que nos ayudará a implementar la arquitectura MVC es **Laravel** (Ighodaro, 2018). Este framework nos garantizará la separación acordada, en primer lugar con *modelos*, clases con propiedades que se corresponden con las tablas de la base de datos. Seguidamente, los *controladores* manejan las acciones de los usuarios y la lógica de la aplicación, además de garantizar CRUD sencillos para cada modelo. Por último, las *vistas* trabajan con el motor de plantillas Blade, el cual nos proporcionará inyección de sintáxis PHP y reutilización de código HTML de una manera amigable.

5.3.1. Models

Cada uno de los modelos de la aplicación representa una tabla en la base de datos y cada instancia una fila de ella. Se ha creado uno para cada tabla especificada en el modelo:

Destination: Simboliza a una ciudad de destino Erasmus, almacena información de la ciudad, su país y la ruta de almacenamiento de sus imágenes asociadas (imagen representativa y GIF para la página principal del destino).

University: Representa a una universidad en un destino concreto. Sus propiedades no son más que el nombre de la universidad y el destino asociado.

User: Representa los credenciales de un usuario del sistema (usuario y contraseña), además de su rol en este: *Administrador de destinos*, *Asociación*, *Estudiante Erasmus* ó, *Administrador de un destino Erasmus*.

Admin: Simboliza al administrador para un destino concreto, guardando exclusivamente su usuario y el destino que gestiona.

Association: Representa a una asociación Erasmus. Reúne información básica de esta (código de asociación, nombre, descripción, acrónimo, email), el destino al que está asociada, opciones de inscripción (abierta ó por solicitud), visibilidad (pública ó privada) y el estado actual (activa ó inactiva). Además se almacenan rutas de imágenes representativas (de perfil y de fondo).

Student: Simboliza el usuario más común de la aplicación, el estudiante. Sus propiedades son de información básica (usuario, nombre, apellidos y email), de su universidad asociada y la ruta de la imagen de perfil.

Location: La localización exacta de cada evento será representada por este modelo. Su información representa la dirección del lugar, su nombre y una latitud y longitud específicas.

Event: Representa un evento Erasmus creado por una asociación. Los datos que lo definen son una identificación única, información representativa (nombre, descripción, tipo, participantes, precio), la asociación a la que pertenece, la localización en el que toma lugar y por último su fecha.

Inscription: Las inscripciones simbolizan la relación entre estudiantes y asociaciones. La información de cada una hará referencia al usuario estudiante, a la asociación y tendrá un estado (aceptada o pendiente), no siendo efectiva hasta ser aceptada.

Participation: Una participación representa la asistencia de un estudiante a un evento. Sus propiedades son las necesarias para suplir la relación entre usuario y evento, además de un id de identificación.

Notification: Simboliza el anuncio enviado por un administrador de un destino concreto hacia una asociación, con la finalidad de hacerle saber que algún cambio le es requerido. Los datos requeridos para su representación son una breve descripción, el estado de la notificación (enviada ó respuesta), una opción de desactivación (si es o no requerida una desactivación de la asociación afectada) y por último las fechas de creación y actualización de la notificación.

El ejemplo siguiente muestra el modelo de notificaciones. Este define las reglas de validación de datos en solicitudes POST o PATCH, los atributos asignables por el usuario (fillable) y dos métodos que devuelven la asociación o administrador asignados a la notificación, mediante el acceso a la base de datos y obteniendo el registro indicado en cada una de sus tablas.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

/**
 * Class Notification
 *
 * @property $id
 * @property $admin
 * @property $codAsociation
 * @property $description
 * @property $state
 * @property $deactivate
 */
class Notification extends Model
{
    static $rules = [
                'admin' => 'required',
                'codAsociation' => 'required',
                'description' => 'required',
                'state' => 'required',
                'deactivate' => 'required',
    ];

    /**
     * Attributes that should be mass-assignable.
     *
     * @var array
     */
    protected $fillable = ['admin','codAsociation','description','
       state','deactivate'];

    /**
     * @return \Illuminate\Database\Eloquent\Relations\HasOne
     */
    public function admin()
    {
        return $this->hasOne('App\Models\Admin', 'username', '
           admin');
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\HasOne
```

```
44          */
45      public function asociation()
46      {
47          return $this->hasOne('App\Models\Asociation', '
                codAsociation', 'codAsociation');
48      }
49  }
```

5.3.2. Controllers

Los controladores manejan la mayoría de la lógica de la aplicación. Pues se ocupan de recibir los datos del usuario, procesarlos según sus privilegios y cargar la vista correspondiente.

Controller: Es el controlador base del que todos los demás heredan, en él se han escrito funciones y comprobaciones que puedan ser utilizadas por los demás.

Auth\LoginController: Este controlador maneja la autenticación de usuarios de la aplicación. Si se realiza satisfactoriamente redirigirá a los usuarios a la pantalla de inicio del destino.

Auth\RegisterController: Tanto el registro de nuevos estudiantes y asociaciones, como la validación y creación de sus respectivos usuarios, serán llevados a cabo por este controlador. El formulario de registro a su vez recibirá las universidades disponibles dinámicamente según el destino seleccionado.

HomeController: El controlador de inicio maneja la vista de inicio según el tipo de usuario que haya iniciado sesión. En caso de ser estudiante ó administrador de destino, será redirigido a la página inicial de eventos próximos. Las asociaciones son directamente redirigidas a su perfil. El administrador general sólo y exclusivamente tendrá acceso a la gestión de nuevos destinos. El controlador además implementará el método de búsqueda según eventos, asociaciones y localizaciones.

DestinationsController: Es el controlador asociado al registro de destinos por parte del administrador. Entre sus métodos se encuentra la creación de destinos (y sus administradores), su modificación y eliminado, y el indexado de estos.

UniversitiesController: Es el controlador asociado al registro de universidades por parte del administrador de un destino. Entre sus métodos se encuentra la creación de universidades, su modificación y el indexado de universidades.

StudentsController: El rol de estudiantes estará prácticamente compuesto por este controlador, pues despliega las vistas de su perfil, modificación, eliminación, asociaciones a las que pertenece y eventos a los que participa.

AssociationsController: Por otra parte, las funcionalidades de las asociaciones vienen definidas en este controlador. Mostrado, modificación y eliminación de asociaciones, índice de asociaciones por destino para todos los usuarios, índice de eventos, índice de miembros de la asociación y gestión de inscripciones a la asociación (creación, aceptación y borrado). A su vez, un método que muestra el índice de todas las asociaciones en el destino, creado para ser desplegado solo a administradores.

EventsController: Este controlador implementará el CRUD completo de eventos, con sus funciones de modificación solo accesibles para asociaciones. Además, la participación de eventos a elección de usuario, verificando que pertenezca a la asociación previamente. Asimismo, el calendario interactivo de eventos según fecha, precio, participantes o tipo de evento.

NotificationsController: En último lugar, la gestión de notificaciones se realiza desde este controlador. Se permitirá el CRUD completo de notificaciones por parte de los administradores y el visionado y respuesta por parte de las asociaciones.

A continuación se muestra el ejemplo de un método de controlador. El presente es llamado cuando se quiere actualizar la información de una universidad asociada a un destino. Al igual que todos gran parte de los métodos de controlador, recibe datos para su procesamiento. En este caso recibe una solicitud con los datos a modificar y el código de identificación de la universidad. Seguidamente valida los datos, actualiza el registro y devuelve una vista (Redirección a la ruta del índice de universidades, que a su vez llamará a un método de controlador que devolverá la correspondiente vista).

```
/**
 * Update the specified university in storage.
 *
 * @param  \Illuminate\Http\Request $request
 * @param  University $university
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, University $university)
{

    request()->validate($this->rulesStore($request));


    $university->update([
        'id' => strtoupper($request->input('id')),
        'university_name' => $request->input('university_name'),
    ]);


    return redirect()->route('universities.index')
        ->with('success', 'University updated successfully');
}
```

En el siguiente ejemplo se puede apreciar el retorno de una vista. En este se despliegan

todas las asociaciones de un destino (paginadas), junto a un par de eventos próximos. Finalmente se devuelve una vista y se adjuntan los datos de las asociaciones y sus eventos para ser revelados desde Blade.

```
/**
 * Returns user view of all associations of a destination
 *
 * @param String $city
 * @return mixed
 */
public function indexCity($city){

    $destination = Destination::where('city',$city)->first();

    if(empty($destination) || !$this->itsMyDestination($city))
        return view('errors.404');

    $asociations=Asociation::where("destination",$destination->id)
       ->paginate(4);

    $asociations_events=array();

    foreach ($asociations as $asociation){
        $events=Event::where('codAsociation',$asociation->
           codAsociation)->where("date",'>=', Carbon::now()->
           startOfDay())->get();
        $nextevents=array();
        $this->addLocationToEvents($nextevents,$events);

    usort($nextevents, array('App\Models\Event','dateComparator'))
       ;

    $nextevents=array_slice($nextevents, 0, 2, true);

    $asociations_events[$asociation->codAsociation]=$nextevents;
    }

    return view('asociations.index_city',['asociations' =>
       $asociations, 'asociations_events' => $asociations_events])
       ;
}
```

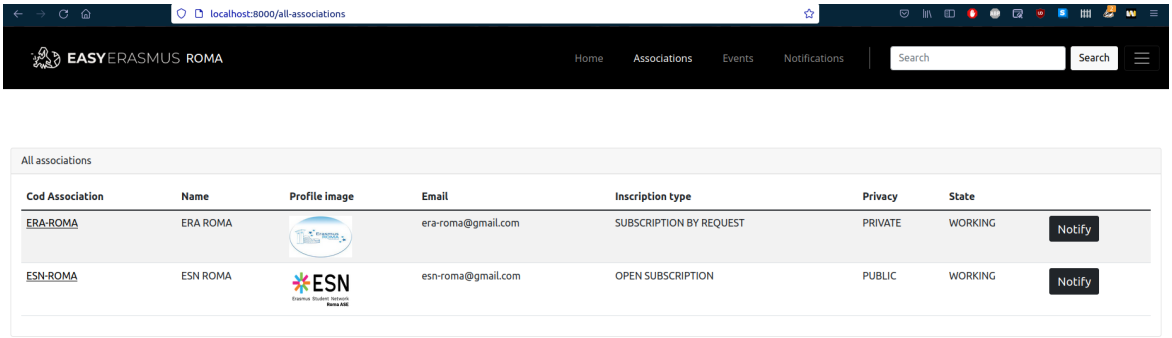
5.3.3. Views

Sucesivamente, las vistas son generadas gracias a plantillas Blade: código HTML al que se le ha añadido marcas para ejecutar código PHP y añadirlo a la vista generada. Este viene introducido dentro de las marcas {{}} o después del signo @.
En el próximo ejemplo se puede observar parte del código de la vista del índice de asociaciones (administrador). Esta extiende parte de su código de la plantilla de navegación (barra de navegación y footer). Además, recibe los datos en la variable *$asociations*, que serán

detalladas mediante un bucle *@foreach*. Finalmente, el método *links()* se encarga de paginar la información en caso de que haya numerosos registros.

```
@extends('layouts.navegation-template')
@section('titlepage',"All Associations - ")
@section('body')
.
.
.
<div class="card-body">
    @if(count($asociations)==0)
    <h1 style="text-align: center; margin: 5% auto 5% auto;">There
        are no associations...</h1>
    @else
    <div class="table-responsive">
    <table class="table table-striped table-hover">
    .
    .
    .
    <tbody class="tabla-event-index">
        @foreach ($asociations as $asociation)
        <tr>
        <td>{{ strtoupper($asociation->codAsociation) }}</td>
                <td>{{ $asociation->name }}</td>
        <td><img src="{{ $asociation->profile_image }}" alt="{{
           $asociation->name }}"></td>
                <td>{{ $asociation->email }}</td>
                <td>@if($asociation->inscription_type) OPEN
                   SUBSCRIPTION @else SUBSCRIPTION BY REQUEST
                   @endif</td>
                <td>@if($asociation->privacy)PUBLIC @else PRIVATE
                   @endif</td>
                <td>@if($asociation->state)WORKING @else SUSPENDED
                    @endif</td>
        <td><a class="btn btn-dark "href="{{ route('notifications.
           create',['codAsociation'=>$asociation->codAsociation])
           }}">Notify</a>
             .
             .
             .
        </td> </tr>
        @endforeach
    </tbody> </table> </div>
    @endif
</div>
{{ $asociations->links() }}
.
.
.
@endsection
```

Figura 5.3.3: Vista de administración del índice de asociaciones



5.3.4. Roles y funciones

Administrador de destinos: Es aquel tipo de usuario que se encargará de gestionar los destinos en los que se encuentra disponible la aplicación. Sus funcionalidades son:

- Creación, modificación y borrado de destinos Erasmus.
- Asociación de un administrador a cada destino.
- Creación de la universidad más relevante del destino dado de alta.

Asociación Erasmus: Los usuarios encargados de gestionar asociaciones son unos de los grandes actores de la aplicación. Entre sus competencias se encuentran:

- Creación, vista, modificación y borrado de su asociación Erasmus.
- Creación, vista, modificación y borrado de eventos de su asociación (la creación no será posible si se encuentra inactiva).
- Índice de eventos de la asociación (vista de asociación y de estudiante).
- Índice de miembros de la asociación.
- Aceptación o eliminación de miembros de la asociación.
- Acceso a perfil de usuarios miembros de la asociación.
- Vista de otras asociaciones o eventos en destino (vista de estudiante).
- Índice de notificaciones recibidas.
- Resolución de notificaciones recibidas.
- Uso del método de búsqueda de la aplicación (vista de estudiante).

Estudiante Erasmus: Los estudiantes Erasmus son el principal usuario de la aplicación, contando con las vistas más pulidas y atractivas para la exhibición de eventos y asociaciones en su destino Erasmus. Sus posibilidades en la aplicación son:

- Creación, vista, modificación y borrado de su usuario Erasmus.
- Vista inicial personalizada con los eventos próximos en destino.
- Vista del perfil de una asociación.
- Vista del evento de una asociación.
- Calendario interactivo de eventos en destino.
- Posibilidad de inscripción en asociaciones en destino.
- Posibilidad de participación en eventos de asociaciones a las que se pertenezca.
- Vista de asociaciones a las que se pertenece.
- Vista de los eventos a los que se participa.
- Uso del método de búsqueda de la aplicación.

Administrador de un destino Erasmus: El administrador asociado a un destino Erasmus contará con varias funcionalidades para garantizar la correcta marcha de la aplicación en una ciudad. Sus ocupaciones son las siguientes:

- Creación, modificación y borrado de universidades en destino.
- Aceptación de nuevas asociaciones en destino.
- Creación y modificación de notificaciones a asociaciones en destino (que requieran de su desactivación o no).
- Índice de notificaciones recibidas.
- Índice de todas las asociaciones por destino.
- Vista inicial con los eventos próximos en destino (sin importar la privacidad).
- Vista de asociaciones y sus eventos en destino (sin importar la privacidad).
- Calendario interactivo de eventos en destino (sin importar la privacidad).
- Uso del método de búsqueda de la aplicación (sin importar la privacidad).

Capítulo 6

Pruebas

En la penúltima etapa del proyecto trataremos de poner a prueba el sistema haciendo uso de técnicas de testing de software, de una manera paralela al desarrollo, para evitar problemas en la funcionalidad de manera previa a su lanzamiento. Mediante las pruebas de caja blanca, en tiempo de producción, probaremos la estructura interna, el diseño y la codificación del software para verificar el flujo de entrada y salida de datos. Posteriormente, cuando el sistema esté operativo, llegará el turno de las pruebas de caja negra, en las cuales la funcionalidad se verifica sin tomar en cuenta la estructura interna del código.

6.1. Tests de caja blanca

Las pruebas de caja blanca se pueden realizar a nivel de sistema o funcionalidad. Verifican el flujo de trabajo de la aplicación (Ebooks Online, s.f.), probando un conjunto de entradas predefinidas con salidas esperadas o requeridas, de modo que no resulte en error. Estas pruebas nos otorgarán varios beneficios en cuanto a robustez y encontrar:

- Agujeros de seguridad internos
- Rutas rotas o mal estructuradas
- Flujo correcto de las entradas a través del código
- Rendimiento esperado
- Pruebas de cada enunciado, objeto o función individual

Las pruebas las realizaremos con el mismo framework **Laravel** (Laravel Docs, s.f.), pues incluye herramientas de testing con PHPUnit. Escribiremos entonces tests funcionales para cada controlador de la aplicación, para poner a prueba directamente sus métodos (Echazarreta, 2020).

6.1.1. Controlador de Login

Prueba 1: Mostrar vista de login al acceder a su URL

Descripción del caso: Cualquier usuario sin iniciar sesión debe poder acceder al formulario de login si escribe su URL.

```
/** @test */
/* Getting login links displays the login form*/
public function login_displays_the_login_form()
{
    $response = $this->get('/login');

    $response->assertStatus(200);

    $response->assertViewIs('auth.login');
}
```

Resultado: Satisfactorio.

Prueba 2: Enviar solicitud de login sin rellenar los campos

Descripción del caso: Si no rellenamos los campos, la solicitud POST no se mandará.

```
/** @test */
/* Trying to post nothing at login form, it returns errors in both
    fields*/
public function login_displays_validation_errors()
{
    $response = $this->post('/login',[]);

    $response->assertStatus(302);

    $response->assertSessionHasErrors('username');
    $response->assertSessionHasErrors('password');
}
```

Resultado: Satisfactorio.

Prueba 3: Intento de inicio de sesión con usuario no existente

Descripción del caso: Si los credenciales son incorrectos, el formulario mostrará el correspondiente error.

```
/** @test */
/* Trying to post wrong username, it returns an error in this
   field */
public function login_displays_error_with_wrong_username()
{
    $response = $this->post('/login',[
        'username' => 'noexisto',
    ]);

    $response->assertStatus(302);
}
```

Resultado: Satisfactorio.

Prueba 4: Enviar solicitud correcta de login

Descripción del caso: Si el login es satisfactorio, la redirección se hará a la ruta inicial del destino. Actuando como usuario estudiante loggeado, se podrá acceder sus rutas (p. ej.: calendario de eventos).

```
/** @test */
/* Posting valid information in every field of the login form,
   redirects to the route of the city */
public function login_authenticates_and_redirects_user()
{

    $student = \App\Models\Student::factory(Student::class)->
       create();

    $user = User::find($student->username);

    $university = University::find($student->university);

    $destination = Destination::find($university->destination_id);

    $response = $this->post('/login',[
        'username' => $student->username,
        'password' => 'password'
    ]);

    $response->assertRedirect(route('inicio-ciudad',$destination->
       city));
    $this->assertAuthenticatedAs($user);

    $response = $this->actingAs($user)->get(route('events-calendar
       ',$destination->city));

    $response->assertStatus(200);

}
```

Resultado: Satisfactorio.

Figura 6.1.1: Resultado de ejecución de pruebas del controlador de login.

```
[CristianFernandez:~/Escritorio/UNI/TFG/easyerasmus]$ php artisan test tests/Feature/Http/Controllers/Auth/LoginControllerTest.php

  PASS  Tests\Feature\Http\Controllers\Auth\LoginControllerTest
  ✓ login displays the login form
  ✓ login displays validation errors
  ✓ login displays error with wrong username
  ✓ login authenticates and redirects user

  Tests:  4 passed
  Time:   1.29s
```

6.1.2. Controlador de Registro

Prueba 5: Registro correcto de usuario estudiante

Descripción del caso: Registrar correctamente a un estudiante finaliza con la redirección a la ruta del destino seleccionado, a su vez como estudiante, ya podrá realizar sus funciones (p.e.: actualizar su perfil).

```
/** @test */
/* Posting valid information in every field of the register form
   for a student, redirects to the route of the city */
public function register_creates_and_authenticates_a_student(){


    $university = \App\Models\University::factory(University::
       class)->create();

    $destination = Destination::find($university->destination_id);

    $response = $this->post(route('register'), [
        'name' => 'ejemplo',
        'surname' => 'ejemplo',
        'username' => 'ejemplo12',
        'type' => 1,
        'destination_id' => $university->destination_id,
        'university' => $university->id,
        'email' => 'test@example.com',
        'password' => '12345678',
        'password_confirmation' => '12345678',
        'profile_image' => 'imagen'
    ]);

    $this->assertDatabaseHas('users', [
        'username' => 'ejemplo12',
    ]);

    $this->assertDatabaseHas('students', [
        'username' => 'ejemplo12',
    ]);

    $response->assertRedirect(route('inicio-ciudad',$destination->
       city));

    $user = User::find('ejemplo12');

    $response = $this->actingAs($user)->get(route('students-update
       ',$user->username));

    $response->assertStatus(200);
}
```

Resultado: Satisfactorio.

Prueba 6: Registro correcto de usuario estudiante

Descripción del caso: Registrar correctamente a una asociación finaliza con la redirección a la ruta del su perfil, a su vez como asociación, ya podrá acceder a sus funciones (p.e.: indexar sus eventos).

```
 /* Posting valid information in every field of the register form
    for an association, redirects to the route of the city */

/** @test */
public function register_creates_and_authenticates_an_association
    (){


    $university = \App\Models\University::factory(University::
       class)->create();

    $destination = Destination::find($university->destination_id);

    $response = $this->post(route('register'), [
        'name' => 'ejemplo',
        'acronym' => 'eje',
        'username' => 'eje-'.$destination->city,
        'type' => 2,
        'description' => 'Lorem ipsum dolor sit
           ajhguvbnmnbuivybunibivyctuuvybunbvyctvybunmet
           consectetur adipisicing elit. Dicta tenetur veniam
           laboriosam dolor deleniti, mollitia ducimus doloribus
           repellendus architecto exercitationem ipsam! Fuga illo
           sunt eveniet voluptates. Vero nulla error sit
           consequuntur, exercitationem fuga necessitatibus, eos
           laborum quae explicabo, nobis ipsum autem quaerat
           recusandae non! Harum commodi voluptates ex a
           perferendis?',
        'destination_id' => $university->destination_id,
        'email' => 'testeje@example.com',
        'password' => '12345678',
        'password_confirmation' => '12345678',
        'profile_image' => 'imagen',
    ]);

    $response->assertRedirect(route('asociations-show',['city'=>
       $destination->city,'codAsociation' => 'eje-'.$destination->
       city]));


    $this->assertDatabaseHas('users', [
        'username' => 'eje-'.$destination->city,
    ]);

    $this->assertDatabaseHas('asociations', [
```

```
32            'codAsociation' => 'eje-'.$destination->city,
33        ]);
34
35        $user = User::find('eje-'.$destination->city);
36
37        $response = $this->actingAs($user)->get(route('events.index'))
              ;
38
39        $response->assertStatus(200);
40
41    }
```

Resultado: Satisfactorio.

Figura 6.1.2: Resultado de ejecución de pruebas del controlador de registro.

```
[CristianFernandez:~/Escritorio/UNI/TFG/easyerasmus]$ php artisan test tests/Feature/Http/Controllers/Auth/RegisterControllerTest.php

  PASS  Tests\Feature\Http\Controllers\Auth\RegisterControllerTest
  ✓ register creates and authenticates a student
  ✓ register creates and authenticates an association

  Tests:  2 passed
  Time:   1.41s
```

6.1.3. Controlador de Administradores de destinos

Prueba 7: Registro correcto de destino Erasmus

Descripción del caso: Un usuario administrador de destinos podrá acceder a la vista de creación de destinos y al registrar correctamente a un destino finaliza con la redirección a la ruta del índice de ciudades, a su vez se comprueba que el destino en cuestión y su administrador han sido almacenados correctamente.

