



TRABAJO FIN DE GRADO  
GRADO EN INGENIERÍA INFORMÁTICA  
MENCIÓN EN INGENIERÍA DEL SOFTWARE



# **Aplicación web para una ONG que asesora a personas migrantes o en situación de vulnerabilidad social**

**Estudiante:** Miguel Crespo Rozados

**Dirección:** Fernando Bellas Permuy

A Coruña, Septiembre de 2023.

*A mi familia*

### **Agradecimientos**

Quiero comenzar expresando mi agradecimiento a mi familia por el apoyo que me han brindado. En particular, quiero destacar el continuo esfuerzo de mis padres por siempre buscar que pueda alcanzar mi mejor versión.

Deseo extender mi agradecimiento a Fernando, por su colaboración a lo largo de todo el proyecto.

Quiero expresar mi reconocimiento a NTT Data, por ofrecer la posibilidad de realizar un trabajo práctico que resulte de utilidad.

Quiero hacer mención especial a AMIGA, la ONG a la que va destinado este proyecto, por el tiempo dedicado en las reuniones de requisitos. Agradezco que pueda haber contribuido a ayudar a las personas a las que atienden.

## Resumen

El objetivo de este proyecto consiste en el diseño y implementación de una aplicación web que tiene como objetivo agilizar tareas que realizan de manera manual en [AMIGA](#), una ONG que presta asistencia a personas migrantes y/o en situación de vulnerabilidad social. Esta solución informática está enfocada a optimizar su flujo de trabajo, permitiendo el registro de datos de las personas a las que brindan asistencia, así como el seguimiento y registro de las atenciones que llevan a cabo. Adicionalmente, facilita la generación automatizada de estadísticas, anteriormente elaboradas manualmente, y que necesitaban ser enviadas a la Xunta.

## Abstract

The goal of this project consists of the design and implementation of a web application that aims to streamline tasks performed manually in [AMIGA](#), an NGO that provides assistance to migrants and/or people in situations of social vulnerability. This computer solution is focused on optimizing their workflow, allowing the registration of data of the people to whom they provide assistance, as well as the monitoring and registering of the care they carry out. Additionally, it facilitates the automated generation of statistics, previously prepared manually, and that needed to be sent to the Xunta.

### Palabras clave:

- Aplicación web SPA
- Java
- Spring Boot
- JavaScript
- React
- MySQL
- Participantes
- Acogida
- Atención

### Keywords:

- SPA web application
- Java
- Spring Boot
- JavaScript
- React
- MySQL
- Participants
- Welcome
- Care

# Índice general

---

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Contexto . . . . .	1
1.2	Objetivos . . . . .	1
1.3	Visión global del sistema . . . . .	3
<b>2</b>	<b>Metodología</b>	<b>4</b>
2.1	Enfoque Metodológico . . . . .	4
2.2	Razonamiento de la Elección . . . . .	5
<b>3</b>	<b>Análisis de requisitos global</b>	<b>6</b>
3.1	Actores . . . . .	6
3.2	Casos de uso . . . . .	6
3.3	Flujo principal de trabajo . . . . .	12
<b>4</b>	<b>Planificación</b>	<b>14</b>
4.1	Iteraciones . . . . .	14
4.1.1	Iteración 0 . . . . .	14
4.1.2	Iteración 1 . . . . .	15
4.1.3	Iteración 2 . . . . .	15
4.1.4	Iteración 3 . . . . .	16
4.1.5	Iteración 4 . . . . .	16
4.1.6	Iteración 5 . . . . .	17
4.1.7	Iteración 6 . . . . .	17
4.2	Planificación temporal . . . . .	18
4.3	Cálculo de costes . . . . .	18
<b>5</b>	<b>Fundamentos tecnológicos</b>	<b>20</b>
5.1	Tecnologías utilizadas en el backend . . . . .	20

5.2	Tecnologías utilizadas en el frontend . . . . .	21
5.3	Tecnologías complementarias . . . . .	22
<b>6</b>	<b>Desarrollo</b>	<b>23</b>
6.1	Estructura de la aplicación . . . . .	23
6.1.1	Estructura del backend . . . . .	23
6.1.2	Estructura del frontend . . . . .	24
6.2	Modelo de datos . . . . .	26
6.3	Iteración 1 . . . . .	32
6.3.1	Análisis . . . . .	32
6.3.2	Diseño e implementación . . . . .	32
6.4	Iteración 2 . . . . .	41
6.4.1	Análisis . . . . .	41
6.4.2	Diseño e implementación . . . . .	41
6.5	Iteración 3 . . . . .	46
6.5.1	Análisis . . . . .	46
6.5.2	Diseño e implementación . . . . .	47
6.6	Iteración 4 . . . . .	52
6.6.1	Análisis . . . . .	52
6.6.2	Diseño e implementación . . . . .	52
6.7	Iteración 5 . . . . .	57
6.7.1	Análisis . . . . .	57
6.7.2	Diseño e implementación . . . . .	57
6.8	Iteración 6 . . . . .	62
6.8.1	Swagger UI . . . . .	62
6.8.2	Despliegue . . . . .	64
<b>7</b>	<b>Conclusiones</b>	<b>65</b>
7.1	Trabajo futuro . . . . .	66
	<b>Lista de acrónimos</b>	<b>72</b>
	<b>Glosario</b>	<b>73</b>
	<b>Bibliografía</b>	<b>74</b>

# Índice de figuras

---

1.1	Arquitectura del sistema. . . . .	3
3.1	Mockup primera página formulario acogida. . . . .	7
3.2	Mockup página intermedia formulario acogida. . . . .	7
3.3	Mockup página descarga pdf y guardado de datos. . . . .	8
3.4	Mockup búsqueda participante por documento. . . . .	9
3.5	Mockup crear atención. . . . .	10
6.1	Estructura del backend. . . . .	24
6.2	Estructura del frontend. . . . .	26
6.3	Primera parte diagrama de entidades. . . . .	30
6.4	Segunda parte diagrama de entidades. . . . .	31
6.5	Diagrama clases implementadas iteración 1. . . . .	33
6.6	Confirmación formulario acogida. . . . .	35
6.7	Búsqueda por documento identificativo. . . . .	36
6.8	Registro datos año actual participante. . . . .	38
6.9	Visualizar datos de participante. . . . .	40
6.10	Crear atención. . . . .	42
6.11	Ver atención. . . . .	43
6.12	Búsqueda de atenciones. . . . .	45
6.13	Visualizar inserciones laborales. . . . .	46
6.14	Registrar inserciones laborales. . . . .	47
6.15	Excel generado. . . . .	49
6.16	Gráfico nacionalidades. . . . .	50
6.17	Diagrama de secuencia de autorización con JWT. . . . .	53
6.18	Iniciar sesión. . . . .	53
6.19	Cambiar contraseña. . . . .	54
6.20	Buscar usuario. . . . .	55

6.21	Crear usuario. . . . .	56
6.22	Buscar voluntario. . . . .	58
6.23	Ver detalles voluntario. . . . .	59
6.24	Registrar voluntario. . . . .	60
6.25	Modal creación colaboración. . . . .	61
6.26	Modal editar colaboración. . . . .	61
6.27	Swagger UI. . . . .	62
6.28	Detalles endpoint swagger. . . . .	63
1	Primera página formulario acogida. . . . .	68
2	Segunda página formulario acogida. . . . .	69
3	Tercera página formulario acogida. . . . .	70
4	Cuarta página formulario acogida. . . . .	71



# Índice de cuadros

---

4.1	Planificación temporal y tiempo real empleado en el proyecto . . . . .	18
4.2	Cálculo del coste total del proyecto . . . . .	19

# Introducción

---

## 1.1 Contexto

Asociación de Migrantes de Galicia (AMIGA) es una ONG (<https://amiga.gal>) cuyo objetivo general es promover la inclusión sociolaboral y luchar contra la pobreza, contribuyendo a la igualdad de oportunidades, integración, participación y la promoción del bienestar de la población migrante o en situación de vulnerabilidad social.

Los empleados de la organización AMIGA tienen la responsabilidad de gestionar datos relacionados con las personas a las que brindan asesoramiento, quienes pueden ser migrantes o encontrarse en situaciones de vulnerabilidad social. Para referirnos a estas personas, las denominaremos "participantes", conforme al término utilizado en AMIGA. Actualmente, estas labores se realizan de forma manual en hojas de Excel.

Por cada participante, se maneja un conjunto completo de información que abarca su ciclo de vida, incluyendo su registro (datos personales básicos) y su proceso de acogida, así como el asesoramiento tanto jurídico como sociolaboral que puedan requerir. Además de esto, se busca tener la capacidad de gestionar a los voluntarios que colaboran con AMIGA.

El desarrollo de esta aplicación web se plantea como una solución para automatizar la creación de informes estadísticos y la generación de expedientes individuales para cada participante. Estas funcionalidades agilizarán enormemente el trabajo de los empleados de la ONG, con lo que conseguiría un notable ahorro de tiempo.

## 1.2 Objetivos

La aplicación que se ha desarrollado supondrá un gran cambio en la forma en que los trabajadores de AMIGA desempeñan su trabajo, al permitirles realizar una serie de tareas de

manera eficiente. A continuación, se detallan las capacidades clave que la aplicación ofrecerá:

**Gestión del Ciclo de Vida de los Participantes:** La aplicación facilitará el proceso de gestión del ciclo de vida de cada participante, abordando diversos aspectos fundamentales:

- **Registro de Entrada:** Los datos de cada participante serán registrados en el programa.
- **Registro histórico:** Anualmente, se registra y almacena la información del participante con el propósito de mantener una trazabilidad de su evolución.
- **Acogida y Valoración Social:** Se llevará a cabo una evaluación exhaustiva de las necesidades de cada participante.
- **Registro de Atenciones:** Se permitirá el registro de todas las interacciones y asistencias brindadas, abarcando áreas legales y sociolaborales.
- **Registro de Inserciones Laborales:** Se documentarán todas las inserciones laborales logradas en colaboración con los participantes.

**Gestión del Voluntariado:** La aplicación brindará la funcionalidad para la administración del voluntariado, abriendo la posibilidad de que tanto personas externas como los propios participantes desempeñen roles como voluntarios.

**Generación de Información Estadística y Expedientes:** Además de facilitar las operaciones diarias, la aplicación ofrecerá capacidades de generación de informes:

- **Información Estadística:** La generación automatizada de estadísticas será posible, permitiendo la creación de la memoria anual de actividades en formato Excel.
- **Expedientes de Participantes:** Los expedientes de los participantes se generarán en formato PDF, agilizando su gestión y consulta.

**Cumplimiento del Reglamento General de Protección de Datos:** Es importante destacar que la aplicación se ha desarrollado con la idea de cumplir el Reglamento General de Protección de Datos, por lo que no se almacena la información hasta que se obtenga el consentimiento explícito por parte de los participantes.

En resumen, esta aplicación proveerá a **AMIGA** con una herramienta para optimizar sus operaciones diarias, administrar su voluntariado, generar informes esenciales y cumplir con la seguridad de los datos personales.

### 1.3 Visión global del sistema

El sistema a desarrollar consiste en una aplicación web formada por un **backend** y un **frontend**. En la figura 1.1 se puede ver un diagrama del sistema.

El **backend** de la aplicación, en el que está contenida la lógica de negocio de esta, es un API REST que se ha implementado usando tecnologías Java con la ayuda del **framework** Spring Boot, y utilizando MySQL como base de datos.

El **frontend** de la aplicación se trata de una aplicación web **Single Page Application (SPA)** que se ha implementado con JavaScript, con la ayuda de librerías como React, Redux y Material-UI.

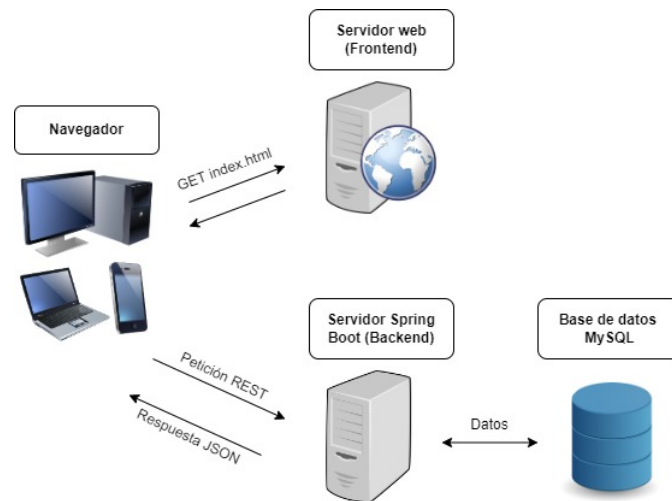


Figura 1.1: Arquitectura del sistema.

## Capítulo 2

# Metodología

---

### 2.1 Enfoque Metodológico

Para el desarrollo de este proyecto, se ha optado por una metodología iterativa e incremental centrada en las funcionalidades del sistema.

La metodología se inicia con un análisis global de los requisitos del sistema. A partir de estos requisitos, se procede a realizar un desarrollo iterativo de la aplicación. En cada iteración, se aborda un conjunto específico de requisitos previamente definidos.

Cada iteración parte del producto desarrollado en la iteración anterior, sobre el cual se aplican las etapas convencionales de análisis, diseño, implementación y pruebas. Al finalizar cada iteración, se produce un producto con mayor funcionalidad en comparación con la iteración precedente, lo que permite que los usuarios puedan aprovecharlo plenamente.

**Las ventajas clave de esta metodología son:**

- El cliente puede obtener resultados significativos y utilizables desde las primeras iteraciones.
- La gestión de los cambios se facilita gracias al feedback del cliente después de cada iteración.
- Se logra una comprensión temprana del progreso real del proyecto.
- El cliente minimiza el riesgo al perder únicamente los recursos invertidos en una iteración, lo que disminuye la posibilidad de fracaso del proyecto en su conjunto.

**Por otro lado, las principales desventajas incluyen:**

- Requiere una participación activa del cliente durante todo el proceso del proyecto.
- Puede surgir dificultad para determinar qué requisitos priorizar para su implementación en las primeras iteraciones.

En la siguiente sección, se proporciona un razonamiento detallado sobre la elección de la metodología iterativa.

## 2.2 Razonamiento de la Elección

A pesar de que Scrum [1] es una de las metodologías iterativas más populares en el desarrollo de software, a continuación, se presentan las razones clave por las cuales se decidió no aplicarla en este proyecto:

- Es una metodología diseñada para proyectos que se desarrollan colaborativamente en equipos. En estos equipos, roles específicos (como Product Owner y Scrum Master) son necesarios para llevar a cabo diversas responsabilidades. Dado que este proyecto es individual y no involucra un equipo, la estructura de roles de Scrum no sería aplicable.
- Además, Scrum opera mediante ciclos de tiempo fijos y cortos llamados "Sprints". La rigidez de estos plazos no se adapta a la flexibilidad de tiempo requerida en este proyecto debido a otras responsabilidades.

Por estas razones, se eligió una metodología iterativa personalizada. Esta se adapta mejor al contexto de un proyecto individual, permitiendo mayor flexibilidad en el tiempo dedicado a cada iteración. Además, la ventaja del desarrollo iterativo de obtener feedback del cliente después de cada iteración resultaba muy adecuada en este caso.

# Análisis de requisitos global

---

A continuación se describen en este capítulo los actores que podrán acceder al sistema y los requisitos globales del mismo.

## 3.1 Actores

Como se ha citado en el apartado 1.2 donde se mencionan los objetivos, los únicos usuarios son los trabajadores de **AMIGA**. Se ha implementado una distinción entre los siguientes roles::

- **Administrador:** Además de las funcionalidades estándar, este rol cuenta con acceso a la gestión de técnicos.
- **Técnico:** Este actor tiene acceso al resto de las funcionalidades proporcionadas por la aplicación, este sería el rol común de la mayoría de los trabajadores.

## 3.2 Casos de uso

Los requisitos del sistema se recogieron como casos de uso y son los expuestos a continuación:

- **CU-01 Acogida participante:** El técnico registra los datos correspondientes al participante y, al completar esta tarea, se genera un archivo PDF que contiene toda la información, incluyendo la legislación de protección de datos. Este documento se descarga y proporciona al solicitante para su firma antes de proceder al guardado de los datos.

La cantidad de datos recopilados en este proceso es muy extensa, y se necesitaron varias reuniones para detallar todos los campos. Por lo tanto, los mockups se utilizaron para mostrar el formato que se seguiría en la primera página (consulte la figura 3.1), en una página intermedia (consulte la figura 3.2), y en la página de confirmación (consulte la figura 3.3).

The mockup shows a web interface for the 'Acogida' (Welcome) section. At the top, there is a navigation bar with a 'Logo' placeholder, four tabs ('Acogida', 'Estadísticas', 'Voluntariado', 'Empleados'), and a user profile dropdown labeled 'Nombre usuario'. Below the navigation bar, the title 'Datos personales' is centered. The form contains several input fields and dropdown menus: 'Nombre' and 'Apellidos' (text inputs), 'Fecha de nacimiento' (date picker), 'Sexo' (dropdown), 'DNI', 'NIE', and 'PAS' (text inputs), 'Provincia' and 'Municipio' (dropdowns), 'Vivienda', 'Estado civil', and 'Nacionalidad' (dropdowns), and 'Tipo convivencia' and 'Pais origen' (dropdowns). A 'Siguiente' (Next) button is located at the bottom center.

Figura 3.1: Mockup primera página formulario acogida.

The mockup shows the intermediate page of the 'Acogida' form, titled 'Formación'. It features the same navigation bar as the first page. The form includes: 'Nivel estudios' (dropdown), 'Homologados' (dropdown), 'Idiomas adicionales' (text input), 'Formación demandada' (text input), 'Situación laboral' (text input), 'Cualificación profesional' (text input), 'Habilidades' (text input), and 'Horario' (text input). At the bottom, there are two buttons: 'Anterior' (Previous) and 'Siguiente' (Next).

Figura 3.2: Mockup página intermedia formulario acogida.



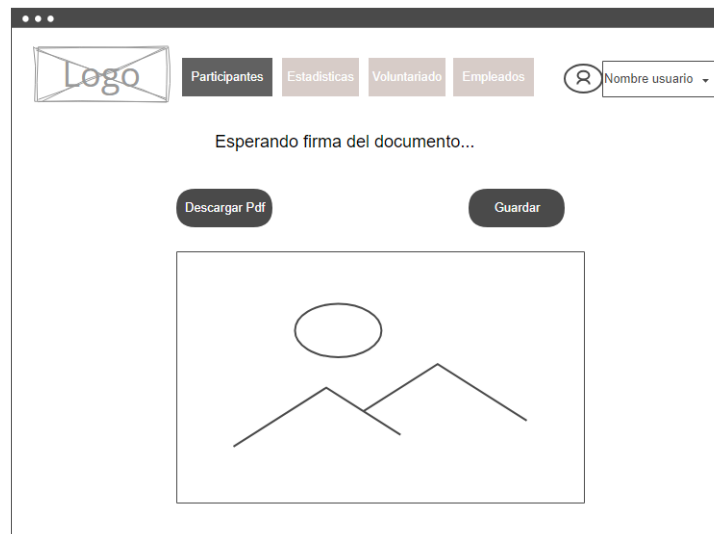


Figura 3.3: Mockup página descarga pdf y guardado de datos.