```
1   /* Posting valid information in every field of the form for a
        destination, redirects to the index of cities */
2   public function test_creation_of_destination(){
3
4       $user = \App\Models\User::factory(User::class)->make([
5           'username' => 'main-admin',
6           'type' => 4,
7           'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC
                /.og/at2.uheWG/igi', // password
8           'remember_token' => Str::random(10),
9       ]);
10
11      $user->save();
12
13      $response = $this->actingAs($user)->get(route('destinations.
            create'));
14
15      $response->assertViewIs('destination.create');
16
17      $response = $this->actingAs($user)->post(route('destinations.
            store'), [
```

```
18            'city' => 'bologna',
19            'country' => 'italia',
20            'profile_image' => 'imagen',
21            'funpic_gif' => 'imagen',
22            'username' => 'admin-bologna',
23            'password' => 'password',
24            'password_confirmation' => 'password',
25            'id' => 'IBOLOGNA1',
26            'university' => 'Universita di Bologna'
27        ]);
28
29        $response->assertRedirect(route('destinations.index'));
30
31        $this->assertDatabaseHas('admins', [
32            'username' => 'admin-bologna',
33        ]);
34
35        $this->assertDatabaseHas('destinations', [
36            'city' => 'bologna',
37            'country' => 'italia',
38        ]);
39
40    }
```

Resultado: Satisfactorio.

Prueba 8: Actualización correcta de destino Erasmus

Descripción del caso: El administrador de destinos podrá acceder al formulario de modificación de cada destino y actualizar cualquier campo, esto conlleva la redirección a la ruta del índice de ciudades y la modificación correspondiente en la base de datos.

```
1   /* Posting valid information in any field of the update form for a
        destination, redirects to the index of cities */
2   public function test_update_of_destination(){
3
4       $user = \App\Models\User::factory(User::class)->make([
5           'username' => 'main-admin',
6           'type' => 4,
7           'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC
               /.og/at2.uheWG/igi', // password
8           'remember_token' => Str::random(10),
9       ]);
10
11      $user->save();
12
13      $destination = \App\Models\Destination::factory(Destination::
           class)->create();
14
15      $response = $this->actingAs($user)->get(route('destinations.
           edit',$destination->id));
```

```
16
17      $response->assertViewIs('destination.edit');
18
19      $response = $this->actingAs($user)->put(route('destinations.
           update',$destination->id), [
20          'city' => 'bolognina',
21          'country' => 'italy',
22      ]);
23
24      $response->assertRedirect(route('destinations.index'));
25
26      $this->assertDatabaseHas('destinations', [
27          'city' => 'bolognina',
28          'country' => 'italy',
29      ]);
30  }
```

Resultado: Insatisfactorio.

Mejoras conseguidas: El input de imágenes era obligatorio anteriormente, por lo que la comprobación fallaba. Ha sido corregido para evitarlo y permitir que cualquier campo sea actualizado separadamente.

Figura 6.1.3: Resultado de ejecución de pruebas del controlador de administradores de destinos.

```
[CristianFernandez:~/Escritorio/UNI/TFG/easyerasmus]$ php artisan test tests/Feature/AdminDestinationTest.php

  PASS  Tests\Feature\AdminDestinationTest
  ✓ creation of destination
  ✓ update of destination

  Tests:  2 passed
  Time:   1.29s
```

6.1.4. Controlador de Asociaciones Erasmus

Prueba 9: Acceso a todas las rutas disponibles y específicas para asociaciones

Descripción del caso: La asociación con la sesión iniciada podrá acceder a todas sus funcionalidades: perfil de asociación, índice de sus eventos, modificación de sus datos, índice de sus miembros, lista de asociaciones en destino, creación de eventos e índice de notificaciones.

```
1  /* Trying to access every route available for the association
      should be satisfactory */
2  public function test_get_all_routes_available_for_associations()
3  {
4      $asociation = \App\Models\Asociation::factory(Asociation::
          class)->create();
5
6      $destination = Destination::find($asociation->destination);
7
```

```
8       $user= User::find($asociation->codAsociation);
9
10      $response = $this->actingAs($user)->get(route('asociations-
           show',[$destination->city,$asociation->codAsociation]));
11
12      $response->assertViewIs('asociations.show');
13
14      $response = $this->actingAs($user)->get(route('asociations-
           index-events',[$destination->city,$asociation->
           codAsociation]));
15
16      $response->assertViewIs('events.index');
17
18      $response = $this->actingAs($user)->get(route('asociations-
           update',[$destination->city,$asociation->codAsociation]));
19
20      $response->assertViewIs('asociations.update');
21
22      $response = $this->actingAs($user)->get(route('asociations-all
           -members',[$destination->city,$asociation->codAsociation]))
           ;
23
24      $response->assertViewIs('asociations.members');
25
26      $response = $this->actingAs($user)->get(route('asociations-
           index_city',$destination->city));
27
28      $response->assertViewIs('asociations.index_city');
29
30      $response = $this->actingAs($user)->get(route('events.create',
           $asociation->codAsociation));
31
32      $response->assertViewIs('events.create');
33
34      $response = $this->actingAs($user)->get(route('notifications.
           index'));
35
36      $response->assertViewIs('notification.index');
37  }
```

Resultado: Satisfactorio.

Prueba 10: Acceso a todas las rutas prohibidas para asociaciones

Descripción del caso: La asociación con la sesión iniciada no podrá acceder a determinadas rutas o funcionalidades: índice de asociaciones en destino, modificación de otra asociación, modificación de usuarios estudiantes, modificación de eventos ajenos y creación de eventos para asociaciones ajenas.

```
1   /* Trying to access every route prohibited for the association
       should be redirected to error */
```

```
public function test_banned_routes_for_associaton()
{
    $asociation = \App\Models\Asociation::factory(Asociation::class)->create();

    $destination = Destination::find($asociation->destination);

    $user= User::find($asociation->codAsociation);

    $student = \App\Models\Student::factory(Student::class)->create();

    $response = $this->actingAs($user)->get(route('asociations.index'));

    $response->assertViewIs('errors.404');

    $response = $this->actingAs($user)->get(route('asociations-update',[$destination->city,'wrong-association']));

    $response->assertViewIs('errors.404');

    $response = $this->actingAs($user)->get(route('students-update',[$student->username]));

    $response->assertViewIs('errors.404');

    $response = $this->actingAs($user)->get(route('events.edit','wrong-event'));

    $response->assertViewIs('errors.404');

    $response = $this->actingAs($user)->get(route('events.create','wrong-association'));

    $response->assertViewIs('errors.404');
}
```

Resultado: Insatisfactorio.

Mejoras conseguidas: El formulario de actualización de estudiantes era accesible desde cualquier usuario. Ha sido asegurado para no permitir esto.

Prueba 11: Registro correcto de eventos

Descripción del caso: La asociación con la sesión iniciada podrá acceder al formulario de creación de eventos y registrar un evento, lo que llevará a su redirección y almacenamiento de sus datos y la localización en la que toma lugar.

```
/* Posting the information of an event on its route shall redirect to the event profile */
```

```
2   public function test_event_creation_and_redirection()
3   {
4       $asociation = \App\Models\Asociation::factory(Asociation::
            class)->create();
5
6       $destination = Destination::find($asociation->destination);
7
8       $user = User::find($asociation->codAsociation);
9
10      $response = $this->actingAs($user)->post(route('events.store')
            , [
11          'name' => 'example_event',
12          'description' => 'Lorem ipsum dolor sit, amet consectetur
                adipisicing elit. Similique repudiandae molestiae ex
                sit, enim neque dolore commodi. Iure in neque minus
                molestiae totam nemo repellendus laboriosam ducimus
                quis! Deleniti, alias.',
13          'type' => 'SPORTS',
14          'date' => Carbon::now(),
15          'participants' => 0,
16          'cost' => 0,
17          'codAsociation' => $asociation->codAsociation,
18          'password_confirmation' => '12345678',
19          'profile_image' => 'imagen',
20          'location' => "example_location",
21          'search_addr' => "address",
22          'search_latitude' => 12,
23          'search_longitude' => 45,
24      ]);
25
26      $response->assertRedirect(route('asociations-index-events',['
            city' => $destination->city, 'codAsociation' => $asociation
            ->codAsociation]));
27
28
29      $this->assertDatabaseHas('events', [
30          'name' => 'example_event',
31      ]);
32
33      $this->assertDatabaseHas('locations', [
34          'name' => 'example_location',
35      ]);
36  }
```

Resultado: Satisfactorio.

Prueba 12: Inscripción y aceptación de miembros

Descripción del caso: Un usuario estudiante solicita la inscripción a una asociación cerrada y se verifica su almacenamiento. Esta desde su perfil responde a la asociación, redirigiéndose al índice de miembros y comprobando su modificación.

```
 /* Creating and accepting an inscripcion redirect and modify the
    row correctly */
public function test_inscription_and_acceptance_of_user()
{
    $asociation = \App\Models\Asociation::factory(Asociation::
       class)->create();

    $destination = Destination::find($asociation->destination);

    $student = \App\Models\Student::factory(Student::class)->
       create();

    $admin = User::find($asociation->codAsociation);

    $user = User::find($student->username);

    $response = $this->actingAs($user)->patch(route('asociations-
       create_inscription',['codAsociation' => $asociation->
       codAsociation,'student'=>$user->username]));

    $response->assertRedirect(route('asociations-show',['city'=>
       $destination->city,'codAsociation' => $asociation->
       codAsociation]));

    $this->assertDatabaseHas('inscriptions', [
        'user' => $user->username,
        'codAsociation' => $asociation->codAsociation,
        'accepted' => 0
    ]);

    $inscription = Inscription::where('user',$user->username)->
       where('codAsociation',$asociation->codAsociation)->first();

    $response = $this->actingAs($admin)->patch(route('asociations-
       accept_inscription',$inscription->id));

    $response->assertRedirect(route('asociations-all-members',['
       city' => $destination->city,'codAsociation' => $asociation
       ->codAsociation]));

    $this->assertDatabaseHas('inscriptions', [
        'user' => $user->username,
        'codAsociation' => $asociation->codAsociation,
        'accepted' => 1
    ]);
}
```

Resultado: Satisfactorio.

Figura 6.1.4: Resultado de ejecución de pruebas del controlador de asociaciones.

```
[CristianFernandez:~/Escritorio/UNI/TFG/easyerasmus]$ php artisan test tests/Feature/Http/Controllers/AsociationControllerTest.php

  PASS  Tests\Feature\Http\Controllers\AsociationControllerTest
  ✓ get all routes available for associations
  ✓ banned routes for associaton
  ✓ event creation and redirection
  ✓ inscription and acceptance of user

  Tests:  4 passed
  Time:   1.22s
```

6.1.5. Controlador de Administradores por destino

Prueba 13: Acceso a eventos de asociaciones privadas

Descripción del caso: Un administrador puede acceder a asociaciones privadas y ver sus eventos.

```
/* Trying to access a private event should be successful */
public function test_show_private_event()
{
    $event = \App\Models\Event::factory(Event::class)->create();

    $asociation = Asociation::find($event->codAsociation);

    $admin = \App\Models\Admin::factory(Admin::class)->create();

    $user = User::find($admin->username);

    $response = $this->actingAs($user)->get(route('events.show',['
        event' => $event->id,'codAsociation'=>$event->codAsociation
        ]));

    $response->assertViewIs('events.show');
}
```

Resultado: Satisfactorio.

Prueba 14: Acceso a todas las rutas disponibles y específicas para administradores

Descripción del caso: El administrador con la sesión iniciada podrá acceder a todas sus funcionalidades: perfiles de asociaciones, índice de asociaciones en ese destino, lista de asociaciones (vista de usuario), índice de notificaciones, calendario de eventos y método de búsqueda.

```
/* Trying to access every route available for the admin should be
    satisfactory */
public function
    test_get_all_routes_available_for_destination_admins()
{
    $event = \App\Models\Event::factory(Event::class)->create();

    $asociation = Asociation::find($event->codAsociation);
```

```
7
8       $admin = \App\Models\Admin::factory(Admin::class)->create();
9
10      $user = User::find($admin->username);
11
12      $destination = Destination::find($admin->destination);
13
14      $response = $this->actingAs($user)->get(route('asociations-
           show',[$destination->city,$asociation->codAsociation]));
15
16      $response->assertViewIs('asociations.show');
17
18      $response = $this->actingAs($user)->get(route('asociations.
           index'));
19
20      $response->assertViewIs('asociations.index_associations');
21
22      $response = $this->actingAs($user)->get(route('asociations-
           index_city',$destination->city));
23
24      $response->assertViewIs('asociations.index_city');
25
26      $response = $this->actingAs($user)->get(route('notifications.
           index'));
27
28      $response->assertViewIs('notification.index');
29
30      $response = $this->actingAs($user)->get(route('events-calendar
           ',$destination->city));
31
32      $response->assertViewIs('events.calendar');
33
34      $response = $this->actingAs($user)->get('/search?city='.
           $destination->city);
35
36      $response->assertViewIs('search');
37  }
```

Resultado: Insatisfactorio.

Mejoras conseguidas: El acceso al método de búsqueda devolvía una vista erronea si no se mandaban parámetros en la URL. Ha sido solucionado para redireccionar correctamente con el parámetro POST de ciudad.

Prueba 15: Creación de notificación de desactivación a asociación y posterior reactivación

Descripción del caso: El administrador con la sesión iniciada podrá crear asociaciones dirigidas a las asociaciones en destino, el envío de una asociación de desactivación redireccionará al índice de notificaciones y almacenará esta con estado *ENVIADA*. Además, al crear esta notificación se desactivará automaticamente a la asociación. Si la asociación accede a la

notificación recibida y la responde, se modificará con estado *RESPUESTA*. Posteriormente, si el administrador accede de nuevo y envía la solicitud de borrado de notificación, esta se eliminará y la asociación volverá a reactivarse.

```
 /* Notifying an association could mean in its deactivation, with
    the possibility of reactivation later */
public function test_notify_asociation_deactivate_and_reactivate()
{
    $event = \App\Models\Event::factory(Event::class)->create();

    $asociation = Asociation::find($event->codAsociation);

    $admin = \App\Models\Admin::factory(Admin::class)->create();

    $user = User::find($admin->username);

    $userAsociation = User::find($asociation->codAsociation);

    $destination = Destination::find($admin->destination);

    $response = $this->actingAs($user)->get(route('notifications.
       create',$asociation->codAsociation));

    $response->assertViewIs('notification.create');

    $response = $this->actingAs($user)->post(route('notifications.
       store'), [
        'admin' => $user->username,
        'description' => 'Lorem ipsum dolor sit. imilique
           repudiandae molestiae ex sit, enim neque dolore commodi
           . Iure in neque minus molestiae totam nemo repellendus
           laboriosam ducimus quis! Deleniti, alias.',
        'state' => 0,
        'deactivate' => 1,
        'codAsociation' => $asociation->codAsociation,
    ]);

    $response->assertRedirect(route('notifications.index'));

    $this->assertDatabaseHas('notifications', [
        'codAsociation' => $asociation->codAsociation,
        'admin' => $user->username,
        'state' => 0,

    ]);

    $this->assertDatabaseHas('asociations', [
        'codAsociation' => $asociation->codAsociation,
        'state' => 0
    ]);

```

```
42      $notification=Notification::where('codAsociation', $asociation
           ->codAsociation)->where('admin',$user->username)->first();
43
44      $response = $this->actingAs($userAsociation)->patch(route('
           notifications.respond',$notification->id));
45
46      $this->assertDatabaseHas('notifications', [
47          'codAsociation' => $asociation->codAsociation,
48          'admin' => $user->username,
49          'state' => 1,
50      ]);
51
52      $response = $this->actingAs($user)->delete(route('
           notifications.destroy',$notification->id));
53
54      $this->assertDatabaseMissing('notifications', [
55          'codAsociation' => $asociation->codAsociation,
56          'admin' => $user->username,
57      ]);
58
59      $this->assertDatabaseHas('asociations', [
60          'codAsociation' => $asociation->codAsociation,
61          'state' => 1
62      ]);
63
64  }
```

Resultado: Satisfactorio.

Figura 6.1.5: Resultado de ejecución de pruebas del controlador de administradores por destino.

```
[CristianFernandez:~/Escritorio/UNI/TFG/easyerasmus]$ php artisan test tests/Feature/AdminTest.php

  PASS  Tests\Feature\AdminTest
  ✓ show private event
  ✓ get all routes available for destination admins
  ✓ notify asociation deactivate and reactivate

  Tests:  3 passed
  Time:   1.23s
```

6.1.6. Controlador de Estudiantes Erasmus

Prueba 16: Acceso a todas las rutas específicas para Estudiantes Erasmus

Descripción del caso: El estudiante con la sesión iniciada podrá acceder a todas sus funcionalidades: asociaciones a las que pertenece, eventos a los que participa, perfil de estudiante y modificación de su perfil.

```
1  /* Trying to access every route available for the student should
      be satisfactory */
2  public function test_get_all_routes_available_for_students()
3  {
```

```
4       $student = \App\Models\Student::factory(Student::class)->
            create();
5
6       $uni = University::find($student->university);
7
8       $destination = Destination::find($uni->destination_id);
9
10      $user= User::find($student->username);
11
12      $response = $this->actingAs($user)->get(route('students-
            associations'));
13
14      $response->assertViewIs('students.asociations');
15
16      $response = $this->actingAs($user)->get(route('students-events
            '));
17
18      $response->assertViewIs('students.events');
19
20      $response = $this->actingAs($user)->get(route('students-show',
            $user->username));
21
22      $response->assertViewIs('students.show');
23
24      $response = $this->actingAs($user)->get(route('students-update
            ',$user->username));
25
26      $response->assertViewIs('students.update');
27  }
```

Resultado: Satisfactorio.

Prueba 17: Acceso a todas las rutas prohibidas para Estudiantes Erasmus

Descripción del caso: El estudiante con la sesión iniciada no podrá acceder a determinadas rutas o funcionalidades: índice de asociaciones por destino, modificación de asociaciones, índice de miembros de asociaciones, creación de eventos e índice de notificaciones.

```
1   /* Trying to access every route prohibited for the student should
        be redirected to error */
2   public function test_get_banned_routes_for_students()
3   {
4       $student = \App\Models\Student::factory(Student::class)->
            create();
5
6       $uni = University::find($student->university);
7
8       $destination = Destination::find($uni->destination_id);
9
10      $user= User::find($student->username);
11
```

```
12      $event = \App\Models\Event::factory(Event::class)->create();
13
14      $asociation = Asociation::find($event->codAsociation);
15
16      $response = $this->actingAs($user)->get(route('asociations-
           index-events',[$destination->city,$asociation->
           codAsociation]));
17
18      $response->assertViewIs('errors.404');
19
20      $response = $this->actingAs($user)->get(route('asociations-
           update',[$destination->city,$asociation->codAsociation]));
21
22      $response->assertViewIs('errors.404');
23
24      $response = $this->actingAs($user)->get(route('asociations-all
           -members',[$destination->city,$asociation->codAsociation]))
           ;
25
26      $response->assertViewIs('errors.404');
27
28      $response = $this->actingAs($user)->get(route('events.create',
           $asociation->codAsociation));
29
30      $response->assertViewIs('errors.404');
31
32      $response = $this->actingAs($user)->get(route('notifications.
           index'));
33
34      $response->assertViewIs('errors.404');
35  }
```

Resultado: Insatisfactorio.

Mejoras conseguidas: El acceso al índice de miembros devolvía la vista específica. Ha sido solucionado para no permitirlo y redireccionar a la vista de error.

Prueba 18: Borrado de Estudiantes Erasmus

Descripción del caso: El estudiante con la sesión iniciada podrá acceder a su perfil y envíar la solicitud DELETE para eliminar su perfil. Debería ser redireccionado a la página de inicio con la sesión cerrada y los datos del usuario haber sido eliminados con seguridad.

```
1   /* Student destroy should safe delete the correspondat user and
       its data */
2   public function test_delete_of_student()
3   {
4
5       $student = \App\Models\Student::factory(Student::class)->
           create();
6
```

```
7       $uni = University::find($student->university);
8
9       $destination = Destination::find($uni->destination_id);
10
11      $user= User::find($student->username);
12
13      $response = $this->actingAs($user)->delete(route('students-
           destroy',$user->username));
14
15      $response->assertRedirect(route('inicio'));
16
17      $this->assertDatabaseMissing('users', [
18          'username' => $user->username,
19      ]);
20
21      $this->assertDatabaseMissing('students', [
22          'username' => $user->username,
23      ]);
24  }
```

Resultado: Insatisfactorio.

Mejoras conseguidas: Si el usuario no contaba con imagen de perfil, en el borrado se intentaba eliminar el directorio actual. Ha sido modificado para comprobar si la imagen está definida y existe.

Prueba 19: Participación en eventos de asociaciones pertenecientes

Descripción del caso: El estudiante con la sesión iniciada e inscrito a una asociación, podrá acceder a sus eventos y envíar la solicitud PATCH para participar en ellos. Debería ser redireccionado a la página del evento con la participación almacenada. A su vez, podría cancelar su participación al enviar la solicitud DELETE desde esta misma página, y los datos de la relación serían eliminados con seguridad.

```
1   /* Students can participate in events and cancel it anytime */
2   public function test_participation_and_cancellation_of_event()
3   {
4       $event = \App\Models\Event::factory(Event::class)->create();
5
6       $asociation = Asociation::find($event->codAsociation);
7
8       $student = \App\Models\Student::factory(Student::class)->
           create();
9
10      $admin = User::find($asociation->codAsociation);
11
12      $user = User::find($student->username);
13
14      $inscription = \App\Models\Inscription::factory(Inscription::
           class)->make([
15          'user' => $user->username,
```

```
16            'codAsociation' => $asociation->codAsociation,
17            'accepted' => 1,
18        ]);
19
20        $inscription->save();
21
22        $response = $this->actingAs($user)->patch(route('events.
             create_participation',['id' => $event->id,'student'=>$user
             ->username]));
23
24        $response->assertRedirect(route('events.show',['event' =>
             $event->id, 'codAsociation' => $asociation->codAsociation])
             );
25
26        $this->assertDatabaseHas('participations', [
27            'user' => $user->username,
28            'idEvent' => $event->id
29        ]);
30
31        $response = $this->actingAs($user)->delete(route('events.
             destroy_participation',['id' => $event->id,'student'=>$user
             ->username]));
32
33        $response->assertRedirect(route('events.show',['event' =>
             $event->id, 'codAsociation' => $asociation->codAsociation])
             );
34
35        $this->assertDatabaseMissing('participations', [
36            'user' => $user->username,
37            'idEvent' => $event->id
38        ]);
39    }
```

Resultado: Satisfactorio.

Prueba 20: Intento de acceso a eventos de asociaciones privadas

Descripción del caso: El estudiante que no pertenezca a una asociación privada, no podrá acceder a sus eventos, sino ser redirigido a una vista de error.

```
1   /* Trying to access private events */
2   public function test_show_private_event()
3   {
4       $event = \App\Models\Event::factory(Event::class)->create();
5
6       $asociation = Asociation::find($event->codAsociation);
7
8       $student = \App\Models\Student::factory(Student::class)->
           create();
9
10      $user = User::find($student->username);
```

```
11
12      $response = $this->actingAs($user)->get(route('events.show',['
           event' => $event->id,'codAsociation'=>$event->codAsociation
           ]));
13
14      $response->assertViewIs('errors.404');
15  }
```

Resultado: Insatisfactorio.

Mejoras conseguidas: El test falló en su totalidad. El evento podía ser accedido por cualquier usuario. Se ha modificado para permitirse solo en determinados casos (p.e.: estar inscrito en la asociación o ser administrador del destino).

Figura 6.1.6: Resultado de ejecución de pruebas del controlador de estudiantes.