- **CU-02 Búsqueda participante por documento identificativo:** El sistema permite la opción de buscar entre distintas alternativas las cuales son DNI/NIE/PAS, y se solicita ingresar el número de documento correspondiente. Si se produce una coincidencia, se recuperan los datos básicos del participante. En caso contrario, se señala un error. Además, se realiza una validación del formato ingresado en el caso de DNI o NIE para garantizar que se haya escrito correctamente (consulte la figura 3.4).

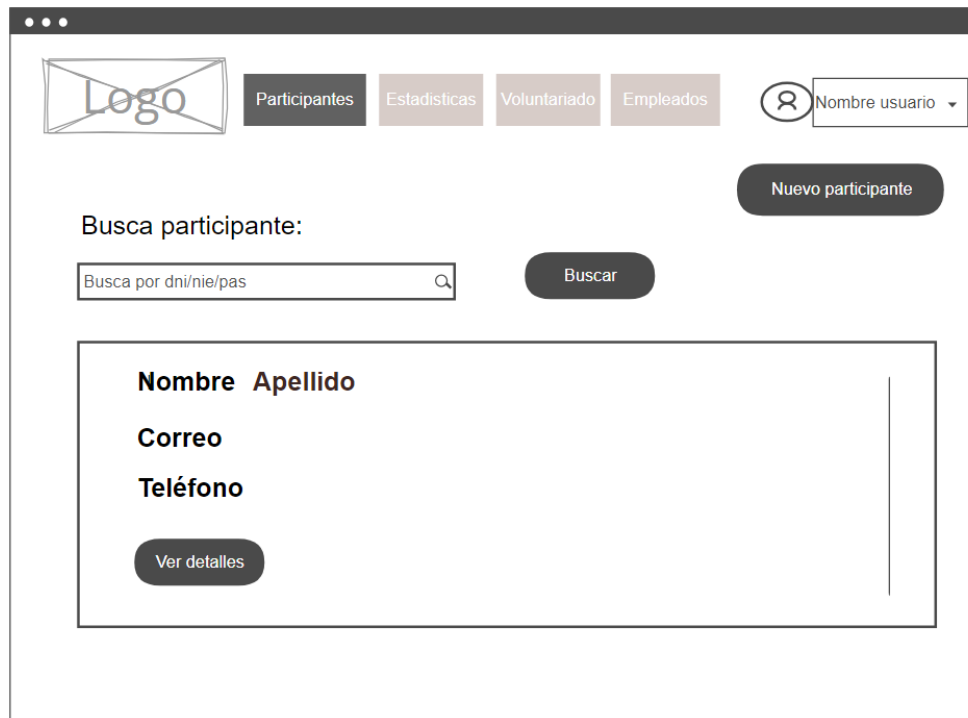



Figura 3.4: Mockup búsqueda participante por documento.

- **CU-03 Registrar datos año actual participante:** Cada participante debe contar con un registro de sus datos correspondientes al año actual. Esto puede deberse tanto a que se trata del primer registro en el año en curso como a que ha vuelto a solicitar asesoramiento. El sistema permite realizar un nuevo registro, en el cual solo se deben indicar los campos que han cambiado con respecto al último año en el que recibió asesoramiento. Este enfoque tiene como objetivo almacenar una trazabilidad de los datos de cada participante a lo largo de los diferentes años.
- **CU-04 Visualizar datos de participante para un año determinado:** El sistema genera un informe en formato PDF con los datos relativos al participante en el año seleccionado.
- **CU-05 Editar datos del participante para un año específico:** El sistema permite editar los datos del participante en el año seleccionado.
- **CU-06 Crear atención:** El sistema permite registrar los datos esenciales de una atención, los cuales comprenden la fecha, el tipo, el título y la descripción (consulte la figura 3.5).



Logo

Participantes Estadísticas Voluntariado Empleados

Nombre usuario

Nueva atención Nombre Apellido

Fecha: 21 21 21 Tipo: nombre

Título:

Texto:

Enviar

Figura 3.5: Mockup crear atención.

- **CU-07 Editar atención:** El usuario tiene la capacidad de modificar la información de una atención.
- **CU-08 Eliminar atención:** El usuario puede eliminar la atención que ha seleccionado.
- **CU-09 Ver detalles atención:** El usuario puede ver todos los datos relacionados con la atención seleccionada.
- **CU-10 Buscar atenciones mediante filtros:** El usuario tiene la opción de aplicar filtros en las atenciones de un participante, tales como el tipo de atención y un rango de fechas que abarca la fecha de inicio, fecha final o un intervalo de fechas.
- **CU-11 Crear inserción laboral:** El usuario puede registrar información sobre las inserciones laborales realizadas por los participantes, incluyendo los siguientes datos: fecha, tipo de contrato, sector, tipo de jornada y, si el tipo de contrato no se encuentra entre las opciones del sistema, se proporciona un campo para introducirlo manualmente.
- **CU-12 Visualizar inserciones laborales:** El usuario tiene la capacidad de visualizar las inserciones laborales de un participante.
- **CU-13 Editar inserción laboral:** El usuario tiene la capacidad de modificar la información de una inserción laboral.
- **CU-14 Eliminar inserción laboral:** El usuario puede eliminar la inserción laboral que ha seleccionado.

- **CU-15 Generar estadísticas:** El sistema genera las siguientes gráficas utilizando los datos de los participantes que han recibido alguna atención dentro del rango de fechas especificado por el usuario:
  - Según sexo
  - Distribución edad por sexo
  - Factores de exclusión por sexo
  - Nacionalidades por sexo
  - Tipo de contratos por sexo
  - Tipo jornada por sexo

Además, se crea un archivo de Excel que contiene los datos de los usuarios, permitiendo a los propios trabajadores generar nuevas gráficas según sus necesidades.

- **CU-16 Iniciar sesión:** Requiere que el usuario introduzca su nombre de usuario y contraseña para acceder al sistema. Una vez se complete la autenticación exitosamente, se concederá acceso a todas las funcionalidades del sistema.
- **CU-17 Cerrar sesión:** Un usuario autenticado tiene la opción de cerrar sesión. Después del cierre de sesión, solo tendrá acceso a la funcionalidad de iniciar sesión.
- **CU-18 Cambiar contraseña:** Un usuario autenticado tiene la capacidad de modificar su contraseña. Para ello, deberá ingresar su contraseña actual e introducir de manera duplicada la nueva contraseña.
- **CU-19 Registrar nuevo trabajador:** Un administrador tiene la capacidad de dar de alta a nuevos usuarios, para que puedan acceder a la aplicación.
- **CU-20 Buscar trabajador por palabra clave:** Un administrador puede buscar empleados utilizando una palabra clave, ya sea un nombre completo o parcial. El sistema buscará coincidencias en nombres y apellidos entre los empleados registrados.
- **CU-21 Cambiar permisos trabajador:** Un administrador tiene la facultad de modificar el rol asignado a un usuario, lo que implica cambiar los permisos y nivel de acceso asociados a ese usuario.
- **CU-22 Eliminar trabajador:** Un administrador tiene la capacidad de eliminar a un usuario del sistema, revocando por completo su acceso y sus permisos.
- **CU-23 Editar datos trabajador:** Un técnico puede editar sus datos personales.

- **CU-24 Registrar datos voluntario:** El técnico registra los datos correspondientes al voluntario que quiere colaborar en las labores de AMIGA.
- **CU-25 Buscar voluntario:** El sistema permite la búsqueda de voluntarios mediante coincidencias en sus nombres y apellidos, además de la capacidad de aplicar filtros para localizar aquellos voluntarios que tienen actualmente colaboraciones activas.
- **CU-26 Ver detalles voluntario:** El sistema facilita la obtención de los datos correspondientes a un voluntario en específico, junto con una lista detallada de las colaboraciones que ha realizado en el pasado y/o que mantiene en la actualidad.
- **CU-27 Editar datos voluntario:** El técnico cuenta con la capacidad de modificar los datos de un voluntario, para facilitar la corrección de algún campo erróneo.
- **CU-28 Crear colaboración:** El proceso implica la creación de un registro dedicado al voluntario, en el cual se almacenan los datos relacionados con su colaboración.
- **CU-29 Editar colaboración:** El sistema posibilita la actualización de los datos de una colaboración con el propósito de lograr los siguientes objetivos: registrar nuevas horas de dedicación, definir la fecha de finalización de la colaboración y/o corregir cualquier información incorrecta.
- **CU-30 Eliminar colaboración:** El usuario puede eliminar la colaboración del voluntario que ha seleccionado.

### 3.3 Flujo principal de trabajo

Dado que se trata de un flujo de trabajo complejo adaptado a la forma de trabajo de AMIGA, a continuación, se proporciona una explicación del flujo principal de trabajo para facilitar la comprensión proceso.

Cuando un participante asiste a AMIGA, se abren varios caminos a seguir, dependiendo de sus circunstancias. Estas posibilidades pueden incluir:

- **Registro Inicial:** Si es la primera vez que el participante acude a AMIGA, se crea un registro inicial en el sistema con sus datos personales **CU-01**.
- **Primera asistencia en el año actual:** Si el participante ha recibido atención en años previos pero es la primera vez que se le asiste en el año actual, se lleva a cabo el proceso denominado **CU-03 Registrar datos del participante para el año actual**. Durante este proceso, se actualizan aquellos datos que hayan cambiado, asegurando así que la

información refleje la situación actual en el presente año y que exista una trazabilidad de la evolución del participante.

- **Registro de una atención:** Cada vez que un participante asiste a AMIGA, el técnico registra la atención proporcionada (**CU-06**). Para obtener un contexto completo sobre la situación del participante, el técnico puede consultar las últimas atenciones realizadas utilizando los filtros de rangos de fechas y tipo de atención (**CU-10**). Además, puede visualizar el informe que contiene los datos del participante en cuestión (**CU-04**). Por otro lado, en el caso de que el participante haya conseguido un empleo, se pueden registrar los datos de la inserción laboral en el sistema mediante su respectivo caso de uso (**CU-11**).

A partir de los datos recopilados en el día a día, existe la capacidad de generar estadísticas sobre los participantes atendidos en un intervalo de fechas específico, el cual es definido por el usuario. Esta funcionalidad permite obtener información sobre la cantidad de participantes asistidos, los tipos de atención proporcionados y otros datos relevantes dentro del período de tiempo seleccionado, los cuales son requeridos por la Xunta en los informes de actividad.

# Planificación

---

En este capítulo, se abordará la planificación del proyecto utilizando una metodología iterativa. Además, se detallará el progreso logrado en cada una de las iteraciones, brindando una descripción de las actividades realizadas en cada etapa.

## 4.1 Iteraciones

El proyecto ha sido dividido en cinco iteraciones, las cuales se describen en detalle a continuación:

### 4.1.1 Iteración 0

El propósito principal de esta iteración es llevar a cabo la fase de concepción del producto. Durante esta etapa, se llevó a cabo un análisis global de los requisitos, con el fin de establecer los casos de uso que se tienen la intención de implementar y obtener una visión global del sistema. Además, en colaboración con el cliente, se llevó a cabo la tarea de establecer la prioridad de los casos de uso y definir la estrategia para dividirlos en las distintas iteraciones. Esta decisión conjunta permitió alinear las metas del proyecto y las expectativas del cliente, asegurando que las funcionalidades más relevantes y fundamentales fueran abordadas de manera temprana en el proceso de desarrollo.

Adicionalmente, durante estas reuniones de concepción del producto, se presentaron al cliente [mockups](#) detallados que representaban las pantallas de la aplicación. Estos mockups desempeñaron un papel fundamental como herramientas visuales para simplificar la comprensión y perfeccionamiento de los requisitos. Esta aproximación visual mejoró la comprensión de la funcionalidad deseada, lo que a su vez facilitó la realización de ajustes y mejoras de manera más eficiente.

### 4.1.2 Iteración 1

Esta etapa, con diferencia, fue la que demandó más tiempo debido a la complejidad del modelado de los datos de los participantes. Fue necesario profundizar en cada detalle durante las reuniones iniciales de la iteración.

**El trabajo realizado en esta iteración se puede resumir de la siguiente manera:**

- **Se creó el esqueleto del proyecto**, abarcando tanto el **backend** como del **frontend**.
- **Se configuraron los contenedores de Docker** para instanciar bases de datos MySQL, estableciendo una para pruebas y otra para desarrollo.
- **Se llevó a cabo el proceso de aprendizaje de React-pdf** para la generación de documentos directamente desde el frontend.
- **Se implementaron los casos de uso relacionados con los datos de los participantes**, los cuales incluyen:
  - CU-01 Acogida participante.
  - CU-02 Búsqueda participante por documento identificativo.
  - CU-03 Registrar datos año actual participante.
  - CU-04 Visualizar datos de participante para un año determinado.
  - CU-05 Editar datos del participante para un año específico.

### 4.1.3 Iteración 2

Durante esta iteración, se llevaron a cabo las implementaciones restantes relacionadas con la gestión del ciclo de vida del participante, puesto que se destacó como una prioridad principal del cliente, dado que es la parte del sistema que tendrá un uso continuo en el día a día. Los casos de uso abordados fueron:

- CU-06 Crear atención.
- CU-07 Editar atención.
- CU-08 Eliminar atención.
- CU-09 Ver detalles atención.
- CU-10 Buscar atenciones mediante filtros.
- CU-11 Crear inserción laboral.



- CU-12 Visualizar inserciones laborales.
- CU-13 Editar inserción laboral.
- CU-14 Eliminar inserción laboral.

#### 4.1.4 Iteración 3

Durante esta fase, se abordó la tarea de generar estadísticas, la cual presentó un desafío considerable debido a la necesidad de aprender a utilizar Apache POI, una herramienta de Java para la generación de archivos Excel, y Recharts, una biblioteca para crear gráficos en el frontend. Debido a la complejidad inherente y la necesidad de aprendizaje, en esta ocasión solo se realizó la implementación de un único caso de uso, el CU-15: Generar estadísticas.

#### 4.1.5 Iteración 4

En esta iteración, el enfoque se dirigió hacia la implementación de los casos de uso relacionados con la autenticación y la administración de los usuarios de la aplicación, específicamente dirigidos a los trabajadores que utilizarán dicha aplicación. Se establecieron dos roles claramente diferenciados: el rol de "técnico", que posee acceso básico a la aplicación y la capacidad de modificar su información personal, y el rol de "admin", quien cuenta con la facultad de agregar y eliminar técnicos. Los casos de uso que se llevaron a cabo durante esta fase son los siguientes:

- CU-16 Iniciar sesión.
- CU-17 Cerrar sesión.
- CU-18 Cambiar contraseña.
- CU-19 Registrar nuevo trabajador.
- CU-20 Buscar trabajador por palabra clave.
- CU-21 Cambiar permisos trabajador.
- CU-22 Eliminar trabajador.
- CU-23 Editar datos trabajador.

#### 4.1.6 Iteración 5

En la última fase de desarrollo de funcionalidades, el enfoque consistió en la implementación de los casos de uso relacionados con la gestión del voluntariado. Las funcionalidades abordadas fueron las siguientes:

- CU-24 Registrar datos voluntario.
- CU-25 Buscar voluntario.
- CU-26 Ver detalles voluntario.
- CU-27 Editar datos voluntario.
- CU-28 Crear colaboración.
- CU-29 Editar colaboración.

#### 4.1.7 Iteración 6

La última etapa se centra principalmente en la elaboración de la memoria utilizando como base la aplicación ya desarrollada. Este período final se caracteriza por un enfoque en la documentación del proyecto y la optimización de la experiencia del usuario a través de ajustes y refinamientos en la interfaz.

## 4.2 Planificación temporal

La planificación temporal del proyecto experimentó variaciones en cada iteración y se estableció al inicio de cada una de ellas. Cada fase estuvo iniciada por reuniones con el cliente que sirvieron para afinar los detalles de la etapa por comenzar y para identificar áreas de mejora basadas en la iteración anterior. Este proceso permitió ajustar tanto el enfoque como la duración de acuerdo con las necesidades particulares de los objetivos a cumplir en cada fase concreta.

Iteración	Objetivo	Fecha inicio	Fecha fin	Estimación(h)	Tiempo real(h)
Iteración 0	Concepción del producto	08/02/2023	25/04/2023	20	17
Iteración 1	Configuración proyecto base, formación tecnológica y datos participantes	26/04/2023	20/06/2023	100	115
Iteración 2	Atenciones y inserciones laborales participantes	21/06/2023	07/07/2022	25	27
Iteración 3	Generar estadísticas	08/07/2023	20/07/2023	30	26
Iteración 4	Gestión de usuarios	21/07/2023	01/08/2023	25	23
Iteración 5	Gestión de voluntariado	02/08/2023	15/08/2022	20	23
Iteración 6	Memoria y mejoras en la aplicación	16/08/2023	05/09/2022	85	98
Tiempo total del proyecto				<b>305</b>	<b>322</b>

Cuadro 4.1: Planificación temporal y tiempo real empleado en el proyecto

## 4.3 Cálculo de costes

Para calcular el coste del proyecto se tuvieron en cuenta distintos costes asociados al mismo, como el coste de las horas trabajadas o diversos recursos necesarios. Se parte de un sueldo de programador junior de 15 euros/hora.

$$\text{Coste de horas trabajadas} = 322 \text{ horas} * 15 \text{ euros/hora} = \mathbf{4830 \text{ euros}}$$

Aunque la dedicación al proyecto fue constante, su distribución no fue uniforme a lo largo de todas las iteraciones. Con el objetivo de calcular una duración proyectada, suponiendo una asignación equitativa de tiempo durante todo el proyecto, se dividió la duración total entre las horas equivalentes a un mes laboral completo, es decir, aproximadamente ocho horas diarias durante veinte días hábiles:

$$\text{Duración} = 322 \text{ horas} / 160 \text{ horas/mes} = \mathbf{2,01 \text{ meses}}$$

De acuerdo con el cálculo realizado, si se mantuviera una dedicación constante a lo largo de todo el proyecto, la duración estimada sería de dos meses.