```
[CristianFernandez:~/Escritorio/UNI/TFG/easyerasmus]$ php artisan test tests/Feature/Http/Controllers/StudentControllerTest.php

  PASS  Tests\Feature\Http\Controllers\StudentControllerTest
  ✓ get all routes available for students
  ✓ get banned routes for students
  ✓ delete of student
  ✓ participation and cancellation of event
  ✓ show private event

  Tests:  5 passed
  Time:   1.19s
```

6.2. Tests de caja negra

Las pruebas de caja negra *"se enfocan en las entradas y salidas del sistema, sin preocuparnos de la estructura interna del programa. Para detallar cuáles deben ser estas entradas y salidas, nos centraremos en los requerimientos de software y especificaciones funcionales"*, como se detalla en La Oficina De Proyectos De Informática (2017). A su vez, a diferencia de las pruebas de caja blanca, nos permitirán trabajar con un número mayor de registros fácilmente, y evaluar la integración de ellos en el sistema. El diseño de pruebas estará basado en los diferentes roles de usuario que tendrá la aplicación.

6.2.1. Administradores de destinos

Prueba 1: Creación de destinos y universidad

Descripción del caso: El administrador enviará la información de un destino, junto con sus imágenes y el usuario gestor. Si los datos son correctos, se creará el registro correspondiente.

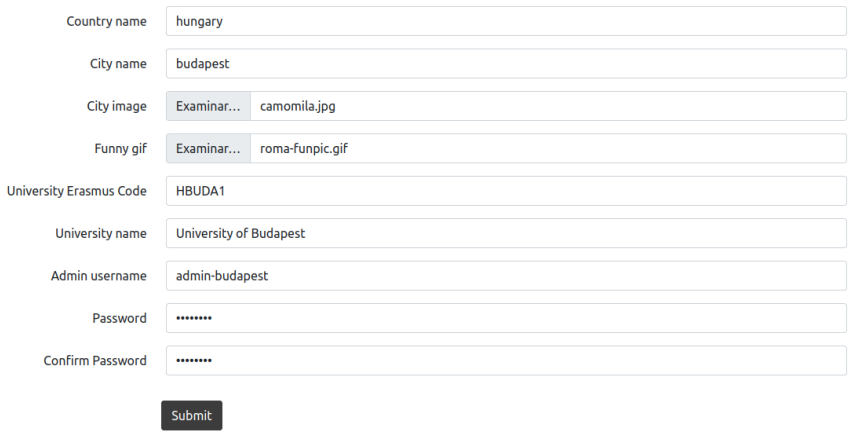
Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso y análisis de valores borde.

- **1º caso:** Datos de entrada: Registrar destino correcto. Resultado esperado (Salida): Redirección al índice de destinos.
- **2º caso:** Datos de entrada: Registrar destino repetido. Resultado esperado (Salida): Error por pantalla de repetición de ciudad.

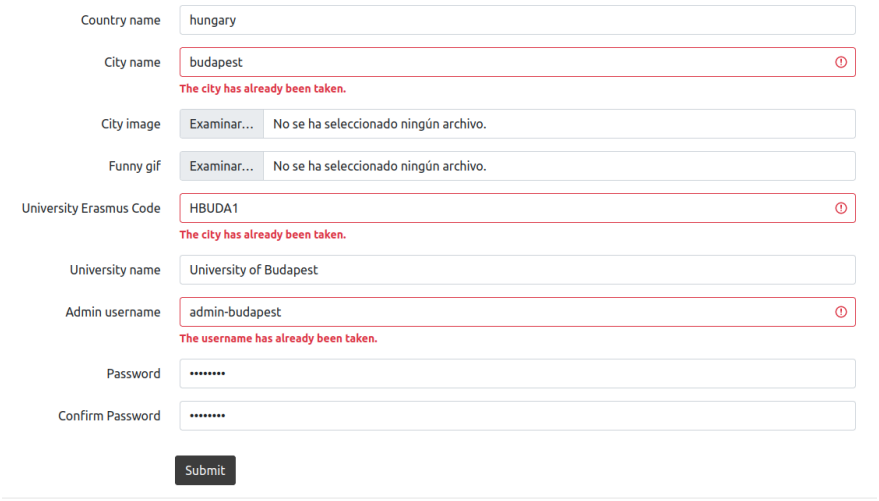
- **3º caso:** Datos de entrada: Registrar universidad repetida. Resultado esperado (Salida): Error por pantalla de repetición de universidad.
- **4º caso:** Datos de entrada: Registrar usuario administrador repetido. Resultado esperado (Salida): Error por pantalla de repetición de usuario.

Mejoras conseguidas: Los casos de prueba **2 y 3** dieron resultados insatisfactorios. Por lo cual se ha corregido la salida para mostrar en pantalla el error correspondiente.

Prueba 1, Caso 1: Registro de destino correcto.



Prueba 1, Caso 2: Registro de destino con datos repetidos.



6.2.2. Asociaciones Erasmus

Prueba 2: Registro de asociación, correcto almacenamiento y redirección

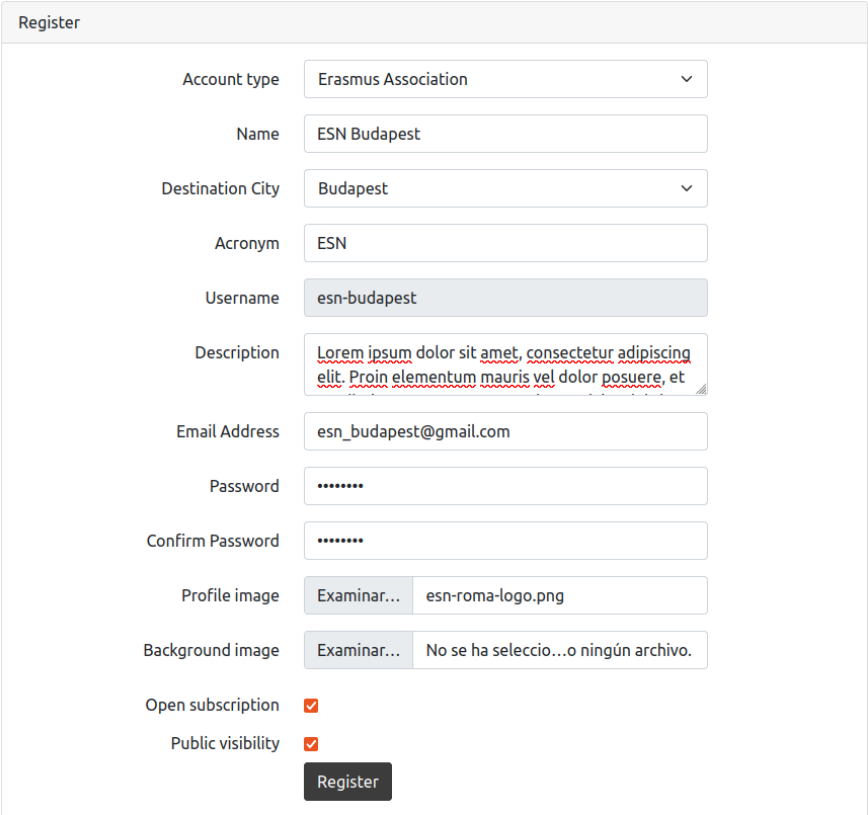
Descripción del caso: El usuario enviará la información de una asociación, seleccionando su ciudad y junto con sus imágenes. Si los datos son correctos, se creará el registro correspondiente.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso y análisis de valores borde.

- **1º caso:** Datos de entrada: Registrar asociación correcta. Resultado esperado (Salida): Redirección al perfil de asociación con estado suspendida.
- **2º caso:** Datos de entrada: Registrar asociación con usuario repetido. Resultado esperado (Salida): Error por pantalla de repetición de usuario.
- **3º caso:** Datos de entrada: Registrar email repetido. Resultado esperado (Salida): Error por pantalla de repetición de email.
- **4º caso:** Datos de entrada: Registrar asociación con acrónimo erróneo. Resultado esperado (Salida): Borrado automático de los caracteres inválidos.

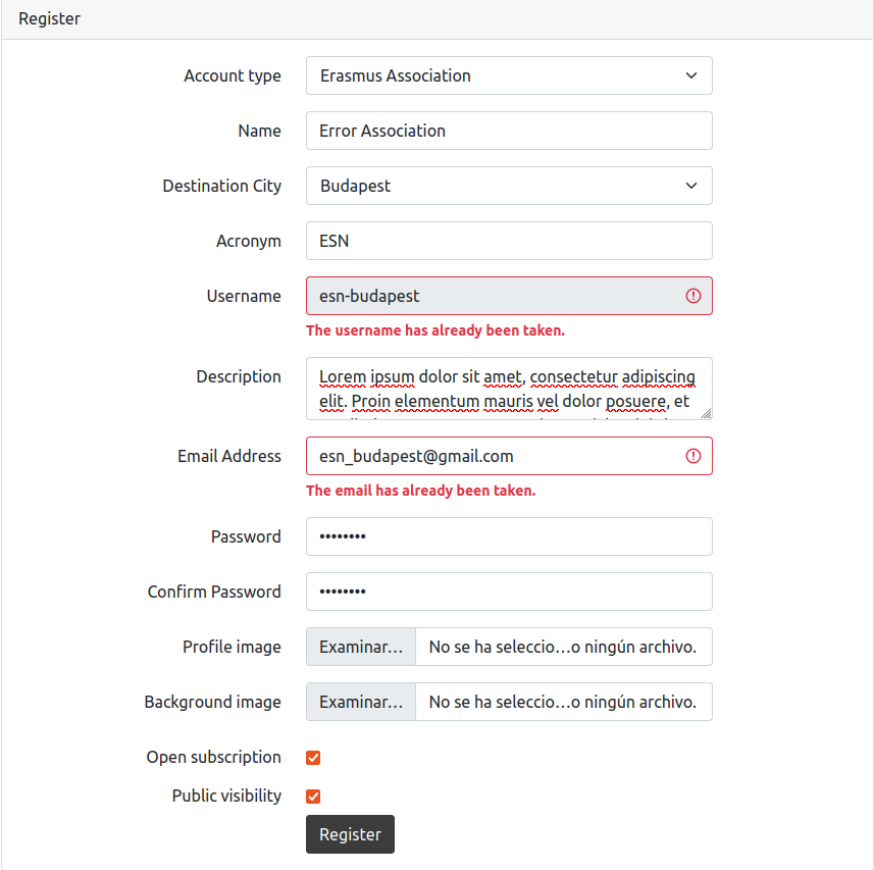
Mejoras conseguidas: El caso de prueba **3** mostró resultados erróneos. Por lo cual se ha corregido el input para conseguir el borrado automático de caracteres problemáticos.

Prueba 2, Caso 1.1: Registro de asociación correcto.

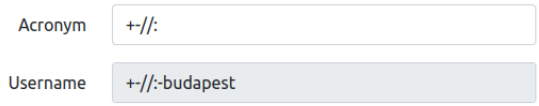


Prueba 2, Caso 1.2: Redirección con perfil inactiva.

Prueba 2, Caso 2 y 3: Repetición de usuario y email.



Prueba 2, Caso 4: Intento de registro de acrónimo incorrecto.



Prueba 3: Registro y actualización de evento, correcto almacenamiento y redirección

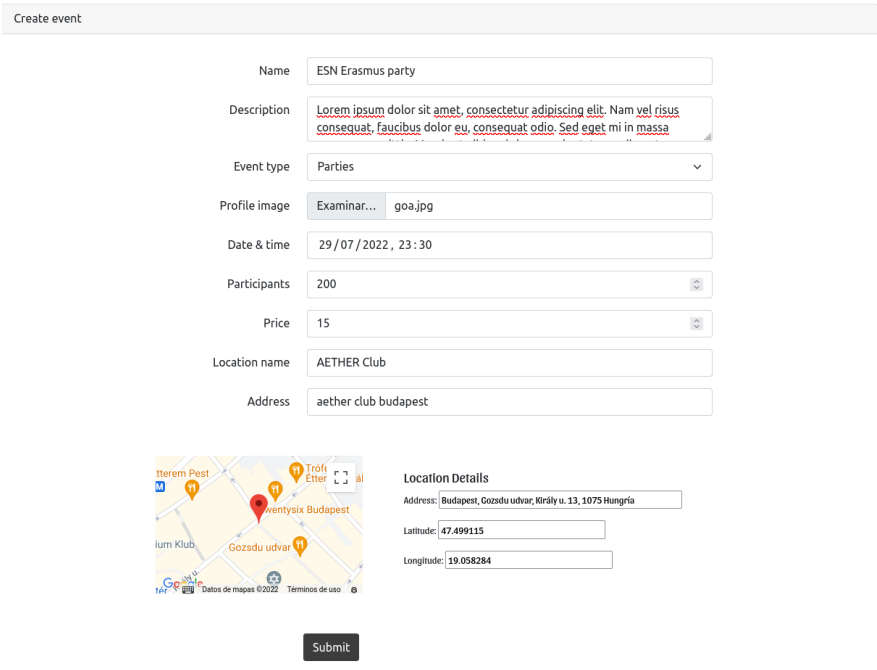
Descripción del caso: El usuario enviará la información de un evento, seleccionando la ubicación y junto con sus imágenes. Si los datos son correctos, se creará el registro correspondiente.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso y análisis de valores borde.

- **1º caso:** Datos de entrada: Registrar evento correcto. Resultado esperado (Salida): Redirección al índice de eventos con este almacenado.
- **2º caso:** Datos de entrada: Intento de registro de evento con caracteres problemáticos para participantes o precio. Resultado esperado (Salida): Error por pantalla y no permitir la inserción.
- **3º caso:** Datos de entrada: Registrar nuevos datos de evento. Resultado esperado (Salida): Redirección al índice de eventos con este actualizado.

Mejoras conseguidas: En el caso de prueba **2**, el input de participantes aceptaba números negativos. Se ha corregido el input para conseguir el borrado automático de signos especiales. Los demás campos están correctamente limitados, solo se permiten fechas a partir del día actual y el nombre del evento puede contener caracteres especiales pues no se muestra en la URL.

Prueba 3, Caso 1.1: Registro de evento correcto.



Prueba 3, Caso 1.2: Redirección al índice de eventos con el evento creado.

Id	Name	Description	Type	Profile Image	Date	Location	Max Participants	Participating	Cost	
1	ESN Erasmus party	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam vel risus consequat, faucibus dolor eu, consequat odio. Sed eget mi in massa accumsan sagittis. Mauris et nibh quis lorem vulputate condimentum nec id tortor. Sed sed nunc ornare, bibendum ligula consequat, lobortis elit. Sed et mi magna. Pellentesque ullamcorper congue diam in dapibus. Praesent neque massa, fermentum non urna quis,	PARTIES		2022-07-29 23:30:00	AETHER Club	200	0	15.00	Show Edit Delete

Prueba 3, Caso 3: Modificación de evento y actualización de sus datos.

Id	Name	Description	Type	Profile Image	Date	Location	Max Participants	Participating	Cost	
1	ESN Erasmus updated party	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam vel risus consequat, faucibus dolor eu, consequat odio. Sed eget mi in massa accumsan sagittis. Mauris et nibh quis lorem vulputate condimentum nec id tortor. Sed sed nunc ornare, bibendum ligula consequat, lobortis elit. Sed et mi magna. Pellentesque ullamcorper congue diam in dapibus. Praesent neque massa, fermentum non urna quis,	PARTIES		2022-07-30 23:30:00	Jack Doyle's Irish Pub	100	0	5.00	Show Edit Delete

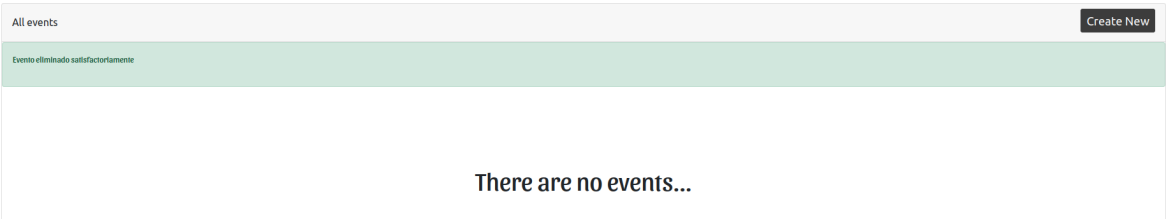
Prueba 4: Eliminación de evento y sus datos

Descripción del caso: El usuario elimina un evento desde el índice. Si se produce correctamente se debe eliminar el registro.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

- **1º caso:** Datos de entrada: Eliminar evento seleccionado. Resultado esperado (Salida): Redirección al índice de eventos sin este disponible.

Prueba 4, Caso 1: Eliminación de evento.



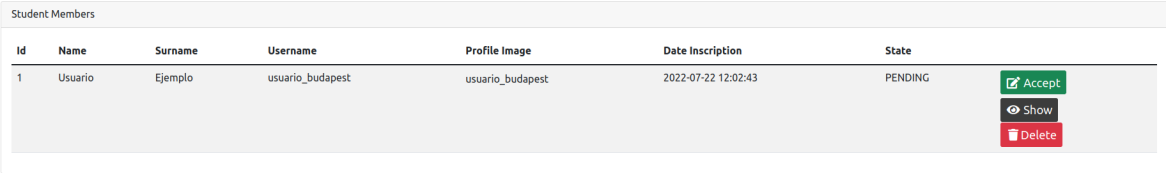
Prueba 5: Aceptación y eliminación de usuario en asociación

Descripción del caso: La asociación requiere de solicitud y un usuario ha enviado una previamente. Desde el índice de miembros se podrá aceptar al usuario, observar su perfil y eliminar su inscripción.

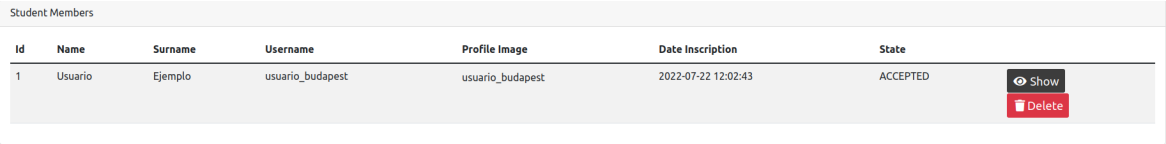
Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

- **1º caso:** Datos de entrada: Aceptar inscripción de usuario seleccionado. Resultado esperado (Salida): Redirección al índice de miembros con la solicitud aceptada.
- **2º caso:** Datos de entrada: Mostrar usuario miembro seleccionado. Resultado esperado (Salida): Redirección al perfil de usuario.
- **3º caso:** Datos de entrada: Eliminar inscripción de usuario seleccionado (aceptada o no). Resultado esperado (Salida): Redirección al índice de miembros sin este disponible.

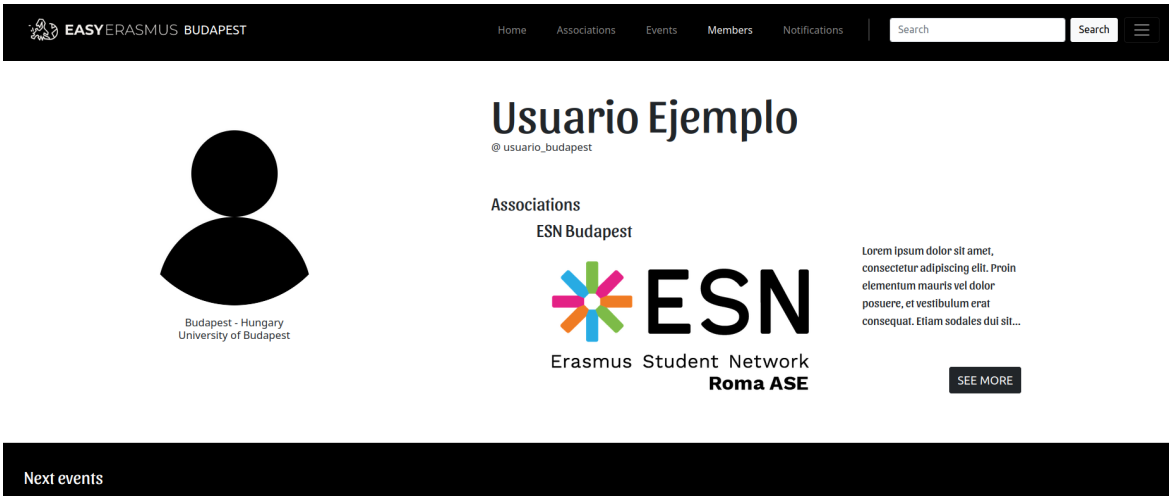
Prueba 5, Caso 1.1: Aceptar inscripción de usuario.



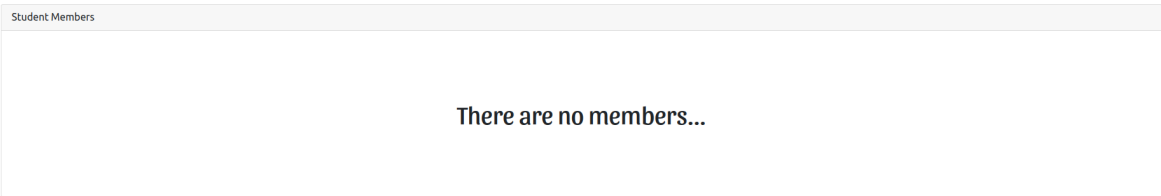
Prueba 5, Caso 1.2: Redirección a índice de usuarios.