Los recursos que se necesitaron para realizar el proyecto son:

- **Ordenador:** Se adquirió un ordenador portátil con un coste de 1.200 euros. A continuación se calcula el coste de amortización del ordenador, con la siguiente fórmula:

$$C \cdot \frac{t}{v}$$

Donde  $C$  es coste del equipo (1200€),  $t$  son los meses que llevó el proyecto (2 meses) y  $v$  tiempo de vida útil del portátil (24 meses).

Siguiendo la fórmula anterior, el coste de amortización del ordenador es 100 euros.

- **Oficina:** Para el alquiler de una oficina de pocos metros cuadrados, se aproxima un coste de 180 euros/mes, por lo tanto serían 360 euros (2 meses).
- **Electricidad:** Se calcula que se gastarán sobre 60 euros en total (2 meses).
- **Agua:** Tuvo un coste total de 25 euros (2 meses).
- **Internet:** Contratando una tarifa básica de internet, por 20 euros al mes, por lo que fueron 40 euros en total (2 meses).

En la tabla a continuación se muestra el cálculo del coste total del proyecto

Concepto	Coste (euros)
Coste horas trabajadas	4830
Ordenador	100
Alquiler oficina	360
Electricidad	60
Agua	25
Internet	40
<b>Coste total del proyecto</b>	<b>5415</b>

Cuadro 4.2: Cálculo del coste total del proyecto

El coste total del proyecto fueron **5415 euros**.

## Fundamentos tecnológicos

---

### 5.1 Tecnologías utilizadas en el **backend**

- **Java** [2, 3]: es un lenguaje de programación interpretado, imperativo y orientado a objetos. Una vez compilada, una aplicación estándar Java puede ser ejecutada sobre cualquier sistema para el que haya disponible alguna versión del entorno de ejecución de Java, conocido como [Java Runtime Environment \(JRE\)](#).
- **Spring Boot** [4]: Spring Boot es un [framework](#) que facilita la creación de proyectos basados en Spring. Permite crear aplicaciones autocontenidas, configura dependencias y despliega la aplicación a un servidor de aplicaciones.
- **MySQL** [5]: es un sistema de gestión de bases de datos de código abierto con una versión comercial gestionada por Oracle.
- **Docker** [6]: es un software de código abierto que permite desplegar aplicaciones dentro de contenedores virtuales, que son entornos aislados y portables.
- **Maven** [7]: es una herramienta de código abierto que simplifica procesos de build. La gestión de dependencias se realiza a través del archivo de configuración de Maven llamado [Project Object Model \(POM\)](#).
- **IntelliJ Idea Ultimate** [8]: es un [IDE](#) desarrollado por JetBrains, que soporta varios lenguajes de programación.
- **Postman** [9]: es una aplicación que permite enviar peticiones HTTP a APIs REST y es útil para hacer pruebas sobre la API.
- **Apache Poi** [10]: es una librería de código abierto que permite a los programas Java leer y escribir archivos en formato Microsoft Office, como Excel, Word y PowerPoint.

- **Mapstruct** [11]: es una herramienta que permite crear mapeos entre objetos en Java mediante anotaciones.
- **Swagger UI** [12]: es una herramienta que te permite visualizar y probar las API. Con Swagger UI puedes explorar las operaciones, parámetros, respuestas y otros detalles de una API de forma interactiva y sencilla.

## 5.2 Tecnologías utilizadas en el *frontend*

- **Node.js** [13]: es un entorno en tiempo de ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en JavaScript.
- **Npm** [14]: es el gestor de paquetes de JavaScript que viene incluido con Node.js, un entorno de ejecución de JavaScript. NPM te permite instalar, compartir y administrar herramientas, módulos y dependencias de JavaScript.
- **JavaScript** [15]: es un lenguaje de programación interpretado, orientado a objetos, débilmente tipado y dinámico, utilizado en desarrollo web para mejorar la interfaz de usuario y en páginas web dinámicas.
- **Css**: son las siglas de Cascading Style Sheets, que significa hojas de estilo en cascada. Es un lenguaje de programación que se usa para definir el aspecto visual de las páginas web.
- **React** [16, 17]: es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones [Single Page Application \(SPA\)](#). Es mantenido por Facebook y la comunidad de software libre.
- **Redux** [18]: es un contenedor predecible del estado de aplicaciones JavaScript, utilizado para controlar el flujo de datos en una dirección.
- **Material-UI** [19]: es una librería de código abierto que provee componentes para utilizar con React, siguiendo principios de Material Design.
- **IntelliJ Idea Ultimate** [8]: el mismo [IDE](#) usado en el [backend](#).
- **Recharts** [20]: es una biblioteca de componentes de React que puede utilizar para crear varios tipos de tablas y gráficos. Está construido sobre D3. Todas las visualizaciones tienen un aspecto limpio y moderno.
- **React-pdf** [21]: es una biblioteca de React que permite mostrar archivos PDF como si fueran imágenes en una aplicación React sin problemas o usando varios plugins. Admite

una variedad de estilos y propiedades CSS, proporciona una API de nodo, es altamente personalizable y fácil de usar.

### 5.3 Tecnologías complementarias

- **Git** [22]: es un sistema de control de versiones distribuido utilizado para llevar registro de los cambios en los archivos de un proyecto y coordinar el trabajo de varias personas.

## Capítulo 6

# Desarrollo

---

En este capítulo, se presentarán de manera enumerada los elementos más destacados del proceso de desarrollo de la aplicación. Para comenzar, se describirá la estructura y el modelo de datos que fundamentan la aplicación. A continuación, se proporcionará un desglose detallado de las distintas iteraciones en las que se organizó el proyecto, incluyendo los objetivos alcanzados en cada fase.

### 6.1 Estructura de la aplicación

#### 6.1.1 Estructura del backend

El **backend** de la aplicación (ver 6.1) se encuentra conformado por los siguientes archivos y directorios:

- **/src/main/java**: Directorio que contiene el código fuente del **backend**. A su vez, se divide en los siguientes paquetes:
  - **model**: Este paquete contiene la estructura organizativa y clases relacionadas con las capas de acceso a datos y la lógica de negocio del **backend**. La capa de acceso a datos está compuesta por entidades y **DAO**, mientras que la capa de lógica de negocio incluye servicios, sus correspondientes excepciones y los mapeadores que transforman las entidades a **DTO**.
  - **rest**: En este paquete se encuentra la capa de servicios del **backend**. Incluye controladores **REST**, **DTO**, configuración de seguridad en Spring, y clases relacionadas con la generación y gestión de **JWT**.
  - **Application.java**: Es la clase raíz de la aplicación que utiliza Spring Boot, y es necesaria para arrancar esta.



- **/src/main/resources:** En este directorio se localizan los recursos necesarios para las clases Java del proyecto. También contiene archivos de internacionalización para mensajes generados por el **backend**, así como el archivo de configuración principal, en este caso, `application.yml`.
- **/src/test/java:** Esta carpeta engloba todas las clases de pruebas creadas para evaluar el código del **backend**.
- **/src/test/resources:** Aquí se almacenan los recursos necesarios para las pruebas del **backend**, así como la configuración correspondiente, contenida en el archivo `application.yml`.
- **pom.xml:** Dado que el proyecto utiliza Maven, este archivo **XML** alberga la configuración y las dependencias del proyecto, entre otras cosas.

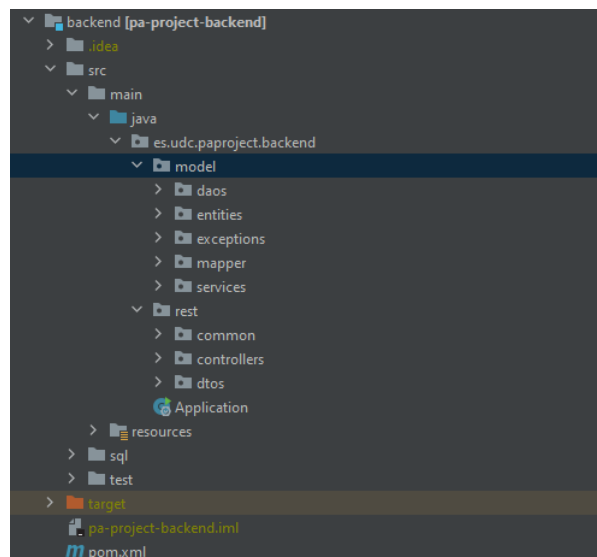


Figura 6.1: Estructura del backend.

### 6.1.2 Estructura del frontend

El **frontend** de la aplicación como se ilustra en 6.2 está compuesto por los siguientes archivos y directorios:

- **/public:** Directorio que contiene las imágenes que se muestran en la aplicación.
- **/src/backend:** Contiene la capa de acceso a servicios del **frontend**, realiza las peticiones **REST HTTP** al **backend**.

- **/src/modules:** En este directorio se alojan los diversos módulos que conforman la interfaz de usuario de la aplicación. Cada módulo sigue la siguiente estructura:
  - **/components:** Directorio que contiene los componentes del módulo concreto.
  - **actionTypes.js:** En este fichero se definen los tipos de acciones relacionadas con el módulo.
  - **actions.js:** Detalla como debe proceder el cliente para cada acción concreta.
  - **reducer.js:** Especifica cómo se gestionará el estado del módulo, incluyendo cómo cambiará el estado en función del tipo de acción activada.
  - **selector.js:** Contiene varias funciones encargadas de extraer los valores almacenados en el estado del módulo.
  - **index.js:** Define lo que el módulo exporta para que otros módulos puedan utilizarlo.
- **/src/i18n:** En este directorio se encuentran los archivos relacionados con la internacionalización de la aplicación.
- **/src/store:** Contiene el reductor raíz del cliente que combina los reductores de todos los módulos, es decir, conforma el estado global de la aplicación.
- **/src/index.js:** Renderiza la aplicación utilizando el componente App.js como base. También configura el store de Redux y la internacionalización.
- **package.json:** Este archivo incorpora información fundamental del proyecto, como dependencias, scripts y otros detalles.

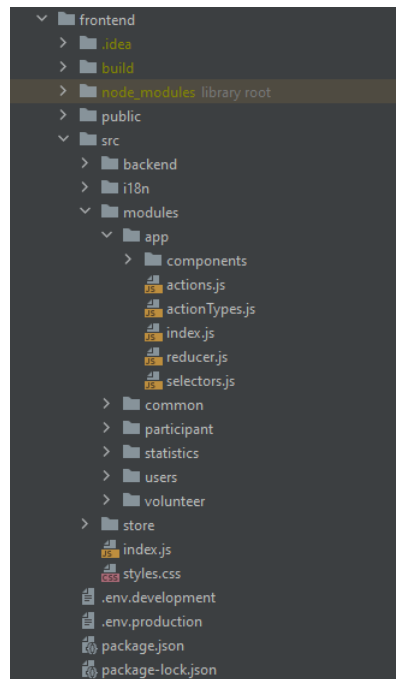


Figura 6.2: Estructura del frontend.

## 6.2 Modelo de datos

En las figuras 6.3 y 6.4 se presentan los diagramas de entidades de la aplicación. A continuación se explican de forma detallada el significado de las entidades y relaciones del modelo de datos.

Cada individuo que recibe asesoramiento de AMIGA es referido como un "participante". Para cada participante, se guardan una serie de datos básicos, como su nombre, apellidos, número de teléfono, entre otros. Estos datos se representan en la entidad denominada **Participant**. Además de estos datos fundamentales, para cada participante también se registra información adicional que debe registrarse anualmente, como los datos sobre el lugar de residencia, programas en los que participa, situación laboral, entre otros datos relevantes.

La representación de esta información adicional se lleva a cabo mediante la entidad **AnnualData**. Si un participante recibe asesoramiento de AMIGA durante un periodo de "n" años, se creará una instancia de la entidad **Participant** que contendrá sus datos básicos, junto con "n" instancias asociadas de **AnnualData**, cada una de ellas correspondiente a un año específico.

Es importante destacar que, en cada nuevo año en el que un participante desee continuar

recibiendo asesoramiento de AMIGA, se espera que visite las oficinas de la ONG para comunicar y actualizar sus datos. En su primera visita, se creará una instancia de **Participant** y otra de **AnnualData** (asociada a la primera) correspondiente al año en curso. En futuras visitas, si el participante decide volver a solicitar el asesoramiento de AMIGA, se procederá a actualizar posiblemente algunos de los datos en la entidad **Participant**, como el número de teléfono, por ejemplo. Además, se creará una nueva instancia de **AnnualData** para ese año específico, la cual contendrá los valores actualizados de los datos que deben mantenerse para ese periodo anual.

A continuación se describen las entidades que definen características de los participantes, se pueden dividir en dos grupos:

1. Entidades cuyo contenido se generó a partir de la información obtenida de documentos CSV. Para lograr esto, se utilizó un script en JavaScript que transformó los datos de estos archivos CSV en sentencias SQL de tipo "INSERT". Aquí se incluyen:
  - **Language:** Esta entidad modela un idioma que habla un participante. Los datos anuales de un participante puede reflejar que un participante habla 1 o varios idiomas.
  - **Country:** En esta entidad se almacenan los distintos países. Esta clase tiene una relación con la entidad "Participant" en la que se guarda el país de origen del participante, y otra relación con "AnnualData" en la que se registran las nacionalidades que posee el participante.
  - **Province:** Esta entidad modela una provincia en la cual se encuentra un municipio. Actualmente solo almacena las provincias gallegas.
  - **Municipality:** Modela cada uno de los municipios gallegos, y hace referencia a la provincia en la que se encuentran. Cada año se registra en que municipio esta domiciliado el participante.
2. Entidades cuyo contenido fue detallado por los responsables de AMIGA. En este grupo tenemos las siguientes entidades:
  - **Exclusion factor:** Las posibles limitaciones que pueda tener un participante, los datos anuales de un participante puede reflejar que un participante sufre de 1 o varios de estos factores.
  - **Program:** Recoge los posibles programas en los que se puede participar. Se creó una entidad intermedia entre "Program" y "AnnualData" debido a que cada participante puede estar apuntado en más de un programa y para cada uno necesitan

saber si en itinerario o no, lo que equivale a determinar si se le está haciendo un seguimiento explícito en ese programa.

- **Study:** Los diferentes niveles de estudios.
- **MaritalStatus:** Los posibles estados civiles.
- **Employment:** Las diferentes situaciones laborales.
- **Demand:** Tipos de demanda.
- **Housing:** Tipos de vivienda.
- **Cohabitation:** Tipos de convivencia familiar.
- **Contract:** Tipos de contrato, esta entidad es referenciada en la entidad que modela las inserciones laborales.

La decisión de implementar estos datos en forma de las anteriormente entidades se basa en la facilidad para la introducción de nuevos campos en los desplegables sin necesidad de modificar el código. Esto significa que si en el futuro se necesita agregar nuevos elementos a los desplegables, se puede hacer directamente a través de operaciones de inserción en la base de datos, sin necesidad de realizar cambios en el código.

Además, como se puede observar en los diagramas de entidades, se han representado algunos campos con opciones limitadas como enumerados. Estos datos, como el género o la aprobación de estudios, se indicó que no cambiarán en el futuro. En todos estos casos, se ha optado por utilizar la notación `@Enumerated(EnumType.STRING)`, lo que implicará que en la tabla asociada a la entidad correspondiente se guarde el valor del enumerado.

El resto de entidades que se incluyen en el modelado son las siguientes:

- **Kid:** En esta entidad se guarda la fecha de nacimiento y el sexo de los menores a cargo del participante.
- **Observation:** En esta entidad se almacenan los datos relacionados con las atenciones que se realizan sobre los participantes.
- **WorkInsertion:** Guarda la información relativa a las inserciones laborales de los participantes atendidos.
- **User:** En esta entidad se almacenan los datos relacionados con los trabajadores que utilizarán la aplicación, como es el rol y los datos necesarios para iniciar sesión.
- **Volunteer:** Guarda la información básica de los voluntarios que colaboran con AMIGA.

- **Collaboration:** En esta entidad se almacenan los datos relacionados con las actividades llevadas a cabo por los voluntarios.

Este proceso de diseño y modelado de datos demandó una gran cantidad de tiempo, con varias reuniones destinadas a clarificar la gran cantidad de información que debía ser registrada.

El modelado de datos desempeña un papel crucial en la aplicación, ya que establece la base para la extracción de estadísticas. Además, su importancia se ve aumentada ante la posibilidad de futuras expansiones del aplicativo, ya que es necesaria una estructura escalable que pueda adaptarse a nuevas funcionalidades o requisitos cambiantes.