Prueba 5, Caso 2: Mostrar usuario.



Prueba 5, Caso 3: Eliminar inscripción de usuario.



Prueba 6: Resolución de notificaciones y cambios de estado

Descripción del caso: Un administrador ha notificado a la asociación previamente, dependiendo del tipo de notificación de resolverá de una forma u otra.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso y Tablas de decisión

Condición	Regla 1	Regla 2	Regla 3
Recepción de notificación sencilla	Verdadero	Falso	Verdadero
Recepción de notificación de desactivación	Falso	Verdadero	Verdadero
Resultado			
Estado de asociación	Activa	Inactiva	Inactiva

Con la información de la tabla de decisión, se definen los siguientes casos de prueba:

- **1º caso:** Datos de entrada: Recepción de notificación sencilla. Aceptación por parte de la asociación. Resultado esperado (Salida): Redirección al índice de notificaciones, manteniendo el mismo estado en el que se encontraba previamente.
- **2º caso:** Datos de entrada: Recepción de notificación que requiere desactivación. Resultado esperado (Salida): Redirección al índice de notificaciones, con la notificación enviada de vuelta y manteniendo el estado inactivo.
- **3º caso:** Datos de entrada: Recepción de notificación sencilla y de desactivación. Asociación resuelve la notificación sencilla. Resultado esperado (Salida): Redirección al índice de notificaciones, manteniendo el estado de desactivación que manda la otra notificación.

Prueba 6, Caso 1: Recepción de notificación sencilla.

Id	Description	State	Deactivate association	Created	Updated	
4	Esta notificación hace que la asociación pase a estar desactivada	SEND	YES	22 Jul 2022 14:06	22 Jul 2022 14:06	Resolve

Prueba 6, Caso 3.1: Recepción de notificación de desactivación.

Id	Description	State	Deactivate association	Created	Updated	
3	Esta es una notificación simple que no requiere de desactivación.	SEND	NO	22 Jul 2022 14:06	22 Jul 2022 14:06	Resolve
4	Esta notificación hace que la asociación pase a estar desactivada	SEND	YES	22 Jul 2022 14:06	22 Jul 2022 14:06	Resolve

Prueba 6, Caso 3.2: Aceptación de ambas notificaciones.

Id	Description	State	Deactivate association	Created	Updated
4	Esta notificación hace que la asociación pase a estar desactivada	ANSWERED	YES	22 Jul 2022 14:06	22 Jul 2022 14:09

Prueba 7: Imposibilidad de creación de eventos al encontrarse inactiva

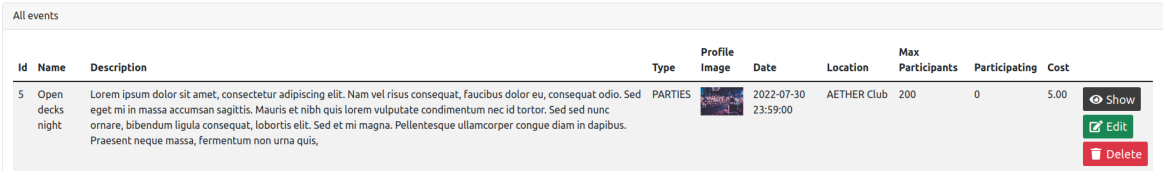
Descripción del caso: La asociación inactiva no tiene acceso a la vista de creación de eventos.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

- **1º caso:** Datos de entrada: Desde el perfil de la asociación. Resultado esperado (Salida): No aparece el botón de creación de evento.
- **2º caso:** Datos de entrada: Desde el índice de eventos. Resultado esperado (Salida): No aparece el botón de creación de evento.
- **3º caso:** Datos de entrada: Escribiendo directamente la URL de la vista. Resultado esperado (Salida): Redirección a página de error.

Prueba 7, Caso 1: Asociación inactiva.

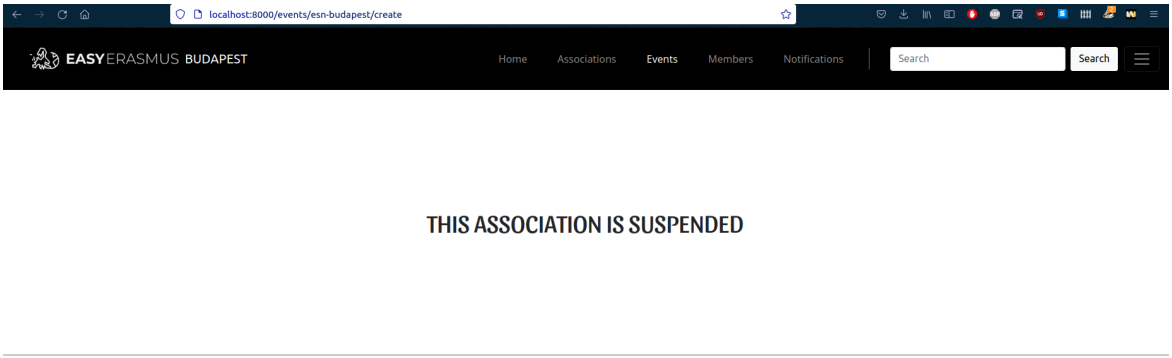
Prueba 7, Caso 2: Índice de eventos de asociación inactiva.



All events

Id	Name	Description	Type	Profile Image	Date	Location	Max Participants	Participating	Cost	
5	Open decks night	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam vel risus consequat, faucibus dolor eu, consequat odio. Sed eget mi in massa accumsan sagittis. Mauris et nibh quis lorem vulputate condimentum nec id tortor. Sed sed nunc ornare, bibendum ligula consequat, lobortis elit. Sed et mi magna. Pellentesque ullamcorper congue diam in dapibus. Praesent neque massa, fermentum non urna quis,	PARTIES		2022-07-30 23:59:00	AETHER Club	200	0	5.00	Show Edit Delete

Prueba 7, Caso 3: Intento de acceso a creación de eventos.



Prueba 8: Imposibilidad de acceso a URLs prohibidas

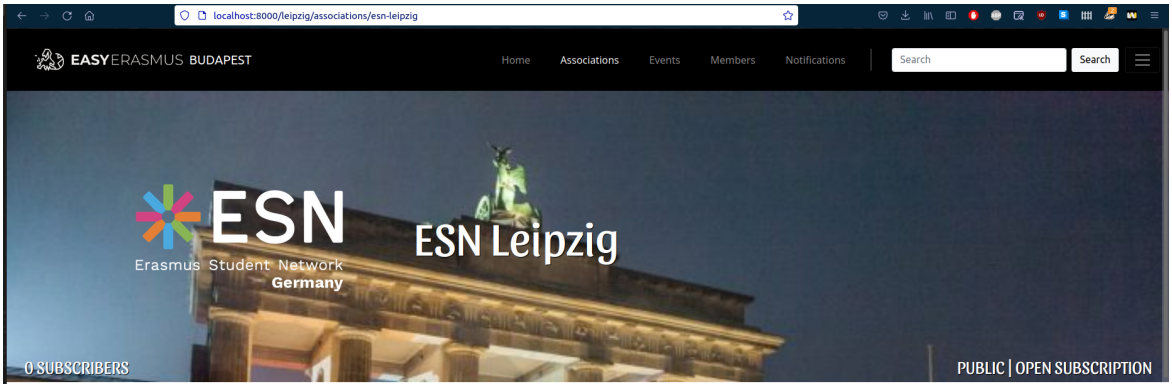
Descripción del caso: La asociación no puede tener acceso a determinadas funciones o vistas, ni mucho menos en otros destinos.

Técnica de prueba de caja negra: Requerimiento no funcional

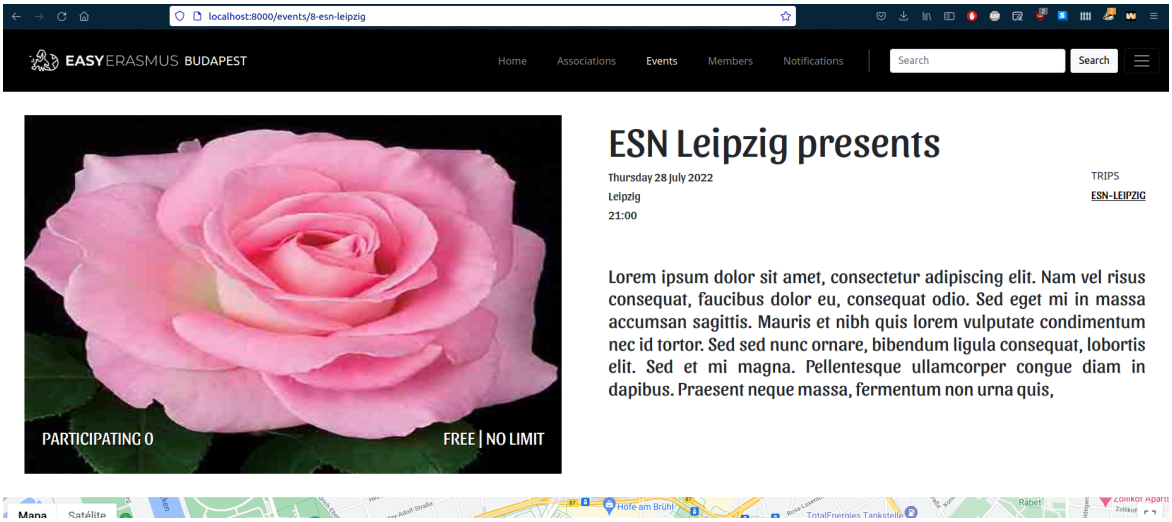
- **1º caso:** Datos de entrada: Modificación de URL para acceder a índice de asociaciones en otra ciudad. Resultado esperado (Salida): Error de página no encontrada.
- **2º caso:** Datos de entrada: Modificación de URL para acceder a asociación en otra ciudad. Resultado esperado (Salida): Error de página no encontrada.
- **3º caso:** Datos de entrada: Modificación de URL para acceder a evento en otra ciudad. Resultado esperado (Salida): Error de página no encontrada.
- **4º caso:** Datos de entrada: Modificación de URL para acceder a calendario de eventos. Resultado esperado (Salida): Error de página no encontrada.

Mejoras conseguidas: Los casos de prueba **2, 3 y 4** mostraron los resultados indeseables. Por lo cual se ha corregido el acceso a esta información.

Prueba 8, Caso 2: Intento de acceso a asociación de otro destino.



Prueba 8, Caso 3: Intento de acceso a evento de otro destino.



6.2.3. Administradores por destino

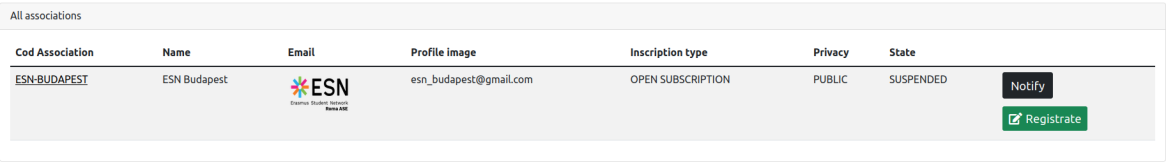
Prueba 9: Aceptación de asociación, correcto almacenamiento y redirección

Descripción del caso: Tras el registro de una asociación, aparecerá en el índice del administrador y tendrá la opción de registrarla o notificarla de cambios. Si es aceptada, pasará a ser activada al instante.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

- **1º caso:** Datos de entrada: Aceptación. Resultado esperado (Salida): Cambio de estado de la asociación y redirección a índice.
- **2º caso:** Datos de entrada: Notificar asociación sin desactivarla. Resultado esperado (Salida): Mantener el estado de la asociación y redirección a índice de notificaciones.
- **3º caso:** Datos de entrada: Notificar asociación con desactivación. Resultado esperado (Salida): Mantener el estado de la asociación y redirección a índice de notificaciones. Cuando esta notificación sea resuelta, se dará de alta a la asociación

Prueba 9, Caso 1: Registro de nueva asociación en el sistema.



Prueba 9, Caso 2: Envío de notificación sencilla a asociación.

Prueba 9, Caso 3.1: Envío de notificación de desactivación.

Notification

Id	Description	State	Deactivate association	Created	Updated	
1	quiero mantener tu estado desactivado quiero mantener tu estado desactivado	SEND	NO	22 Jul 2022 10:49	22 Jul 2022 10:49	Resolve
2	la asociación será activada cuando se finalice este proceso	SEND	YES	22 Jul 2022 10:52	22 Jul 2022 10:52	Resolve

Prueba 9, Caso 3.2: Resolución de notificación de desactivación.

Notification

Id	CodAssociation	Description	State	Deactivate association	Created	Updated	
2	ESN-BUDAPEST	la asociación será activada cuando se finalice este proceso	ANSWERED	YES	22 Jul 2022 10:52	22 Jul 2022 10:53	Accept Resend

Prueba 9, Caso 3.3: Alta automática de asociación.

ESN BUDAPEST

ESN
Erasmus Student Network
Roma ASE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin elementum mauris vel dolor posuere, et vestibulum erat consequat. Etiam sodales dui sit...

There are no events...

SEE MORE

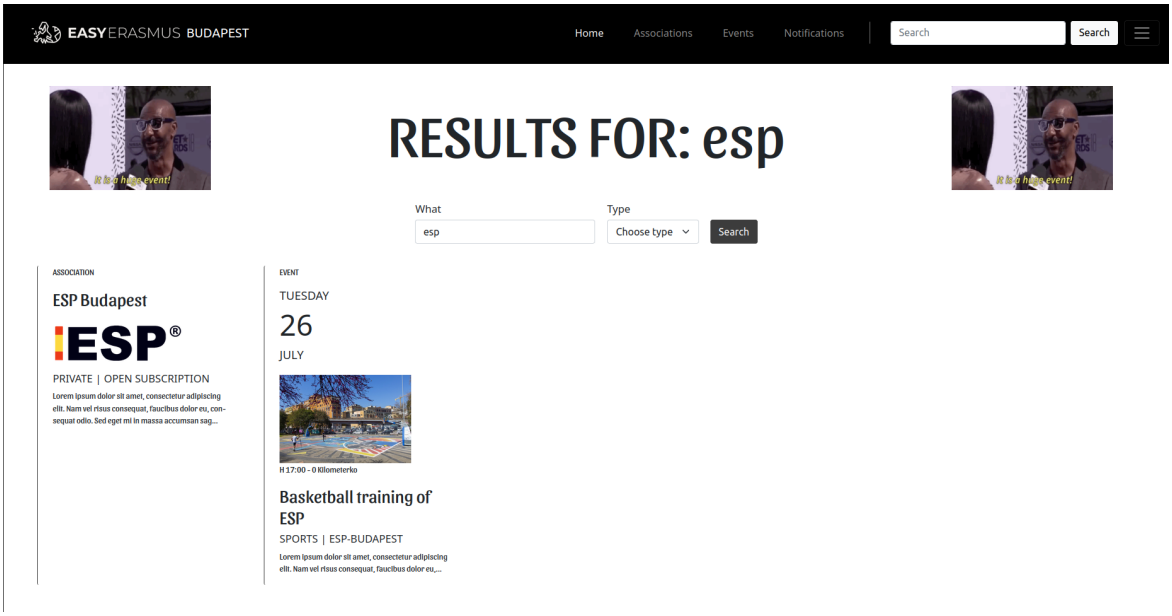
Prueba 10: Verificar que administrador puede visualizar eventos de asociaciones privadas

Descripción del caso: Tras el registro de una asociación privada, sus eventos podrán ser visualizados por el administrador.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

- **1º caso:** Datos de entrada: Página de inicio del destino. Resultado esperado (Salida): Los eventos próximos de la asociación aparecen.
- **2º caso:** Datos de entrada: Página de asociaciones en el destino. Resultado esperado (Salida): La asociación aparece con eventos próximos.
- **3º caso:** Datos de entrada: Página de calendario de eventos en el destino. Resultado esperado (Salida): Los eventos de la asociación se muestran.
- **4º caso:** Datos de entrada: Búsqueda de eventos. Resultado esperado (Salida): Los eventos que coincidan con la búsqueda aparecen.
- **5º caso:** Datos de entrada: Perfil de la asociación. Resultado esperado (Salida): Los eventos de la asociación se muestran.