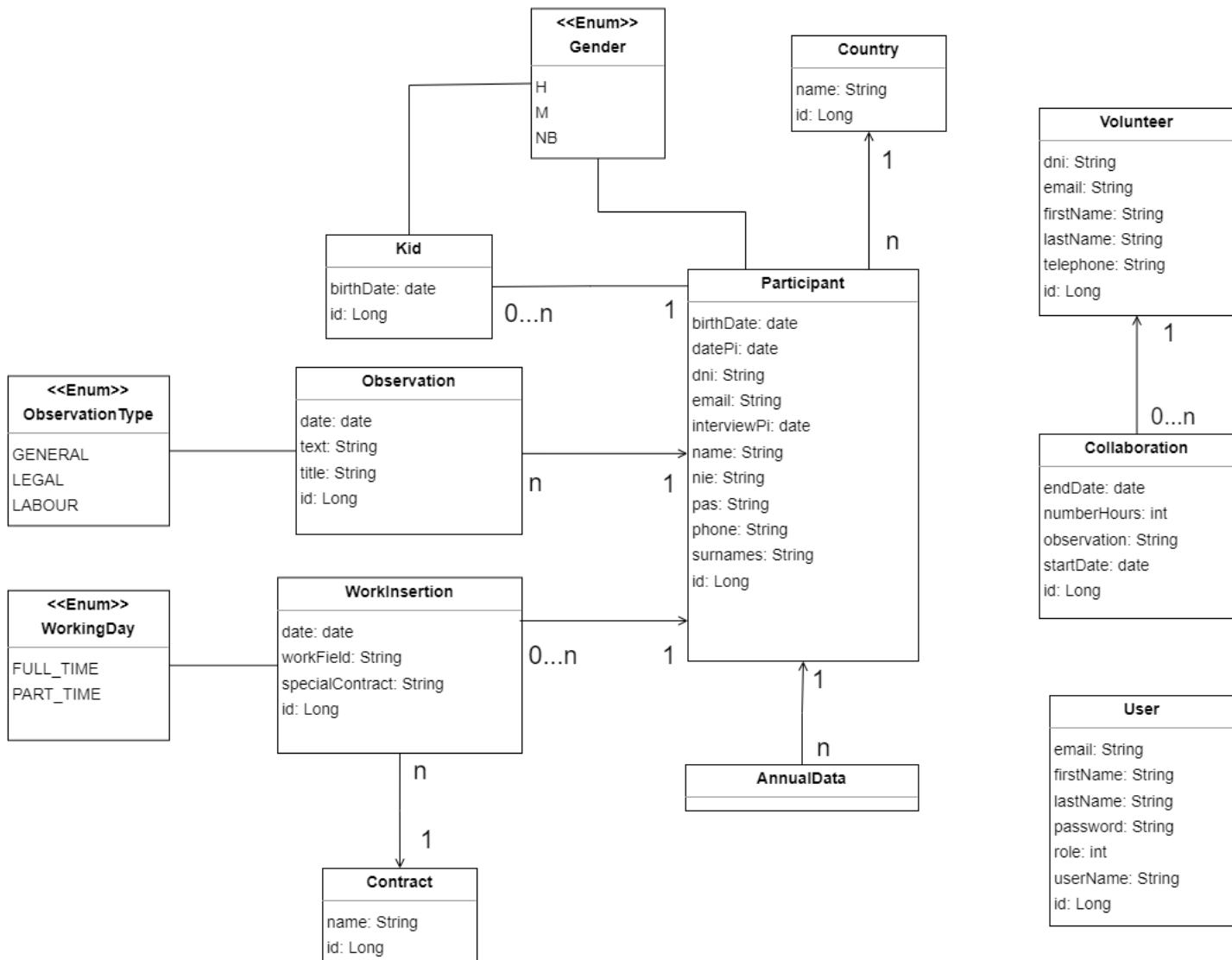


Figura 6.3: Primera parte diagrama de entidades.

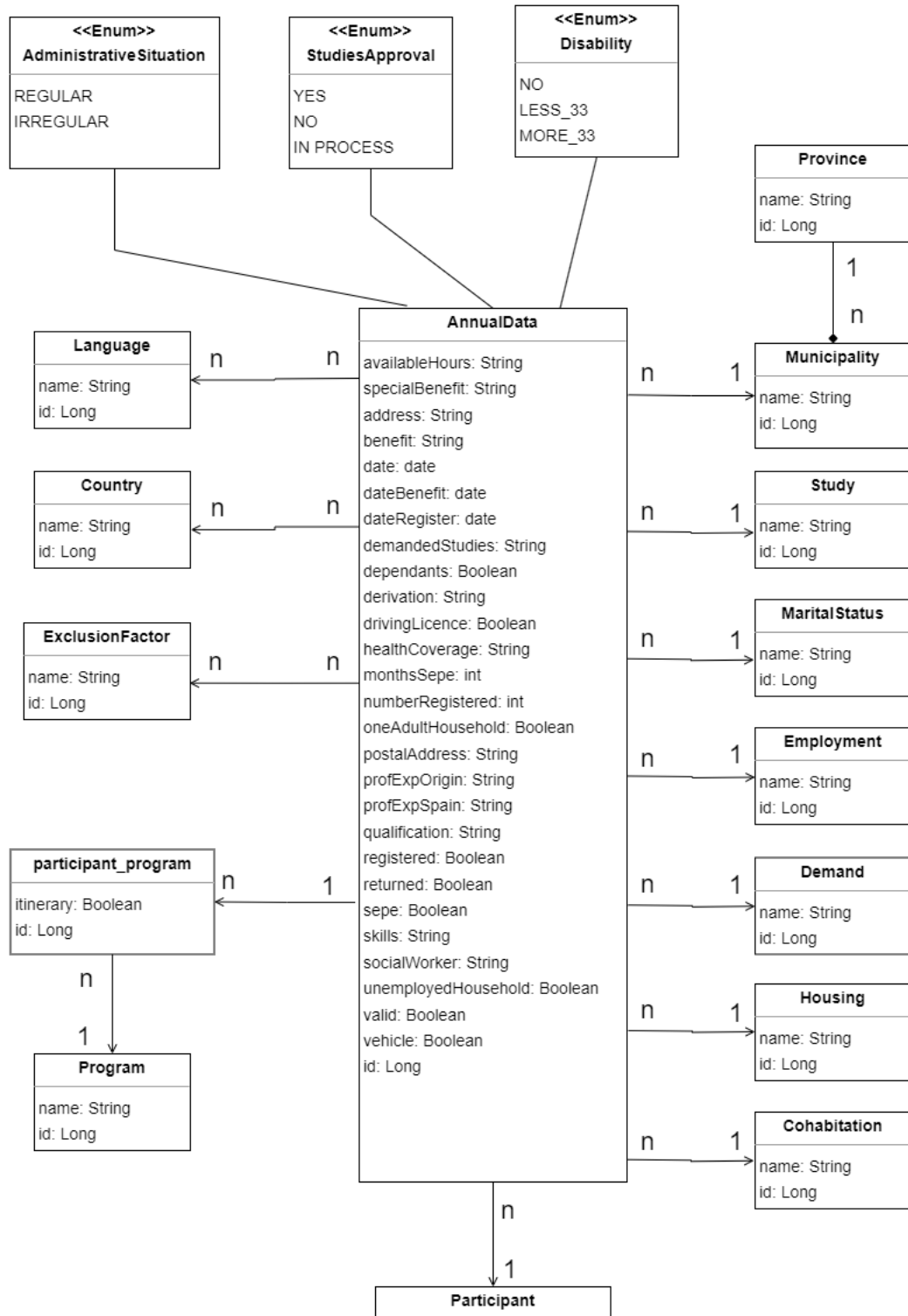


Figura 6.4: Segunda parte diagrama de entidades.



## 6.3 Iteración 1

### 6.3.1 Análisis

Los objetivos de esta iteración fueron:

- **Crear el esqueleto del proyecto**, abarcando tanto el **backend** como del **frontend**.
- **Configurar los contenedores de Docker** para instanciar bases de datos MySQL, estableciendo una para pruebas y otra para desarrollo.
- **Generar los datos que se encuentran predefinidos en la base de datos desde el inicio**, es decir, los datos de los campos con opciones limitadas que se recopilan de los participantes. Se destaca la diferenciación entre los casos en los que el cliente proporcionaba las distintas posibilidades y los casos que como países, idiomas y municipios, fueron generados mediante scripts que realizaban lecturas de la información de archivos CSV a partir de los cuales creaban las sentencias de inserción a la base de datos.
- **Llevar a cabo el proceso de aprendizaje de React-pdf** para la generación de documentos directamente desde el frontend.
- **Implementar los casos de uso relacionados con los datos de los participantes**, los cuales incluyen:
  - CU-01 Acogida participante.
  - CU-02 Registrar nuevos datos participante.
  - CU-03 Búsqueda participante por documento identificativo.
  - CU-04 Visualizar detalles de participante para un año determinado.
  - CU-05 Editar datos del participante para un año específico.

### 6.3.2 Diseño e implementación

Durante las sesiones de reuniones destinadas a establecer el modelado de datos, simultáneamente se inició la formación en React-pdf, siguiendo la guía proporcionada en la página oficial. Así mismo, se creó un proyecto de práctica en un entorno local con el fin de entender correctamente las funcionalidades de esta herramienta.

Una vez concluido el modelado de datos de esta fase se procedió a la creación de los esqueletos del **backend** y **frontend**. Posteriormente, se realizó la configuración del backend para realizar la conexión con los contenedores Docker previamente creados. Estos contenedores albergan las tablas correspondientes a las clases requeridas en esta fase, y se llenaron con los

datos predefinidos que se mencionaron anteriormente. En la figura 6.5 se muestra el diagrama de clases realizado en esta fase.