Prueba 10, Caso 4: Búsqueda de evento de asociación privada.



6.2.4. Usuarios Estudiantes Erasmus

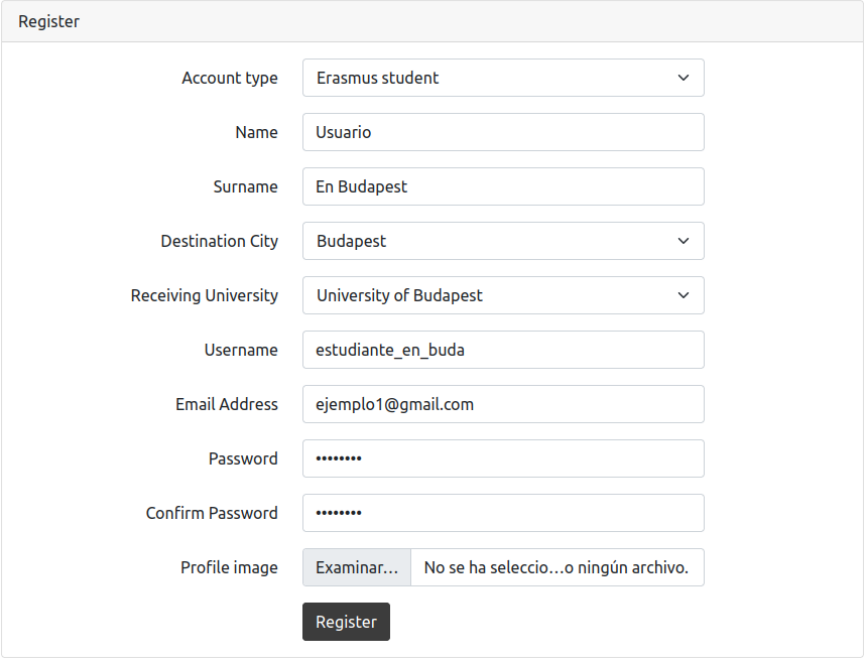
Prueba 11: Registro de estudiante, correcto almacenamiento y redirección

Descripción del caso: El usuario enviará su información, seleccionando su ciudad y universidad. Si los datos son correctos, se creará el registro correspondiente.

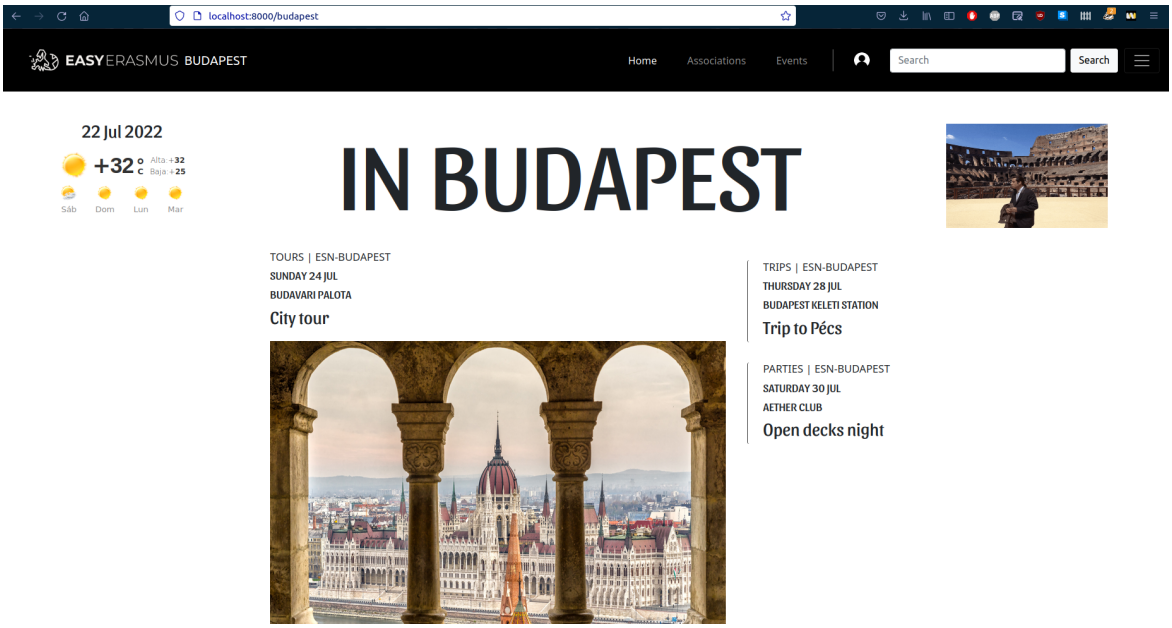
Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso y análisis de valores borde.

- **1º caso:** Datos de entrada: Registrar estudiante correcto. Resultado esperado (Salida): Redirección al inicio del destino.
- **2º caso:** Datos de entrada: Registrar estudiante con usuario repetido. Resultado esperado (Salida): Error por pantalla de repetición de usuario.
- **3º caso:** Datos de entrada: Registrar email repetido. Resultado esperado (Salida): Error por pantalla de repetición de email.
- **4º caso:** Datos de entrada: Registrar usuario con caracteres prohibidos. Resultado esperado (Salida): Borrado automático de los caracteres inválidos.

Prueba 11, Caso 1.1: Registro correcto de estudiante.



Prueba 11, Caso 1.2: Redirección a página de inicio de destino.



Prueba 12: Verificar que estudiante no puede visualizar eventos de asociaciones privadas o inactivas

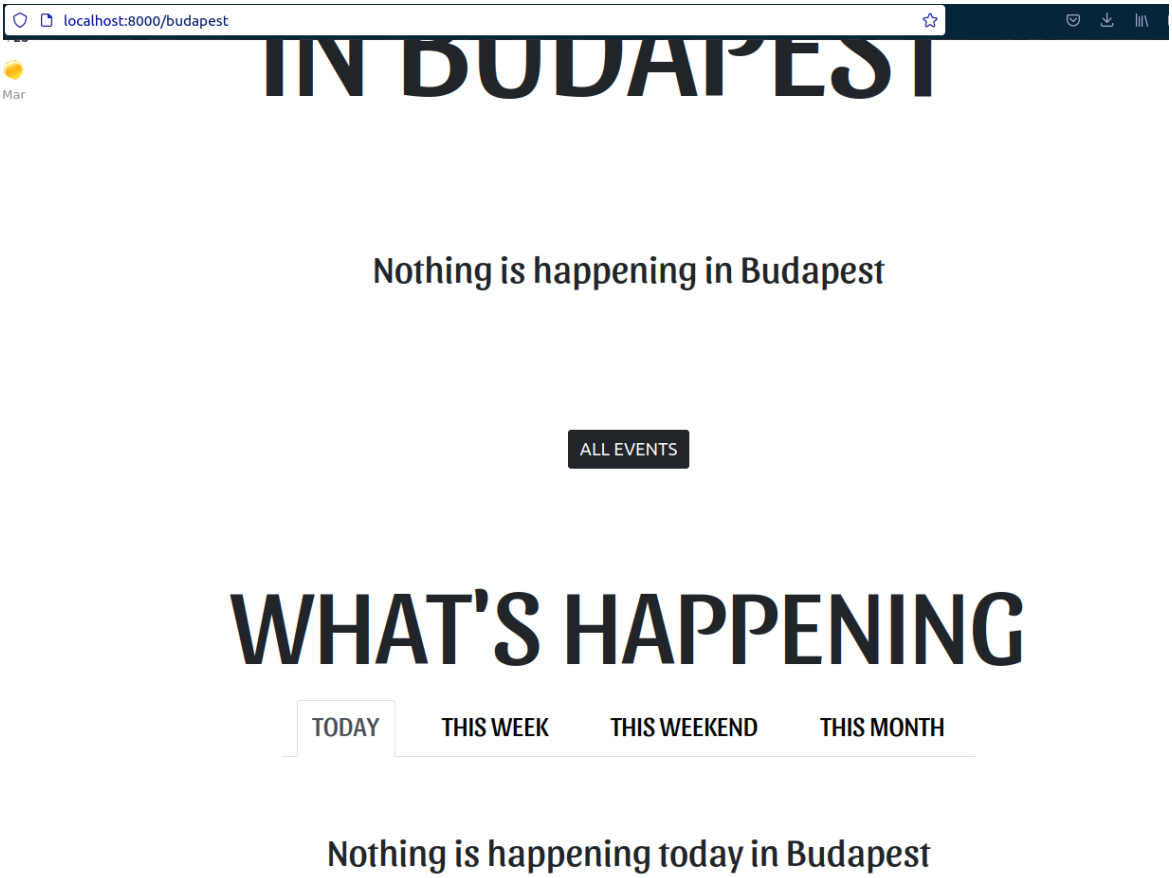
Descripción del caso: Contando con dos asociaciones ESN y ESP, desactivaremos la primera y mantendremos privada la segunda. Por lo tanto ningún evento debe aparecer en destino.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

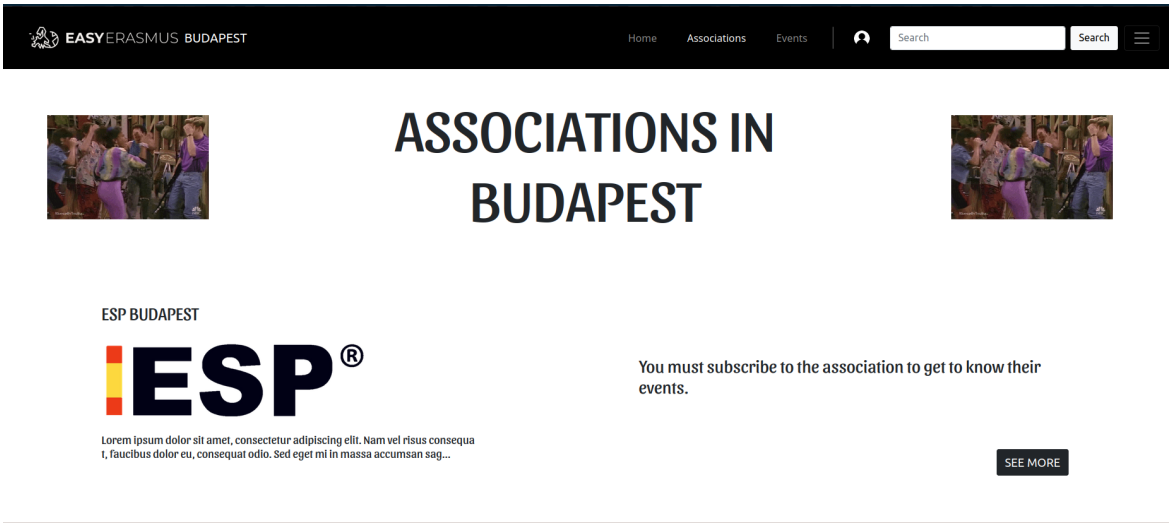
- **1º caso:** Datos de entrada: Página de inicio del destino. Resultado esperado (Salida): Los eventos próximos de las asociaciones no aparecen.
- **2º caso:** Datos de entrada: Página de asociaciones en el destino. Resultado esperado (Salida): La asociación ESP aparece pero sus eventos están ocultos.
- **3º caso:** Datos de entrada: Página de calendario de eventos en el destino. Resultado esperado (Salida): No hay eventos.
- **4º caso:** Datos de entrada: Búsqueda de eventos. Resultado esperado (Salida): Los eventos que coincidan con la búsqueda no aparecen.
- **5º caso:** Datos de entrada: Perfil de la asociación. Resultado esperado (Salida): Los eventos de la asociación no se muestran.
- **6º caso:** Datos de entrada: Acceso mediante URL a la asociación inactiva. Resultado esperado (Salida): Mensaje de asociación inactiva.
- **7º caso:** Datos de entrada: Acceso mediante URL a la evento privado. Resultado esperado (Salida): Mensaje de página no encontrada.

Mejoras conseguidas: El caso de prueba **6** redireccionó a una ruta equivocada (error 404). Por lo cual se ha corregido para mostrar el mensaje deseado.

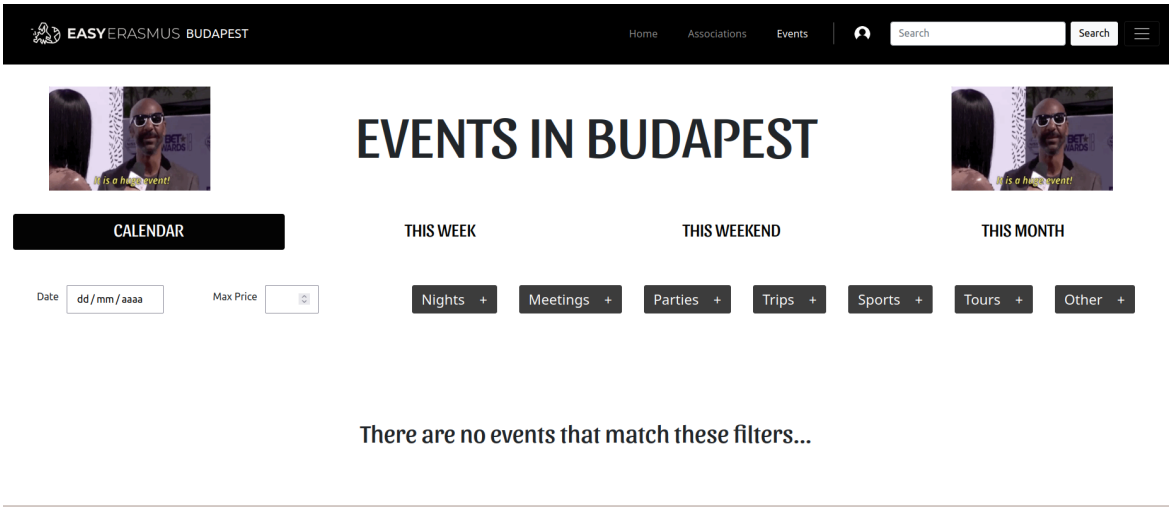
Prueba 12, Caso 1: Página de inicio de destino sin eventos.



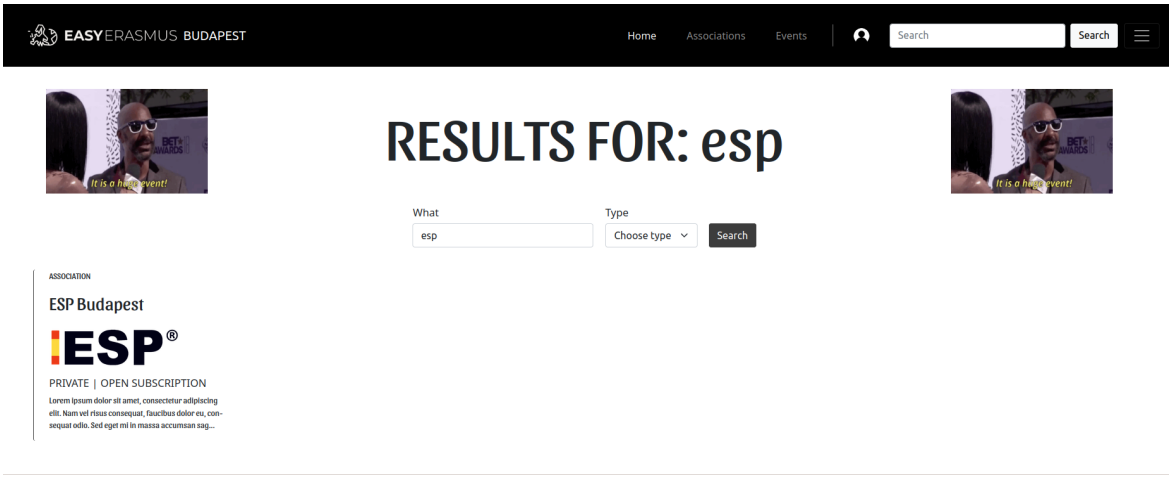
Prueba 12, Caso 2: Página de asociaciones sin mostrar eventos privados.



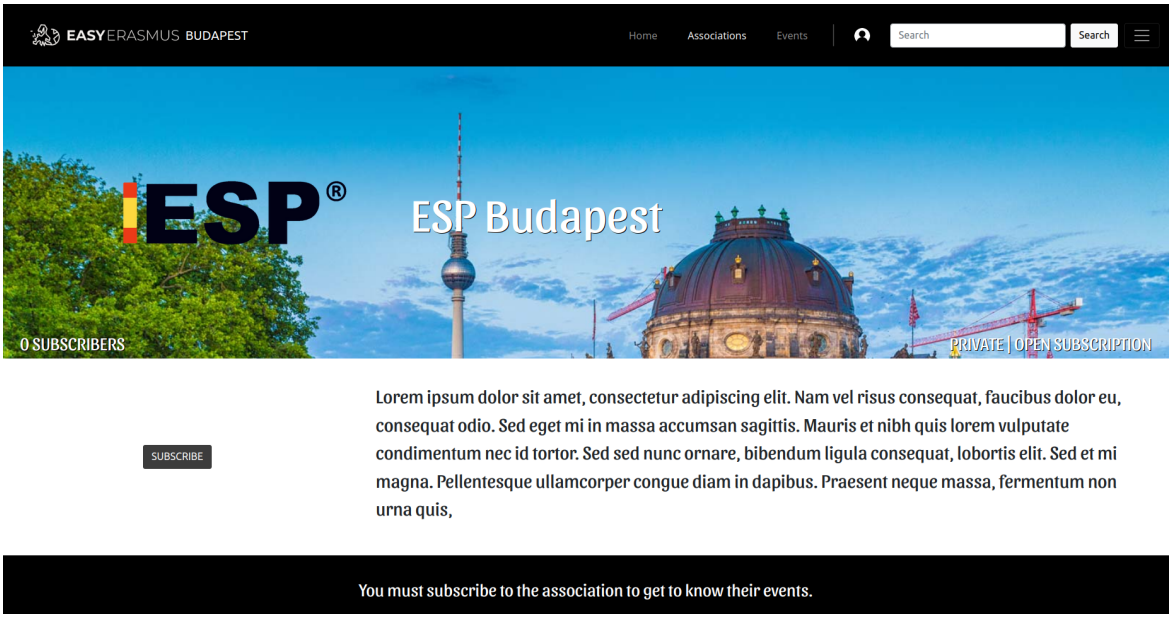
Prueba 12, Caso 3: Calendario de eventos vacío.



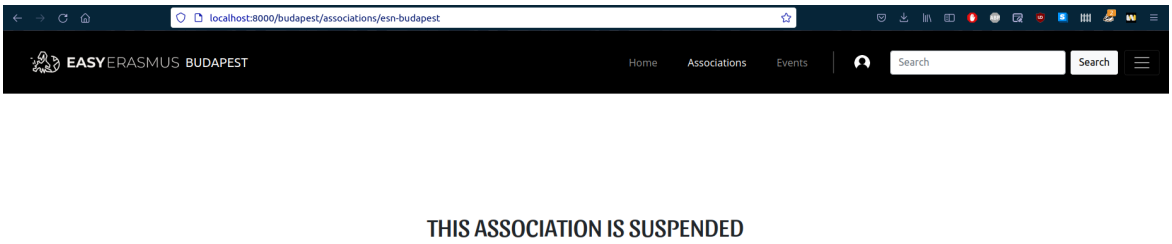
Prueba 12, Caso 4: Búsqueda de eventos privados sin resultado.



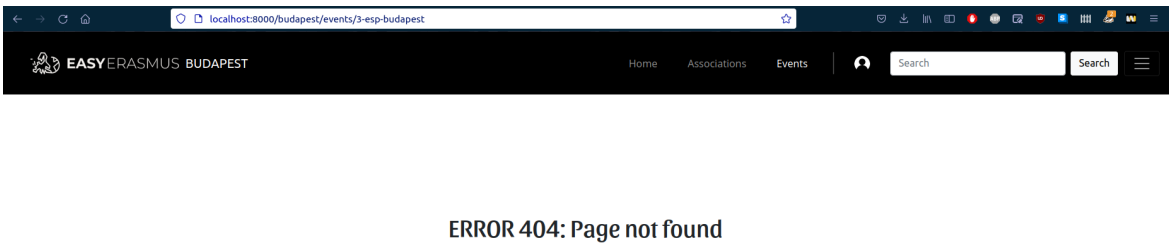
Prueba 12, Caso 5: Perfil de asociación privada sin mostrar eventos.



Prueba 12, Caso 6: Intento de acceso mediante URL a perfil de asociación inactiva.



Prueba 12, Caso 7: Intento de acceso mediante URL a evento privado.



Prueba 13: Inscripción a asociación abierta y cerrada

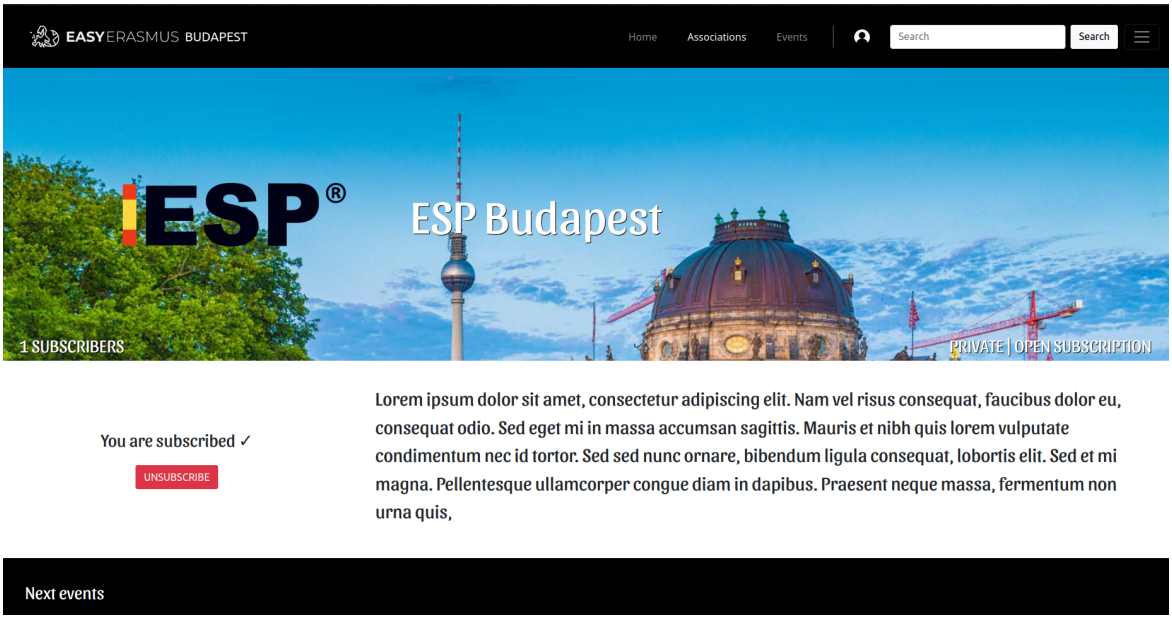
Descripción del caso: Contando con dos asociaciones ESP y ESN, la primera siendo de inscripción abierta y la segunda de inscripción por solicitud, el usuario solicitará unirse a

ambas.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

- **1º caso:** Datos de entrada: Solicitud de inscripción a ESP (abierta). Resultado esperado (Salida): Inscripción registrada instantáneamente.
- **2º caso:** Datos de entrada: Solicitud de inscripción a ESN (cerrada). Resultado esperado (Salida): La inscripción no se formaliza hasta ser aceptada por la asociación.

Prueba 13, Caso 1: Solicitud de inscripción a asociación abierta.



Prueba 13, Caso 2: Solicitud de inscripción a asociación cerrada.

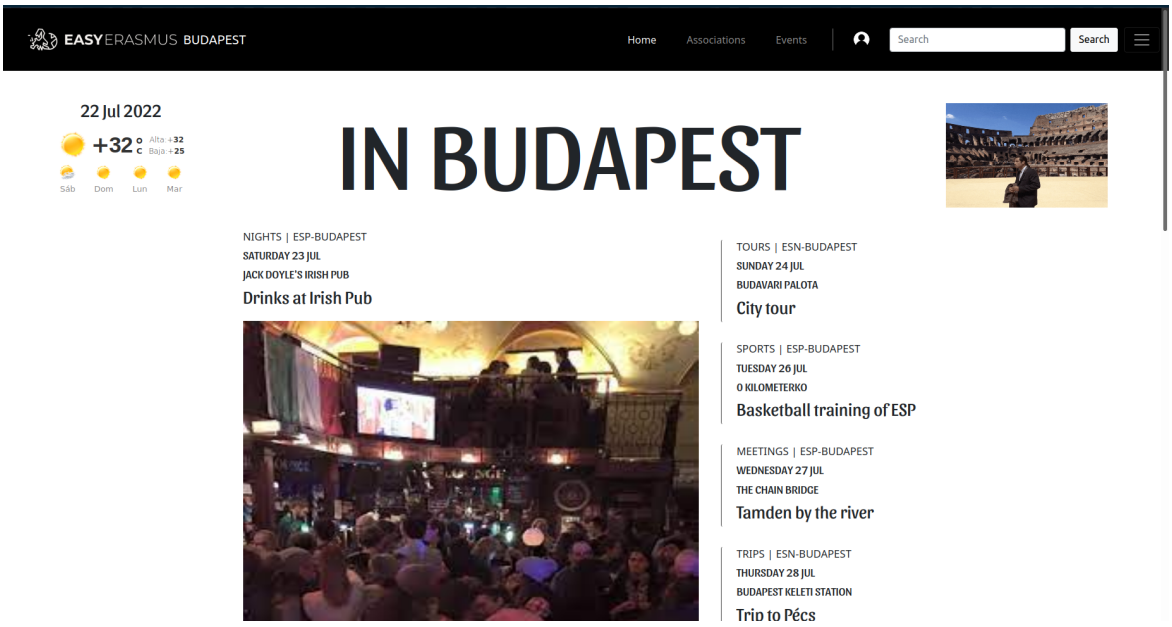
Prueba 14: Verificar que el estudiante puede visualizar eventos de asociaciones privadas si pertenece a ella

Descripción del caso: Contando con dos asociaciones ESN y ESP, la primera siendo pública y nos inscribiremos a la segunda (privada), por lo tanto todos los eventos deben aparecer en destino.

Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

- **1º caso:** Datos de entrada: Página de inicio de destino. Resultado esperado (Salida): Los eventos próximos de las asociaciones aparecen.
- **2º caso:** Datos de entrada: Página de asociaciones en el destino. Resultado esperado (Salida): La asociación ESP aparece pero y sus eventos están visibles.
- **3º caso:** Datos de entrada: Página de calendario de eventos en el destino. Resultado esperado (Salida): Se muestran todos los eventos.
- **4º caso:** Datos de entrada: Búsqueda de eventos. Resultado esperado (Salida): Los eventos que coincidan con la búsqueda aparecen.
- **5º caso:** Datos de entrada: Perfil de la asociación. Resultado esperado (Salida): Los eventos de la asociación se muestran.

Prueba 14, Caso 1: Página de inicio de destino.



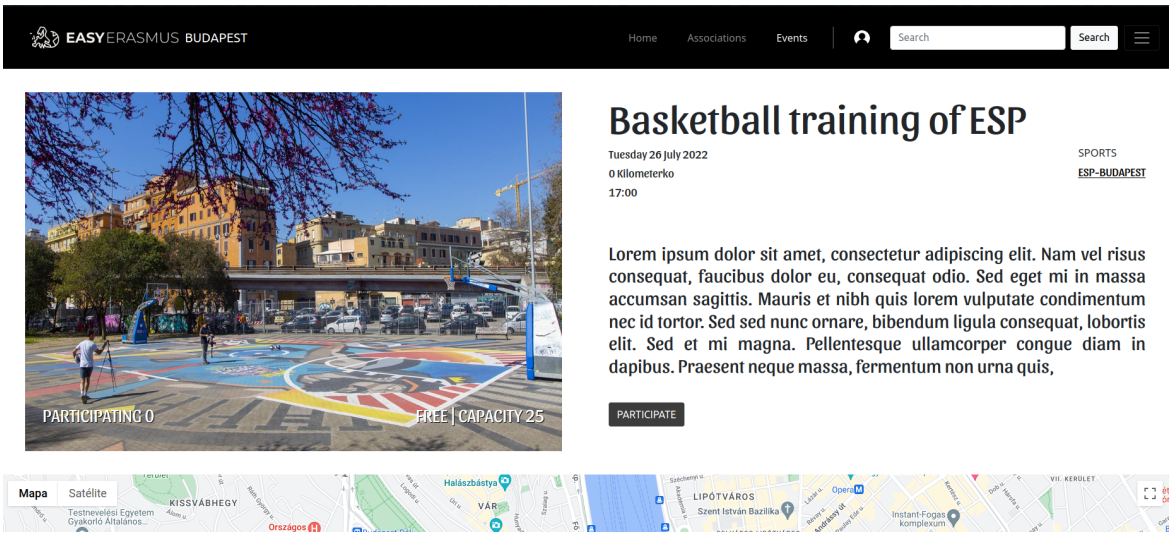
Prueba 15: Participación de eventos

Descripción del caso: Contando con una asociación en la que se participa y que celebra dos eventos, uno sin límite de participaciones y otro que ya ha superado el máximo, el usuario intenta participar en ambos

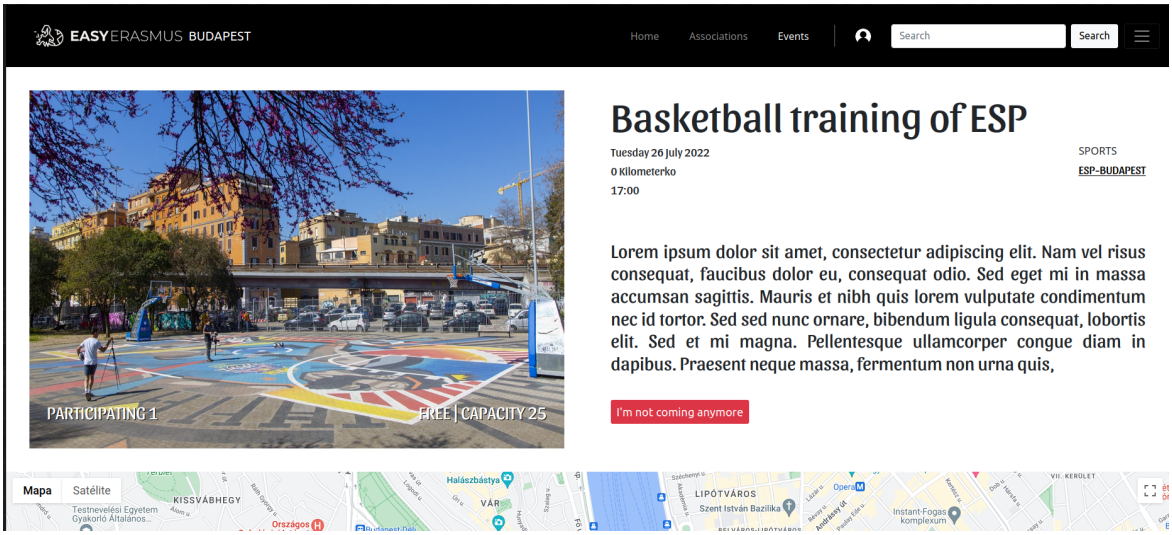
Técnica de prueba de caja negra: Requerimiento funcional / Caso de uso

- **1º caso:** Datos de entrada: Participación en evento sin límite. Resultado esperado (Salida): El botón de participación se muestra y esta se realiza al pulsarlo.
- **2º caso:** Datos de entrada: Participación en evento completo. Resultado esperado (Salida): El botón de participación no se muestra en el perfil del evento.

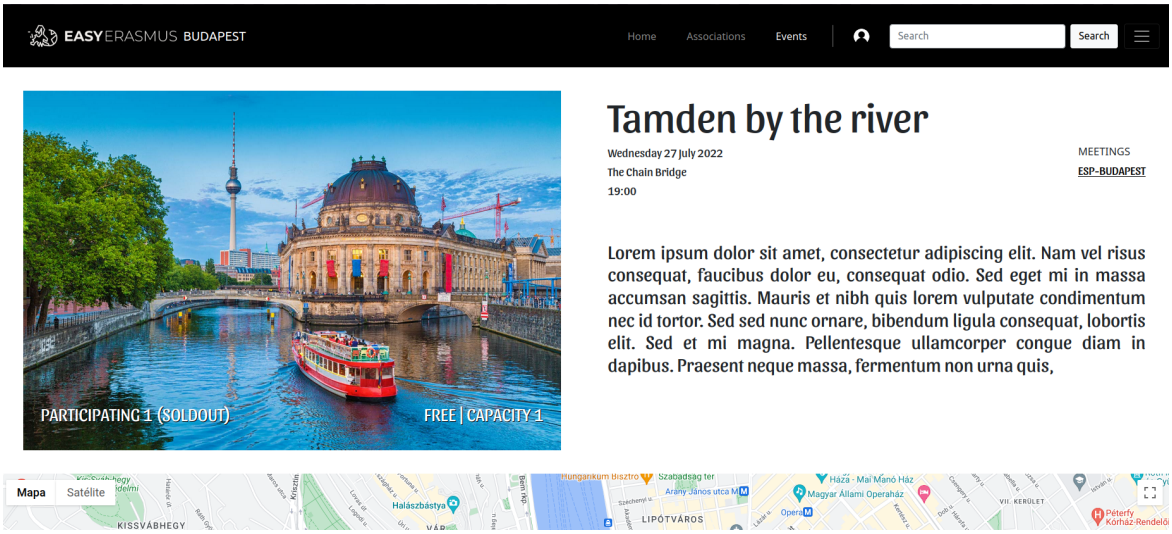
Prueba 15, Caso 1.1: Participación en evento sin límite.



Prueba 15, Caso 1.2: Participación satisfactoria.



Prueba 15, Caso 1.2: Imposibilidad de participación en evento completo.



6.2.5. Criterio responsive

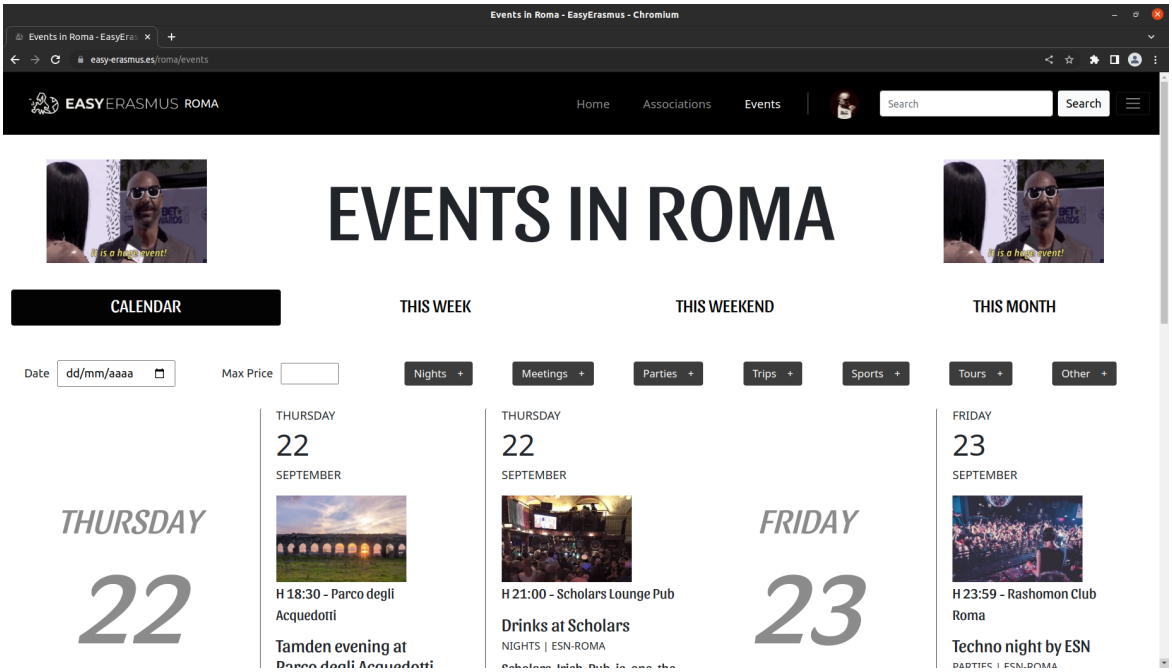
Prueba 16: Visualización en distintos navegadores web

Descripción del caso: Las diferentes vistas de la aplicación deben poder visualizarse con los navegadores web más usados, Google Chrome y Firefox.

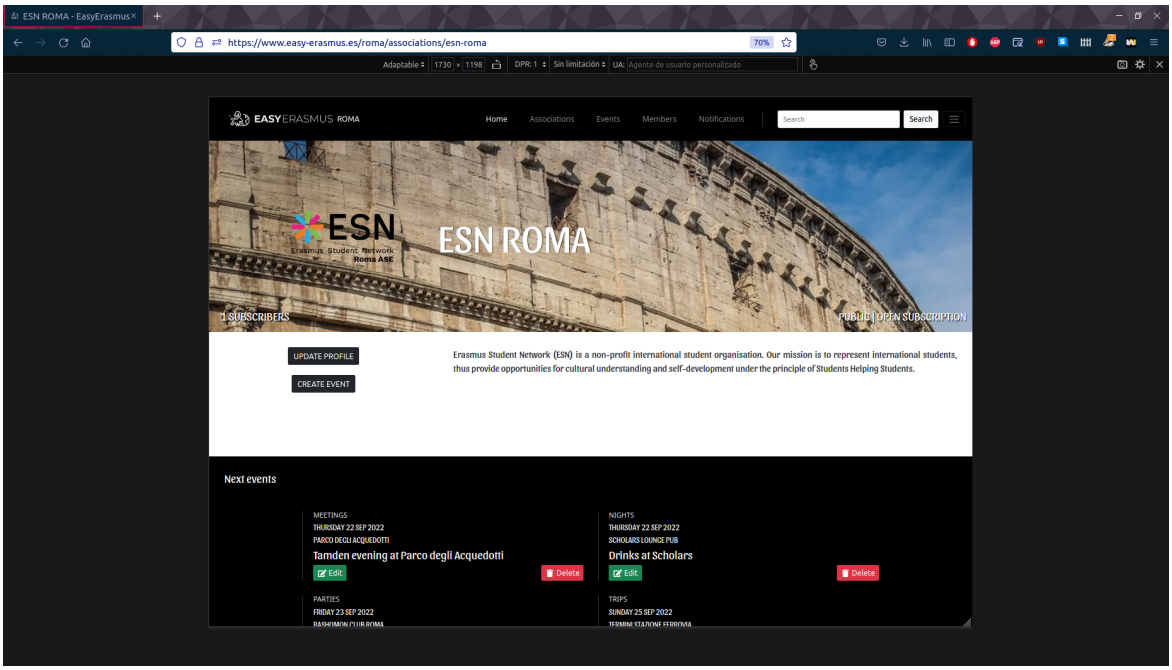
Técnica de prueba de caja negra: Requerimiento no funcional

- **1º caso:** Datos de entrada: Acceder a vistas desde el navegador Google Chrome (Salida): La pantalla se visualiza correctamente y el diseño es responsive a los cambios en el tamaño de pantalla y resolución.
- **2º caso:** Datos de entrada: Acceder a vistas desde el navegador Firefox (Salida): La pantalla se visualiza correctamente y el diseño es responsive a los cambios en el tamaño de pantalla y resolución.

Prueba 16, Caso 1: Calendario de eventos visto desde Google Chrome



Prueba 16, Caso 2: Perfil de asociación visto desde Firefox (monitor reducido)



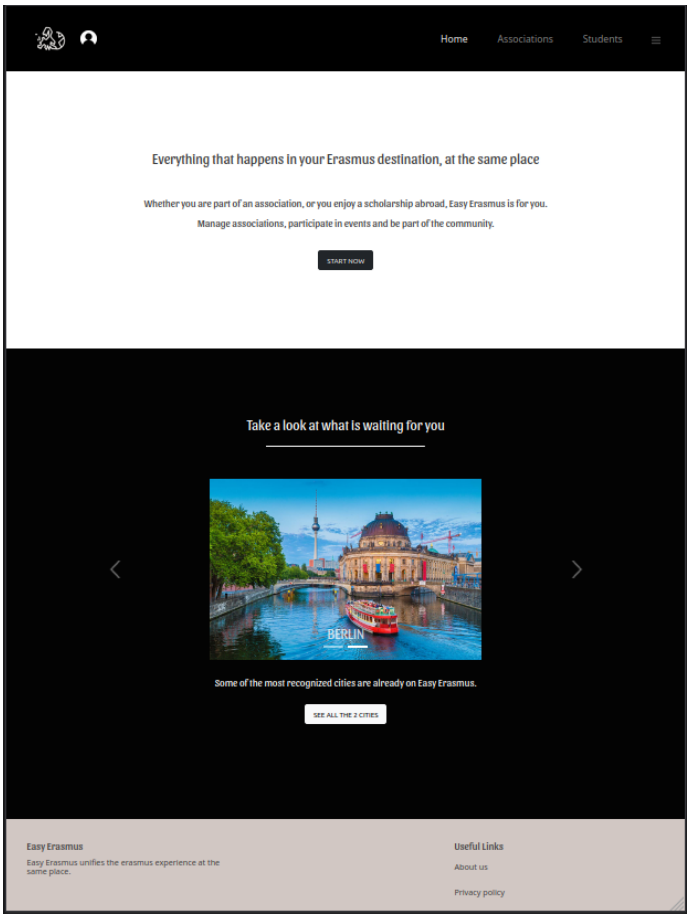
Prueba 17: Visualización en distintos dispositivos móviles

Descripción del caso: Las diferentes vistas de la aplicación deben poder visualizarse en diversos dispositivos móviles con distinto tamaño de pantalla.

Técnica de prueba de caja negra: Requerimiento no funcional. Utilización de **Site-Shot** (Site-Shot, s.f.).

- **1º caso:** Datos de entrada: Acceder a vista inicial desde un iPad Pro 12.9 (1024x1366) (Salida): La pantalla se visualiza correctamente y el diseño se adapta al cambio de resolución.
- **2º caso:** Datos de entrada: Acceder a vista inicial de usuario desde un Galaxy Note 20 Ultra (412x883) (Salida): La pantalla se visualiza correctamente y el diseño se adapta al cambio de resolución.
- **3º caso:** Datos de entrada: Acceder a calendario de eventos desde un iPhone X/XS (375x812) (Salida): La pantalla se visualiza correctamente y el diseño se adapta al cambio de resolución.
- **4º caso:** Datos de entrada: Acceder a el índice de asociaciones desde una tablet Nexus 10 (800x1280) (Salida): La pantalla se visualiza correctamente y el diseño se adapta al cambio de resolución.

Prueba 17, Caso 1: Landing Page vista desde un iPad Pro 12.9 (1024x1366)



Prueba 17, Caso 2: Vista inicial de estudiante desde Galaxy Note 20 Ultra (412x883)

Prueba 17, Caso 3: Calendario de eventos visto desde iPhone X/XS (375x812)

Prueba 17, Caso 4: Índice de asociaciones en destino visto desde tablet Nexus 10 (800x1280)

6.3. Conclusiones tras las pruebas

Con la realización de pruebas software se han obtenido varios beneficios por el poco tiempo que conlleva aplicarlas. Nos han obligado a ganar seguridad y reducir el código mediante separación de funciones y reutilización. Asimismo se ha ganado más robustez en el código fuente por los mismos tests, y legibilidad, ya que nuestro código es más fácil de entender.

Además, en las últimas pruebas, se ha ganado muchísima claridad en la interfaz, con la mejora en la adaptación de diversos tamaños de pantalla. Con esto se ha conseguido claridad y mejor usabilidad de cara al usuario.

Capítulo 7

Despliegue

La última fase de este proyecto consistirá en el despliegue de la aplicación desarrollada de cara al público.

7.1. Hosting seleccionado

LucusHost ha sido el hosting seleccionado, pues además de ser gratuito durante 3 meses, nos otorga acceso a CPanel, PHPMyAdmin y conexión FTP, factores esenciales para el despliegue rápido de los datos. Además, contamos con certificado SSL para todas nuestras páginas por la duración del periodo gratis. (LucusHost, s.f.)

7.2. Subida de la aplicación

Para montar el sistema en línea, se ha comprimido y enviado el código mediante FTP. En la carpeta pública solo se encontrará el código CSS y la página índice, que apuntan al resto del proyecto. Una vez subida, ha sido necesario verificar la versión PHP del hosting, para que concuerde con aquella de Laravel, y activar los certificados SSL. Tras esto solo era necesario migrar la base de datos, para la generación de las tablas del sistema.

7.3. Link de acceso a la aplicación y código

Tras la realización de los pasos anteriores, la aplicación se encuentra disponible y 100 % operativa en ***www.easy-erasmus.es***

Asimismo, el proyecto está disponible publicamente en mi cuenta de GitHub.

Capítulo 8

Conclusiones, disciplinas adquiridas y líneas futuras

8.1. Conclusiones

En la finalización del proyecto se podría decir que el resultado obtenido ha sido satisfactorio, puesto que la totalidad de requisitos diseñados han sido cubiertos. Todo ello ha llevado la puesta en práctica de buena parte de conocimientos adquiridos en el Grado de Ingeniería Informática, además de forzar el aprendizaje de otros muchos.

Podemos decir que al finalizar el proyecto se han alcanzado los objetivos de alto nivel planteados.

- Se han conseguido los objetivos de gestión de eventos y asociaciones.
- Se ha construido una aplicación a medida que permite a los usuarios gestionar sus perfiles, sus participaciones en eventos o inscripciones a asociaciones.
- La aplicación permite al administrador gestionar de forma sencilla las asociaciones en destino.
- Se ha conseguido un nivel de seguridad mínimo a la hora de validar las acciones de los usuarios e identificarles en el sistema.
- La aplicación ha sido validada con satisfacción por el usuario final.

Por otra parte, la temporalización del proyecto se ha visto alargada con su evolución:
La fase de **Análisis** ha tomado realmente un mes más, pues ha sido muy detallada y corregida usualmente, para plantar unos buenos cimientos para el futuro del trabajo. A su vez, ha requerido la transcripción a *LaTeX*, al que ha sido necesario sumarle el tiempo de aprendizaje del software de formateo.
Seguidamente el periodo de **Diseño** se ha ampliado tan solo dos semanas más, en su mayoría debido al debate en el prototipado de la aplicación final y el diseño visual ideado para el sistema.
Posteriormente, la fase de **Implementación** ha tomado un mes más, después de determinar las herramientas a utilizar y haber sido necesario profundizar y ejercitarse en el uso del framework de desarrollo web *Laravel*.
Sin embargo la fase de **Pruebas** ha tomado menos tiempo del pensado, pues el mismo framework *Laravel* implementa herramientas de testing que han facilitado el proceso en gran

parte.
A todo ello hay que sumarle el hecho de haber realizado el proyecto en mi año de intercambio, que presenta más actividad social de la usual, además de la adaptación a las clases en destino.
En definitiva el proyecto se ha alargado un mes y medio más, tiempo que ha sido aprovechado a su vez para mejorar y perfeccionar el software.

Cronología	Octubre				Noviembre				Diciembre				Enero				Febrero				Marzo			
Semana	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Análisis																								
Diseño																								
Implementación																								
Pruebas																								
Despliegue																								
Documentación																								

Tabla 8.1: Temporalización real del proyecto

Cronología	Abril				Mayo				Junio				Julio				Agosto			
Semana	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Análisis																				
Diseño																				
Implementación																				
Pruebas																				
Despliegue																				
Documentación																				

Tabla 8.2: Temporalización real del proyecto. Continuación.

Ha sido una experiencia desafiante realizar este proyecto desde el extranjero, sumándole las responsabilidades que ello conlleva. Sin embargo, gracias a la muy buena comunicación y dedicación de mi tutor de manera telemática, ha sido más sencillo evolucionar el trabajo y llevarlo a buen puerto. Además el interés por el tema escogido, siempre ha sido un plus para dedicarle tiempo y ocupación por mi parte.

8.2. Disciplinas adquiridas

El hecho de haberme sumergido en el mayor proyecto de desarrollo software que he ejecutado hasta la fecha ha supuesto que me esfuerce al límite. Asimismo, tras haberlo realizado de una manera individual ha supuesto mi notoria mejora personal en la resolución de problemas, investigación y documentación. Ni que decir tiene la soltura ganada en dos de los frameworks más usados en el desarrollo software, Laravel y Bootstrap, que cuentan con uno de los públicos más grandes en el mundo del desarrollo web y permiten su utilización en proyectos de cualquier escala.

8.3. Líneas futuras

El sistema desarrollado podría ser ampliado de varias formas. Personalmente las siguientes me parecen apropiadas:

- Doble verificación o confirmación de email en el registro de usuarios.
- Notificación vía email a los usuarios del sistema, cuando se produzcan cambios importantes, p.e.: inscripción a una asociación, desactivación de una asociación a la que se pertenecía o actualización de eventos en los que se participa.
- Laravel permite crear y gestionar APIs de forma sencilla para el acceso a los recursos. Sería muy conveniente de cara al usuario que se definiera una API tal que permitiese desarrollar una aplicación móvil que posibilitara el acceso al sistema de una manera más cómoda desde dispositivos móviles.

Apéndice A

Anexo I: Manual de Usuario

A.1. Introducción

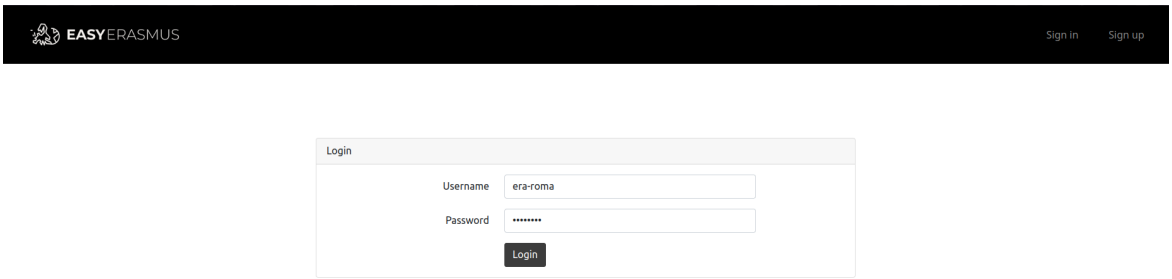
El presente manual le permitirá aprender a utilizar todas las funcionalidades básicas de EasyErasmus.
EasyErasmus es una herramienta para la gestión de actividades y asociaciones Erasmus, con la posibilidad de creación de eventos, asociaciones y la participación de estudiantes en ellas. Cada rol tendrá un determinado nivel de acceso a las diversas funciones del software.

A.2. Instalación

Al encontrarnos ante una aplicación web, los usuarios podrán acceder a ella mediante cualquier navegador web, como pueden ser Mozilla Firefox o Google Chrome.

A.3. Acceso

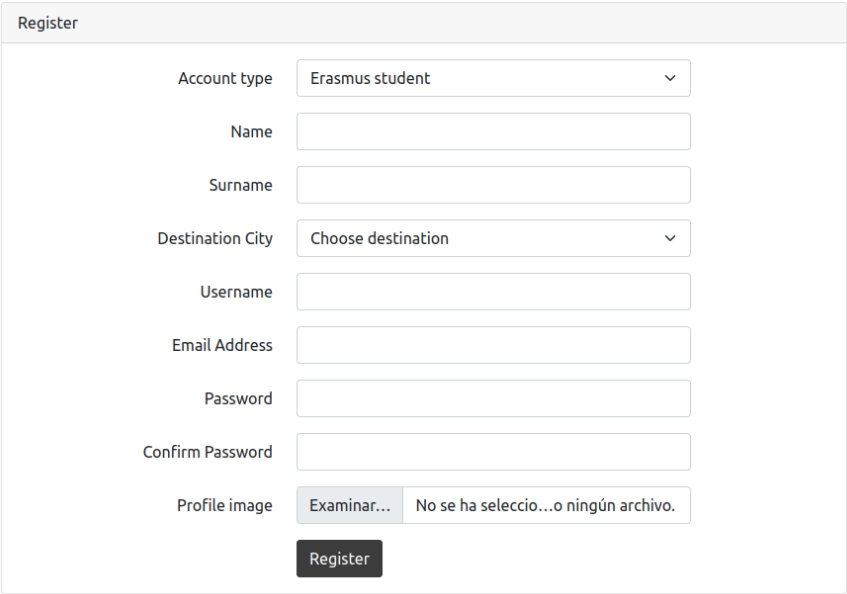
Iniciar Sesión: Acceda a la URL de la aplicación y rellene el formulario con sus credenciales.



Vista 1: Inicio de sesión.

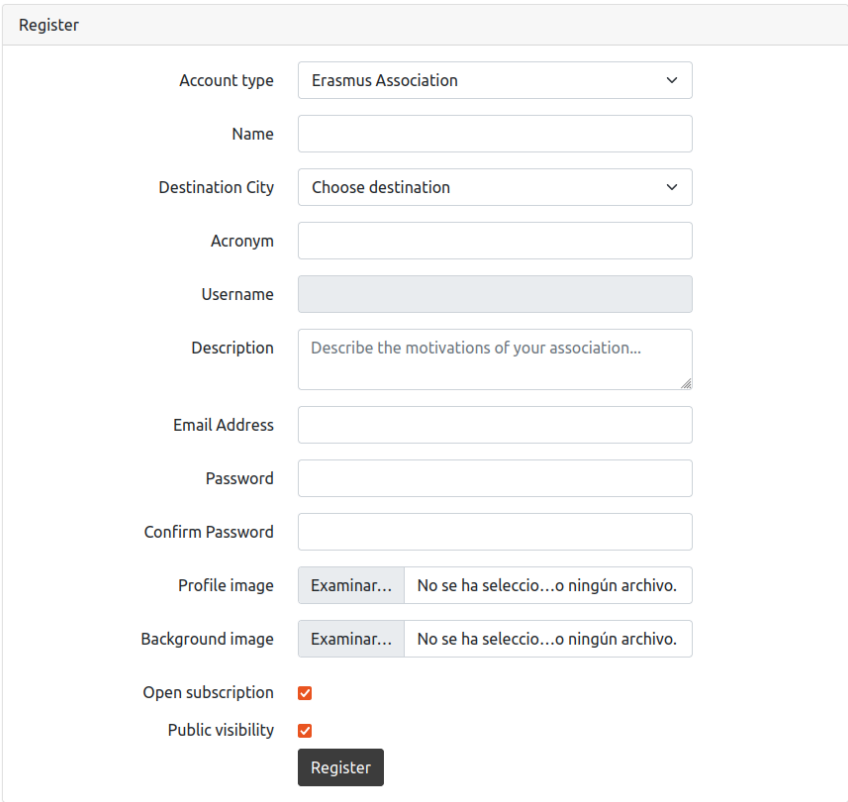
Registro: Acceda a la URL de registro y rellene el formulario con su información.

Registro de estudiantes.



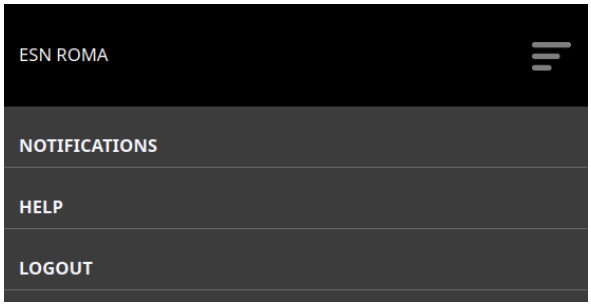
Vista 2: Registro de estudiantes.

Registro de asociaciones.



Vista 3: Registro de asociaciones.

Cerrar sesión: Pulse en el icono de barra lateral derecha y se abrirá el desplegable. A continuación pulsar en 'LOGOUT' para cerrar sesión.

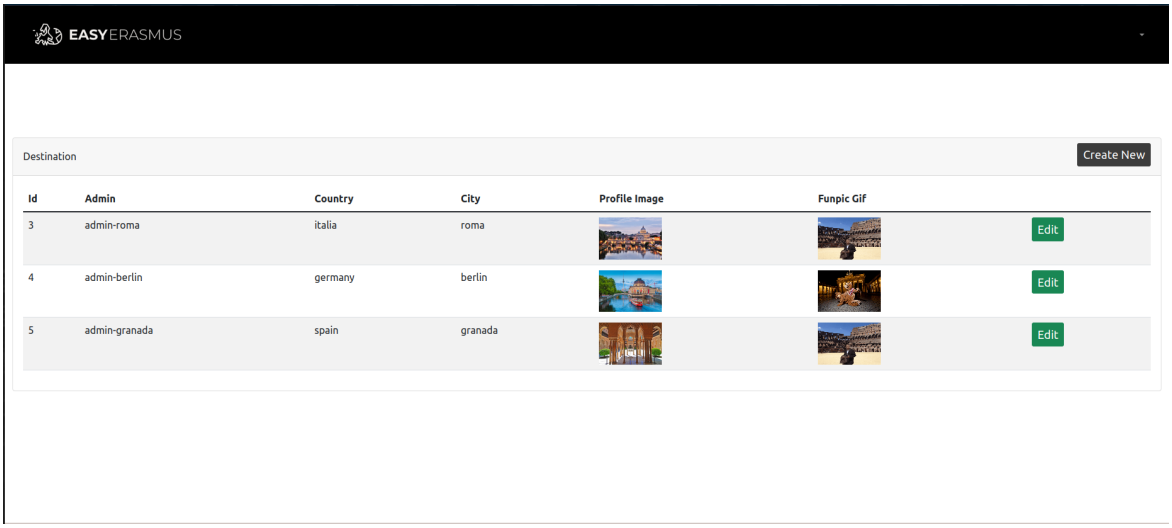


Vista 4: Cierre de sesión.

A.4. Utilización

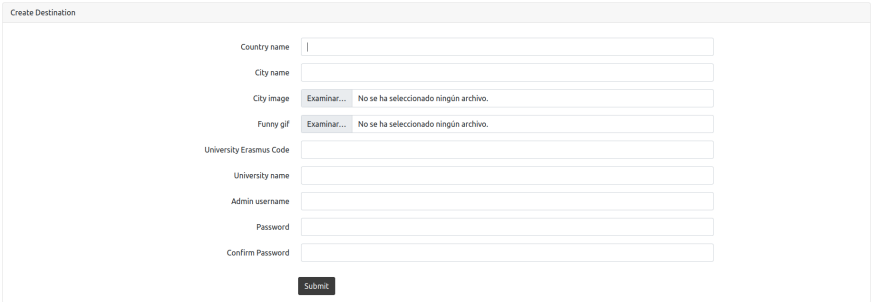
A.4.1. Administrador de destinos

Al iniciar sesión como este administrador, se mostrará un índice con todas las ciudades en las que está disponible la aplicación al momento, con la opción de añadir nuevas o editar las existentes.



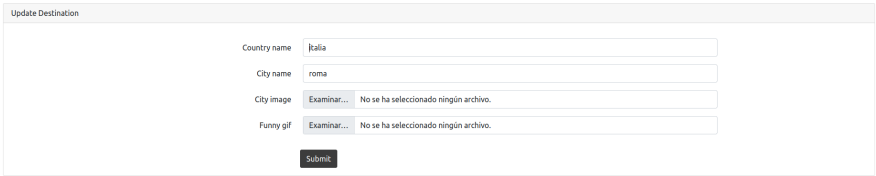
Vista 5: Índice de destinos de la aplicación.

Creación de destinos: En el índice de destinos, pulse en 'CREATE NEW'. Esto le redirigirá al formulario de creación de destinos. Rellenos todos los campos pulse 'Submit'.



Vista 6: Creación de nuevo destino.

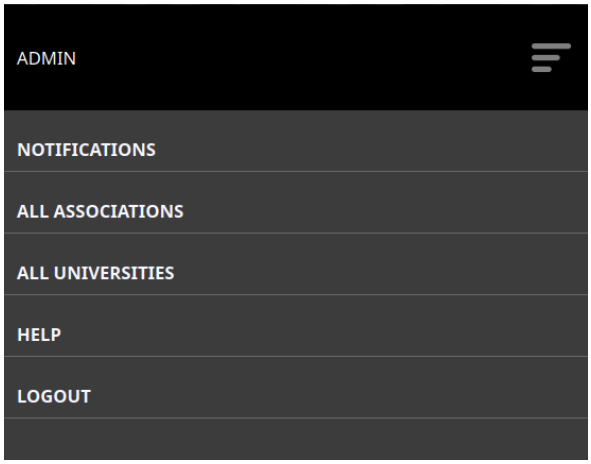
Modificación de destinos: En el índice de destinos, pulse en 'Edit' en la ciudad a modificar. Esto le redirigirá al formulario de modificación de destinos. Modificados los campos pulse 'Submit'.



Vista 7: Modificación de destino.

A.4.2. Administrador por destino

Al iniciar sesión como este administrador, se mostrará una vista de usuario estudiante con todas las asociaciones y eventos disponibles para su visionado (sin importar la privacidad de estos). Además de tener el control de asociaciones en el destino, notificaciones y universidades, desde la barra de navegación.



Vista 8: Barra de navegación de administrador.

Aceptar nueva asociación en destino: En el índice de asociaciones, pulse en 'Registrate' en la nueva asociación. Esto cambiará su estado a activa y le permitirá trabajar desde la

aplicación.

All associations

Cod Association	Name	Profile image	Email	Inscription type	Privacy	State	
ESP-ROMA	ESP ROMA		esp_roma@gmail.com	OPEN SUBSCRIPTION	PUBLIC	SUSPENDED	Notify Registrate
ERA-ROMA	ERA ROMA		era-roma@gmail.com	SUBSCRIPTION BY REQUEST	PRIVATE	WORKING	Notify
ESN-ROMA	ESN ROMA		esn-roma@gmail.com	OPEN SUBSCRIPTION	PUBLIC	WORKING	Notify

Vista 9: Índice de asociaciones en destino.

Notificar asociación: En el índice de asociaciones, pulse en 'Notify'. Esto le llevará al formulario de notificación para hacerle saber a la asociación de los posibles fallos o problemas que tenga. Si la desactivación es necesaria mientras se soluciona, marque la casilla 'Deactivation required'. Pulse 'Submit' para enviar la notificación.

Create Notification

Cod Association: era-roma

Description of changes: Describe the changes to be done by the association...

Deactivation required: ☐

Submit

Vista 10: Creación de notificación a asociación.

Responder notificación de desactivación: En el índice de notificaciones, pulse 'Resend' si el problema no ha sido resuelto y desea reenviar la notificación. Pulse 'Accept' para borrar la notificación y finalizar el proceso.

Notification

Id	CodAssociation	Description	State	Deactivate association	Created	Updated	
6	ERA-ROMA	The event 'Saltimbocca pizza in boca' is offensive, you shall modify the title.	ANSWERED	YES	10 Aug 2022 16:33	10 Aug 2022 16:34	Accept Resend
7	ESP-ROMA	Your association has been accepted in our application. You can now post events!	SEND	NO	10 Aug 2022 16:34	10 Aug 2022 16:34	

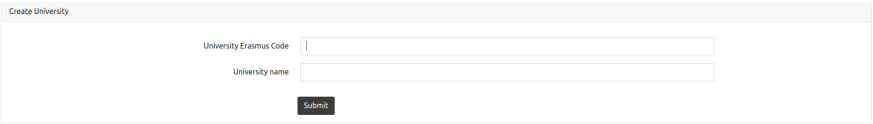
Vista 11: Índice de notificaciones del administrador.

Crear nueva universidad en destino: En el índice de universidades, pulse 'Create new' para acceder al formulario de creación. Pulse 'Submit' tras rellenar los datos para almacenar la universidad.

University — Create New

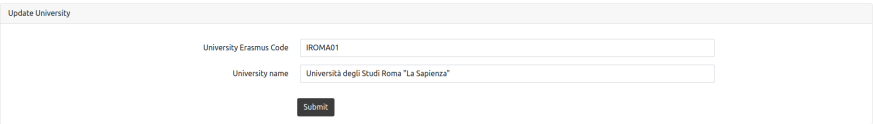
Erasmus Code	University Name	
IROMA01	Università degli Studi Roma "La Sapienza"	Edit
IROMA03	Universitá Roma Tre	Edit

Vista 12: Índice de universidades.



Vista 13: Creación de universidad.

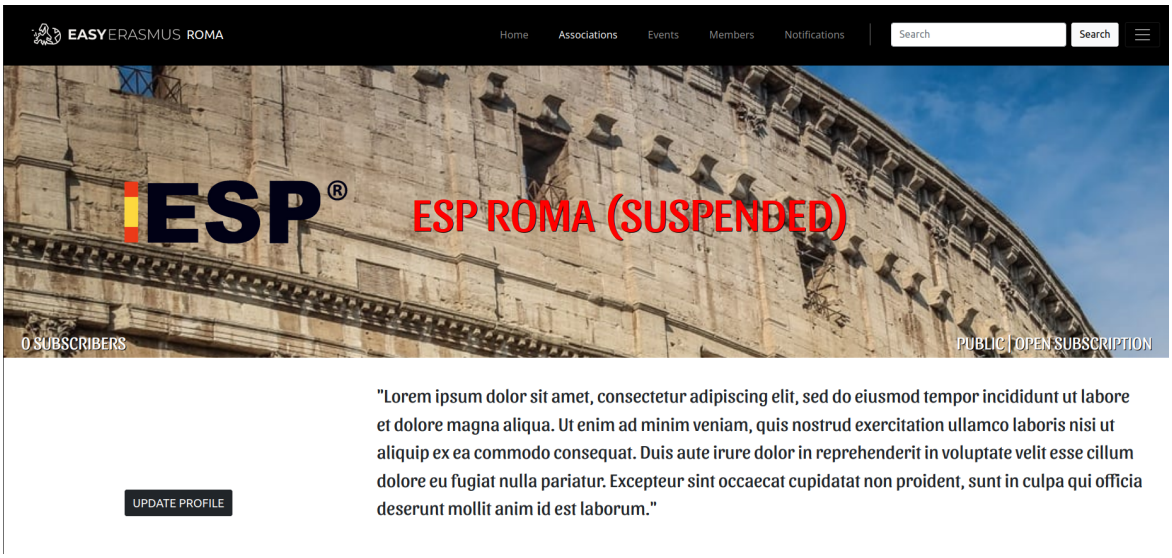
Editar universidad en destino: En el índice de universidades, pulse 'Edit' para acceder al formulario de modificación. Pulse 'Submit' tras actualizar los datos.



Vista 14: Modificación de universidad.

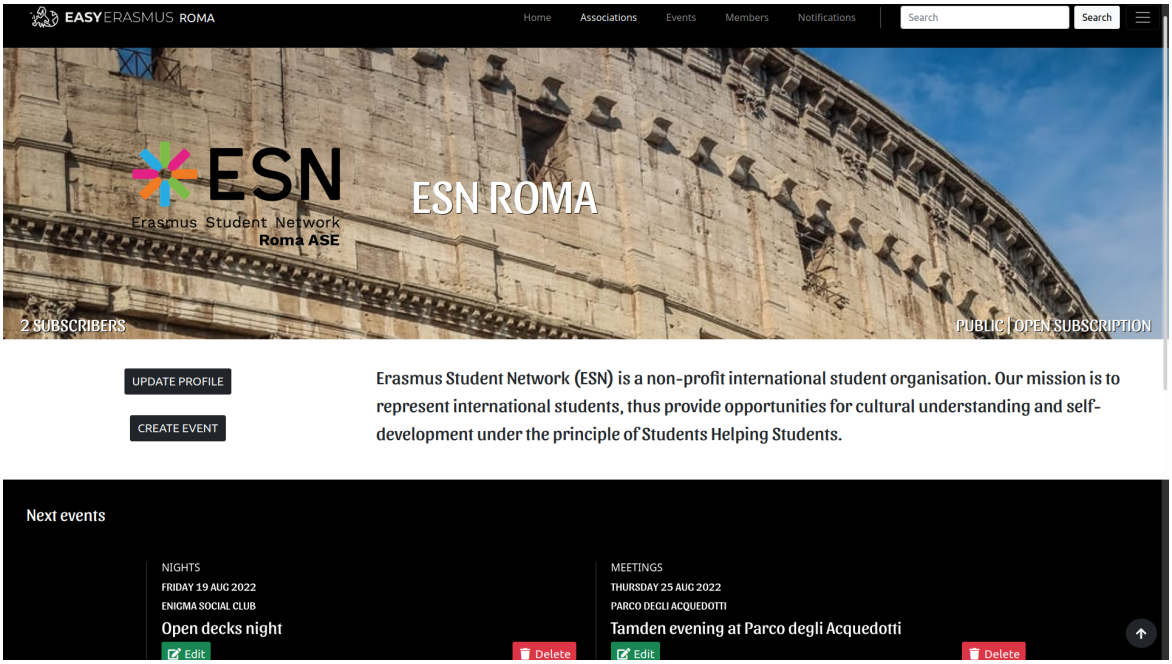
A.4.3. Asociación Erasmus

Al iniciar sesión como esta, se mostrará el perfil de la asociación. Si ha sido recién creada, no se permitirá su uso hasta ser aceptada por el administrador, manteniéndose inactiva.



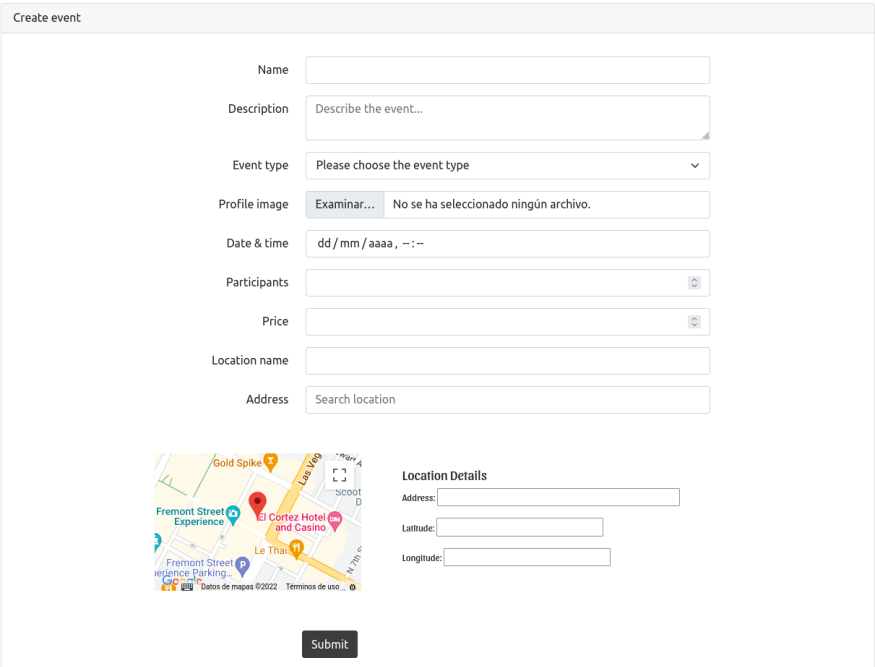
Vista 15: Asociación nueva.

Una vez haya sido activada, la creación de eventos estará permitida. Desde su perfil aparecerán sus eventos con la opción de editarlos o eliminarlos. Además dispondrá de un índice de eventos, miembros y notificaciones en la barra superior.



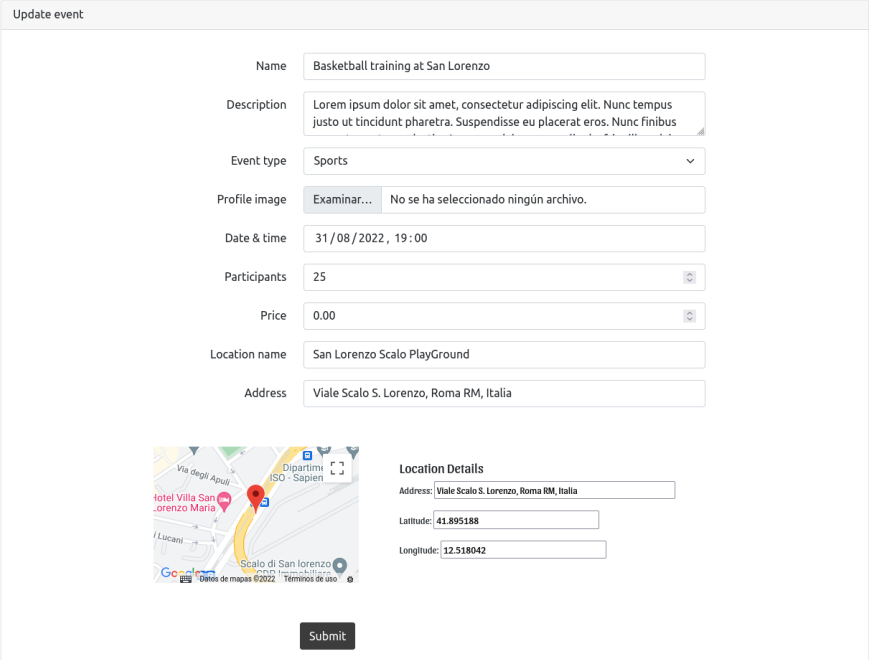
Vista 16: Asociación activa.

Crear evento: Desde el perfil de asociación o el índice de eventos, pulse en 'Create event' para acceder al formulario de creación. Pulse 'Submit' tras rellenar los datos.



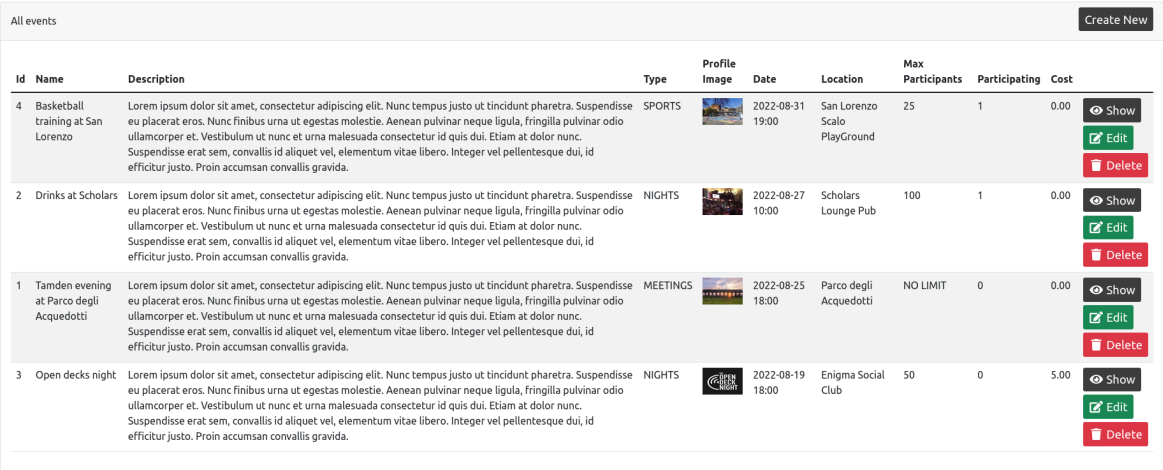
Vista 17: Creación de nuevo evento.

Modificar evento: Desde el índice de eventos o el perfil del evento en cuestión, pulse en 'Edit' para acceder al formulario de modificación. Pulse 'Submit' tras actualizar los datos.



Vista 18: Modificación de evento.

Eliminar evento: Desde el índice de eventos o el perfil del evento en cuestión, pulse en 'Delete' para eliminar del sistema el evento seleccionado.



Vista 19: Eliminación de evento.

Aceptar notificación: Desde el índice de notificaciones, pulse en 'Resolve' para actualizar el estado de la notificación. Si se ha recibido una notificación que requería de desactivación, el estado de la asociación se mantendrá inactivo hasta la revisión por parte de un administrador.

Notification

Id	Description	State	Deactivate association	Created	Updated	
6	The event 'Saltimbocca pizza in boca' is offensive, you shall modify the title.	SEND	YES	10 Aug 2022 16:33	10 Aug 2022 16:33	Resolve

Vista 20: Resolución de notificación.

Aceptar nuevo miembro: Si la solicitud de inscripción de usuarios está activada, el administrador de la asociación deberá aceptar a cada uno de los usuarios que solicite inscribirse. Para ello, desde el índice de miembros, pulse 'Accept' en el usuario seleccionado.

Student Members

Id	Name	Surname	Username	Profile Image	Date Inscription	State	
5	Test	Test	test	test	2022-08-10 18:36:48	PENDING	Accept Show Delete

Vista 21: Índice de miembros de la asociación.

Eliminar miembro: Desde el índice de miembros, pulse 'Delete' en el usuario seleccionado para eliminar su inscripción a la asociación.

Student Members

Id	Name	Surname	Username	Profile Image	Date Inscription	State	
1	Test	Test	test	test	2022-07-21 16:29:35	ACCEPTED	Show Delete
2	Cristian	Fernandez	cristianfj_		2022-07-21 23:04:52	ACCEPTED	Show Delete

Vista 22: Índice de miembros aceptados de la asociación.

Modificar asociación: Desde el perfil de asociación, pulse 'Update profile' para acceder al formulario de modificación. Pulse 'Update' tras actualizar los datos.

Edit association profile

Association name: ESN ROMA

Description: Erasmus Student Network (ESN) is a non-profit international student organisation. Our mission is

Profile image: Examinar... No se ha seleccio...o ningún archivo.

Background image: Examinar... No se ha seleccio...o ningún archivo.

Open subscription ☑

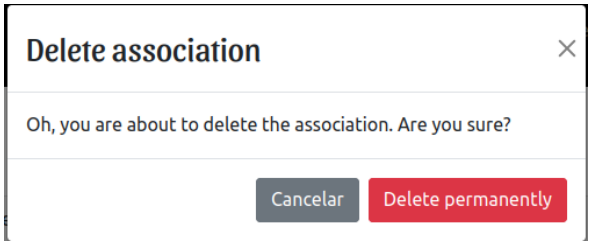
Public visibility ☑

Update

Delete association

Vista 23: Modificación de la asociación.

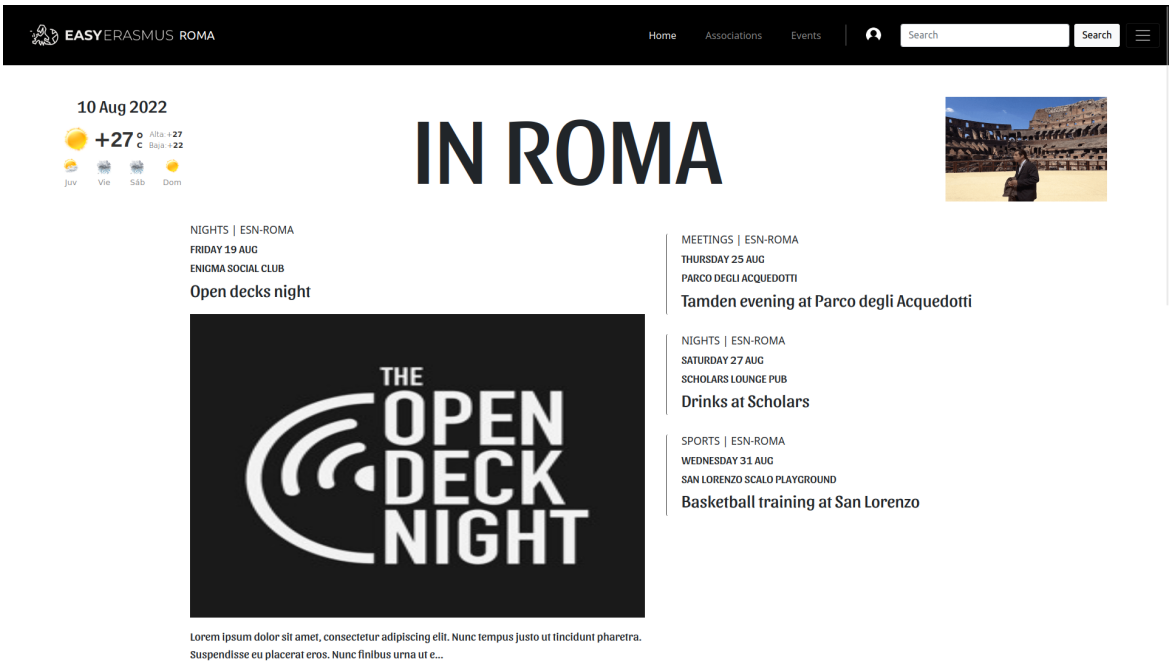
Dar de baja asociación: Desde el perfil de asociación, pulse 'Update profile' para acceder al formulario de modificación. Una vez dentro pulse en 'Delete association' y a continuación 'Delete permanently' en el pop-up emergente.



Vista 24: Baja de la asociación.

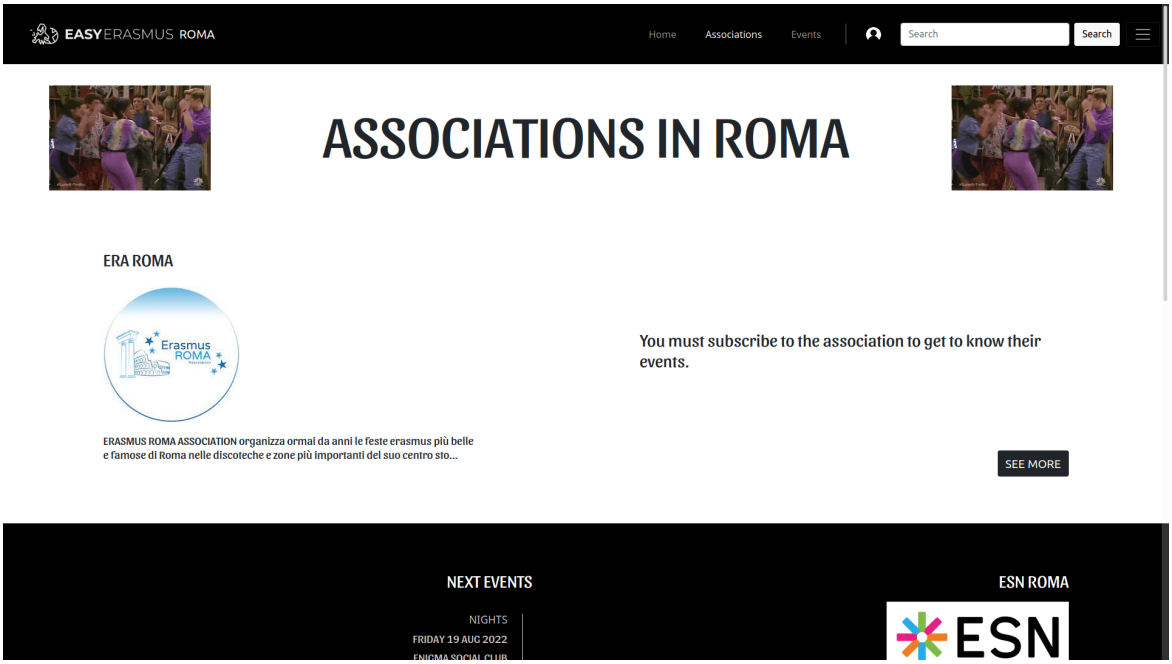
A.4.4. Estudiante Erasmus

Tras iniciar sesión como estudiante, será redirigido a la página de inicio del destino, donde aparecerán los próximos eventos ocurriendo en la ciudad.



Vista 25: Página de inicio en destino.

Visualizar asociaciones en destino: Pulse en el enlace 'Associations' de la barra de navegación para acceder al listado de asociaciones activas en su destino.



Vista 26: Listado de asociaciones en destino.

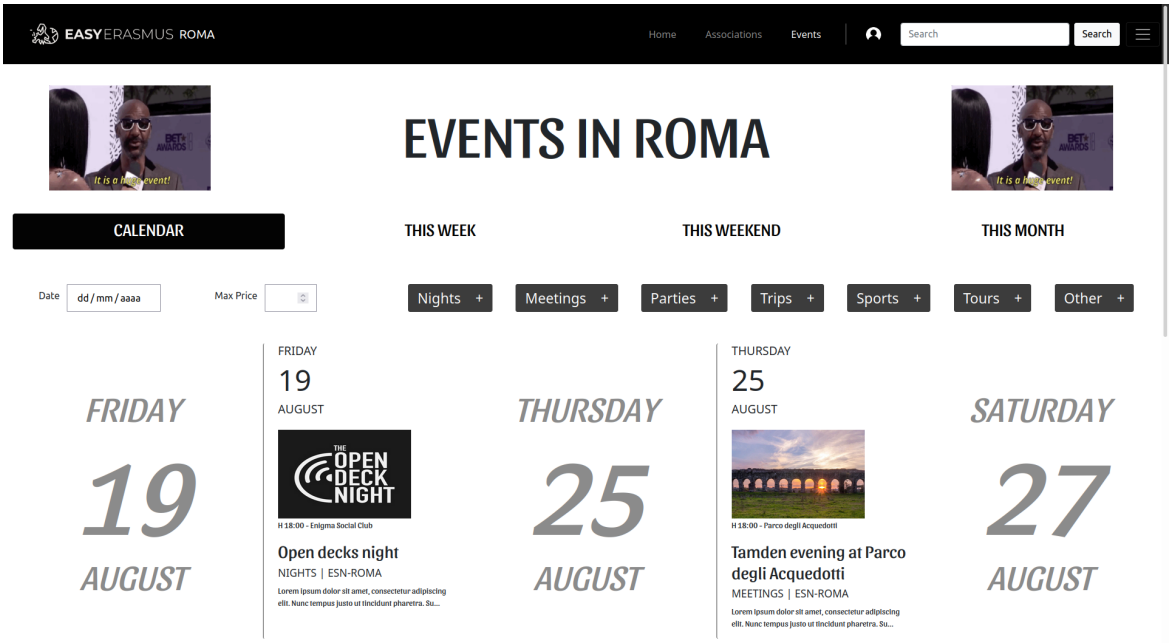
Solicitar suscripción a asociación: Desde el perfil de la asociación, pulse en el botón 'Request subscription'. Su petición será creada y recibida por parte de la asociación en cuestión. Si la asociación es abierta, no requerirá de su aceptación.

Vista 27: Participación en asociación.

Cancelar suscripción a asociación: Desde el perfil de la asociación, pulse en el botón 'Unsubscribe'. Su suscripción será eliminada junto a sus participaciones en eventos de la asociación en cuestión.

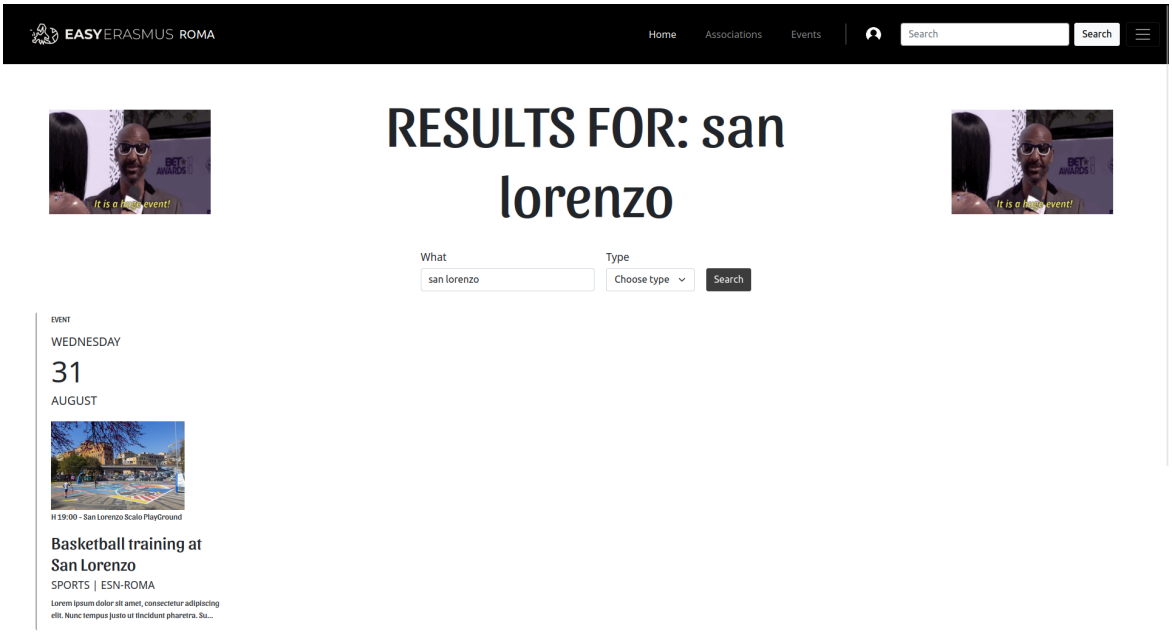
Vista 28: Baja de la suscripción en asociación.

Acceder al calendario de eventos: Pulse en el enlace 'Events' de la barra de navegación para acceder al listado de eventos próximos en su destino. Puede filtrar la selección por fecha, precio y tipo de evento.



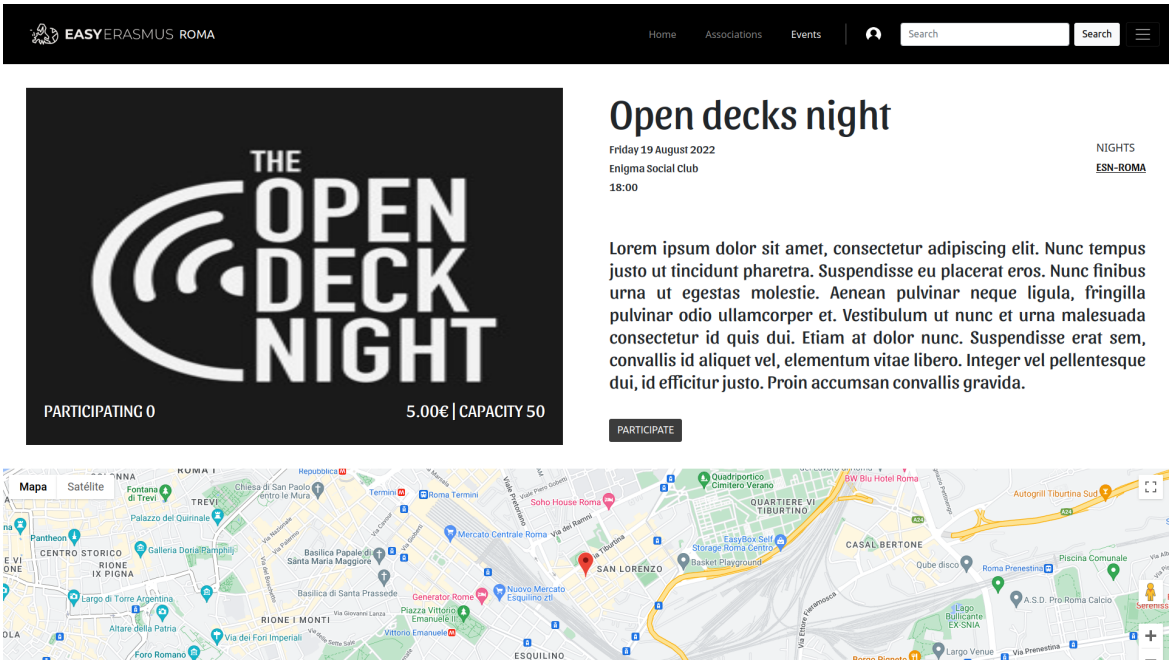
Vista 29: Calendario de eventos en destino.

Búsqueda de asociaciones o eventos en destino: En el campo de búsqueda de la barra de navegación ,escriba información relacionada con la asociación, el evento o el lugar donde se celebra.



Vista 30: Método de búsqueda.

Participar en evento: Desde el perfil de un evento de una asociación a la que se pertenece, se puede dar constancia de su participación pulsando el botón 'Participate'. Esto actualizará el número de participantes del evento. Si el evento se encontrase en 'Sold Out' (Agotado), no se podrá participar hasta que haya plazas de nuevo.

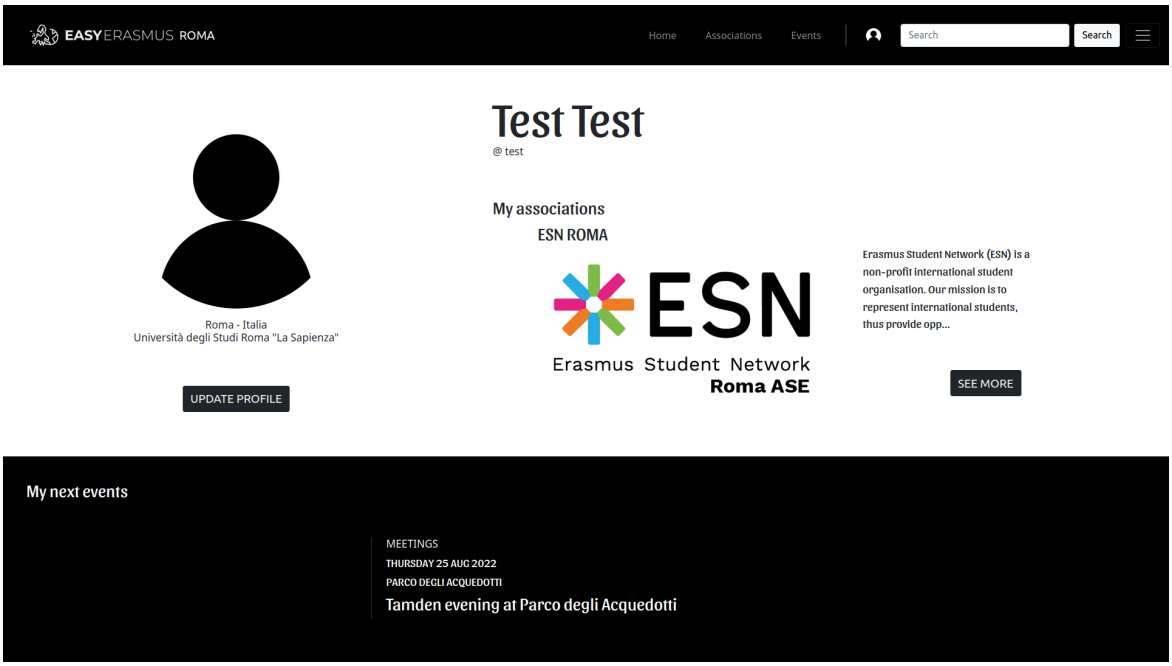


Vista 31: Participación en evento.

Dejar de participar en evento: Desde el perfil de un evento en el que se participa, es posible dejar de hacerlo pulsando el botón 'I'm not coming anymore'. Esto eliminará su participación y actualizará el número de participantes del evento.

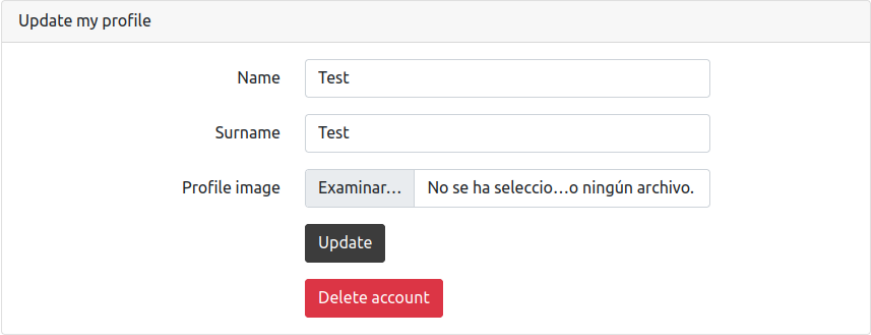
Vista 32: Dejar de participar en evento.

Acceder al perfil de usuario: Clickando en el icono de usuario de la barra de navegación será redirigido a su perfil, donde aparecen sus asociaciones, eventos e información.



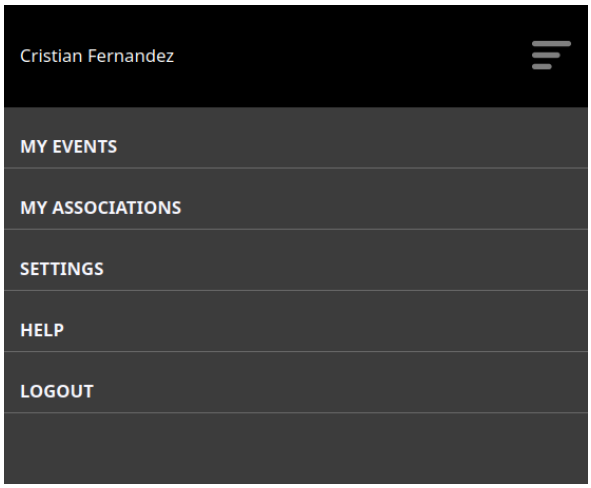
Vista 33: Perfil de usuario.

Modificar perfil de usuario: Desde el perfil de usuario, pulsando en el botón 'Update profile', será redirigido al formulario de actualización de usuario. También se puede acceder pulsando el enlace 'Settings' de la barra lateral.



Vista 34: Actualización de usuario.

Barra de navegación de usuario: Permitirá el acceso a las asociaciones a las que se está suscrito ("My associations"), así como los eventos en los que participa ("My events"). "Settings"le llevará a la modificación de sus datos de usuario.



Vista 35: Barra de navegación lateral de usuario.

Bibliografía

A., DEYIMAR. "¿Qué Es cPanel? Pros Y Contras + Cómo Usarlo." *Tutoriales Hostinger*, 28 junio 2022. Disponible: https://www.hostinger.es/tutoriales/que-es-cpanel (visitado 06/09/22).

ALTUBE VERA, RAFAEL. "Qué Es Laravel: Características Y Ventajas" *OpenWebinars.net*, 24 marzo 2021. Disponible: https://openwebinars.net/blog/que-es-laravel-caracteristicas-y-ventajas/ (visitado 06/09/22).

B., GUSTAVO. "¿Qué Es AJAX Y Cómo Funciona?" *Tutoriales Hostinger,* 2 marzo 2022. Disponible: https://www.hostinger.es/tutoriales/que-es-ajax (visitado 06/09/22).

B., GUSTAVO. "¿Qué Es CSS?" *Tutoriales Hostinger*, 12 julio 2022. Disponible: https://www.hostinger.es/tutoriales/que-es-css (visitado 06/09/22).

CAO, JERRY , ELLIS, MATT Y KHACHATRYAN, NAREK. *The guide to mockup.* UXPin

ECHAZARRETA, MATÍAS. "Haciendo Pruebas Automatizadas En Laravel [PARTE 1]." *Laravel Tip*, 12 junio 2020. Disponible: https://www.laraveltip.com/haciendo-pruebas-automatizadas-en-laravel/ (visitado 06/09/22).

GARCÍA DE ZÚÑIGA, FERNÁN. "¿Qué Es phpMyAdmin Y Cómo Usarlo?" *Blog De arsys.es*, 6 julio 2022. Disponible: https://www.arsys.es/blog/phpmyadmin (visitado 06/09/22).

IGHODARO, NEO. "How Laravel Implements MVC and How to Use It Effectively." *Blog - Pusher*, 22 agosto 2018. Disponible: https://blog.pusher.com/laravel-mvc-use/ (visitado 06/09/22).

JAQUE BARBERO, MIGUEL. "Formas Normales (1FN, 2FN, 3FN Y FNBC)." *Academia 19E37*, 7 noviembre 2015. Disponible: http://19e37.com/blog/formas-normales-1fn-2fn-3fn/ (visitado 06/09/22).

KENDALL, KENNETH E. y KENDALL, JULIE E.. "Capítulo 2, Comprensión de los estilos organizacionales y su impacto sobre los sistemas de información." *Análisis y diseño de sistemas.* Pearson educación, 2005, p 30.

OROZCO, SERGIO. "Uso De UML En Aplicaciones Web." *SG Buzz,* 2008. Disponible: https://sg.com.mx/content/view/584 (visitado 06/09/22).

PALACIOS, DUILIO. "Plantillas Blade - Documentación De Laravel 6." *Styde.net*, 29 mayo 2020. Disponible: https://styde.net/laravel-6-doc-plantillas-blade/ (visitado 06/09/22).

QUIROZ, JAVIER. *El modelo relacional de bases de datos*. Boletín de Política Informática Núm. 6, 2003, p 5.

ROBLEDANO, ÁNGEL. "Qué Es MySQL: Características Y Ventajas." *OpenWebinars.net*, 24 septiembre 2019. Disponible: https://openwebinars.net/blog/que-es-mysql/ (visitado 06/09/22).

SALAS CRUZ, ROCÍO , GÓMEZ MIRANDA, PILAR y MOTA MÁRQUEZ, OSCAR EDUARDO. "Unidad 2, Diseño Arquitectónico." *UPIICSA: Ingeniería de diseño*. pp 4-5.

S. PRESSMAN, ROGER y R. MAXIM, BRUCE. "Modelo de la Cascada." *Ingeniería del Software: Un Enfoque Práctico*. 7ª ed, McGraw-Hill Companies, 1998. pp 33-35.

"Bootstrap: Guía Para Principiantes De Qué Es, Por Qué Y Cómo Usarlo." *Rockcontent Blog*, 12 abril 2020. Disponible: https://rockcontent.com/es/blog/bootstrap/ (visitado 06/09/22).

"Designing for macOS." *Developer Apple*. Disponible: https://developer.apple.com/design/human-interface-guidelines/platforms/designing-for-macos (visitado 06/09/22).

Fundamentos De JavaScript - Aprende Sobre Desarrollo Web | MDN. 5 septiembre 2022. Disponible: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics (visitado 06/09/22).

LucusHost. Disponible: https://www.lucushost.com/hosting-gratis (visitado 06/09/22).

"Pruebas De Caja Negra: Ejemplos." *La Oficina De Proyectos De Informática*, 20 febrero 2017. Disponible: http://www.pmoinformatica.com/2017/02/pruebas-de-caja-negra-ejemplos.html (visitado 06/09/22).

"Qué es HTML." *Desarrollo Web*, 1 enero 2001. Disponible: https://desarrolloweb.com/articulos/que-es-html.html (visitado 06/09/22).

"¿Qué Es Una Prueba De Caja Blanca? Técnicas, Muestras Y Tipos." *Ebooks Online*. Disponible: https://ebooksonline.es/que-es-una-prueba-de-caja-blanca-tecnicas-muestras-y-tipos/ (visitado 06/09/22).

"¿Qué Es Y Para Qué Sirve PHP?" *Quality Devs*, 31 mayo 2021. Disponible: https://www.qualitydevs.com/2021/05/31/que-es-para-que-sirve-php/ (visitado 06/09/22).

Sass: Sass Basics. Disponible: https://sass-lang.com/guide (visitado 06/09/22).

"Site-Shot . Capture a Website Screenshot." *Site-Shot*. Disponible: https://www.site-shot.com/ (visitado 06/09/22).

"Salario En España - Salario Medio." *Talent.com*. Disponible: https://es.talent.com/salary (visitado 06/09/22).

"Testing: Getting Started." *Laravel Docs*, 9.x. Disponible: https://laravel.com/docs/9.x/testing (visitado 06/09/22).

"Web Servers - Apache." *Ubuntu*. Disponible: https://ubuntu.com/server/docs/web-servers-apache (visitado 06/09/22).