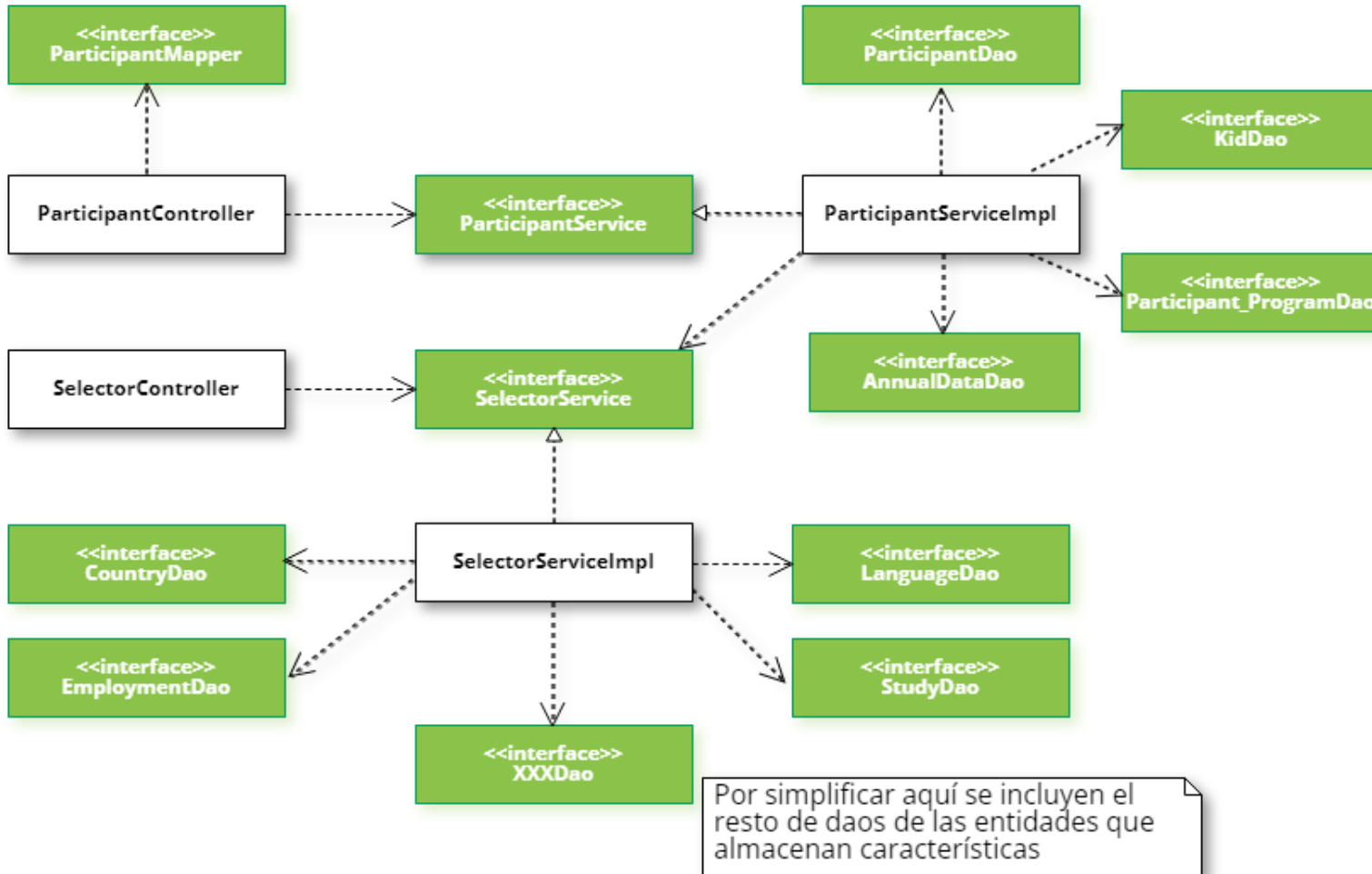


Figura 6.5: Diagrama clases implementadas iteración 1.

Finalmente se llevaron a cabo las implementaciones de los casos de uso previamente mencionados:

#### CU-01 Acogida participante:

Este resultó ser el caso de uso de mayor complejidad debido a la gran cantidad de datos que se manejan, tal como se mencionó en la sección 6.2. El primer paso consistió en crear las numerosas entidades relacionadas con la información del participante. El siguiente paso consistió en definir sus respectivos DAO y el servicio con la lógica de negocio correspondiente. Una vez

completada esta fase, se procedió a la creación del controlador REST relacionado con los participantes, junto con la creación del DTO de "Participant," que combina los datos almacenados en las entidades "Participant" y "AnnualData" (consultar la sección 6.2).

Una vez completada la implementación, se llevó a cabo una fase de pruebas utilizando la herramienta Postman para verificar su correcto funcionamiento.

En el [frontend](#), se creó un "participantService" dentro de la carpeta de backend (ver sección 6.1.2), el cual contiene la petición "fetch" encargada de comunicarse con el [backend](#). Posteriormente, se desarrolló el módulo "Participante," siguiendo la estructura mencionada previamente. Para dar inicio a la ejecución de este caso de uso, se añadió un botón que activa la navegación hacia el componente principal de este caso de uso. Dicho componente engloba a los componentes hijos, que representan cada página del formulario. Este componente tiene la responsabilidad de definir la navegación entre las diferentes páginas y también de gestionar la función a ejecutar cuando se completa y envía el formulario. Esta decisión de dividirlo en páginas con contenido de naturaleza similar fue tomada en colaboración con el cliente, ya que presentar todo el formulario en una única página habría resultado muy abrumador.

En cada una de las páginas, se implementa una validación de los campos antes de permitir avanzar o retroceder. Si algún campo no cumple con los requisitos, se muestra un mensaje de error indicando el motivo específico. En esta situación particular, además de verificar la presencia de campos obligatorios, se realiza una validación específica para los formatos de DNI y NIE.

En la primera página del componente, se presentan los campos para rellenar los datos personales básicos, como se puede observar en la imagen 1.

En la segunda página 2, se muestran otros campos adicionales para completar la información sobre datos personales de los participantes. Entre ellos se encuentran por ejemplo los menores a cargo. Esta funcionalidad se ha implementado en un componente separado, aprovechando uno de los beneficios clave de la arquitectura basada en componentes que React proporciona. En este contexto, se han utilizado varios componentes individuales, lo que destaca uno de los aspectos más valiosos de la modularidad en React. En este caso de uso en específico, se han utilizado 13 componentes distintos para lograr la funcionalidad completa.

Este enfoque modular y basado en componentes no solo simplifica la estructura y la gestión del código, sino que también facilita la reutilización y el mantenimiento a medida que el proyecto evoluciona. Cada componente puede encargarse de una tarea específica, lo que

mejora la claridad y la organización del código en general.

En la tercera página 3, se muestran campos relativos a la experiencia laboral y formación, mientras que en la cuarta [4] se recogen los datos relativos al tipo de demanda y programas en los que participa el participante.

Por último, se presenta la página en la cual se genera un archivo PDF que el trabajador puede descargar y presentar para su firma 6.6. En este paso, se brinda la posibilidad de visualizar el informe antes de proceder a la descarga, lo que permite verificar que los datos registrados sean correctos. Una vez que el participante ha firmado el documento, el técnico puede hacer clic en el botón de "Guardar". En este momento, los datos recogidos en el formulario se almacenan de manera permanente, quedando registrados en el sistema. Este proceso garantiza el cumplimiento de la ley de protección de datos, al tiempo que facilita la generación de documentos y la gestión de la firma de los trabajadores.

Descargar el documento a firmar y guardar datos

ANTERIOR
DESCARGAR PDF
GUARDAR



**AMIGA**  
Asociación de Migrantes de Galicia

**Ficha de acogida**

Fecha: 11-07-2023

NOMBRE Y APELLIDOS: Miguel Crespo Rozados		DNI: 54153223E	NIE: NO
FEC. DE NACIMIENTO: 02-06-2023	SEXO: H	PAS: NO	
DOMICILIO: Plaza Cubela		SITUACIÓN ADMINISTRATIVA: REGULAR	
PROVINCIA: A Coruña	MUNICIPIO: Abegondo	PAÍS DE ORIGEN: Alemania	
C.P.: 15008		NACIONALIDAD/ES: Antigua y Barbuda	
TELÉFONO/S: 684350303		EMPADRONADO/A: SI	
EMAIL: crespo.rozados@gmail.com		FECHA: 01-06-2023	

Figura 6.6: Confirmación formulario acogida.

Otro aspecto significativo de este caso de uso es la implementación del almacenamiento en Redux de los datos de las listas desplegadas al momento de iniciar la aplicación. Esta característica garantiza que las solicitudes para obtener esta información se realicen únicamente en la primera ocasión de apertura de la aplicación. Como resultado, se mejora la eficiencia y la velocidad de carga de la aplicación en sucesivas operaciones.

**CU-02 Búsqueda participante por documento identificativo:**

Esta funcionalidad facilita la búsqueda de información básica de un participante. Permite seleccionar el tipo de documento por el cual se desea realizar la búsqueda y proporcionar el número correspondiente 6.7.

El desarrollo de esta funcionalidad comenzó en la capa del backend, donde se efectúa una consulta a la clase "Participant" para obtener su nombre, apellidos, correo electrónico y número de teléfono. La razón de mostrar estos datos es proporcionar los detalles de contacto del participante sin requerir acceder a su perfil completo. Para este propósito, se creó un nuevo DTO con solo los datos resumidos necesarios para la visualización.

En el frontend, la implementación consistió en un menú desplegable que permite seleccionar entre tres tipos de documentos: DNI, NIE y PAS. Además, se incluyó un campo de entrada para introducir el número de documento correspondiente. En caso de seleccionar DNI o NIE, se realiza una comprobación del formato del documento antes de enviar la solicitud, mostrando un mensaje de error si el formato no es válido. Debido a que existen varios formatos posibles para el PAS, no es posible realizar esta comprobación. Una vez que se envía la solicitud, si hay coincidencias, se muestran los datos mencionados previamente. En caso de que no exista ningún participante con el documento proporcionado, se muestra un mensaje de error. En la figura 6.7 se muestra el resultado de esta implementación.

Esta funcionalidad proporciona una manera rápida y eficiente de obtener información esencial sobre un participante sin necesidad de acceder a detalles más extensos, lo cual puede ser especialmente útil para tareas de contacto.

Seleccionar filtro Documento \*

DNI 54153223E

BUSCAR

Miguel Crespo Rozados

Email: crespo.rozados@gmail.com

Teléfono: 684350303

VER DETALLES

Figura 6.7: Búsqueda por documento identificativo.

**CU-03 Registrar datos año actual participante:**

Si el participante no dispone de un registro correspondiente al año actual, el sistema posibilita llevar a cabo un nuevo registro con el fin de establecer una trazabilidad continua de sus datos a lo largo de los diferentes años.

En esta funcionalidad, se incorporó una lista de años a la clase "Participante", la cual contiene los registros de años en los que el participante tiene historial en la organización. Esto se hizo con el propósito de facilitar la presentación de informes anuales, ya que permite conocer de antemano qué años están disponibles para visualización sin necesidad de consultar todos los registros de "AnnualData" en la base de datos del participante. Esta mejora tiene un impacto positivo en el rendimiento, ya que solo se consultan los datos anuales cuando sea necesario.

En el backend, se implementó otro endpoint en el controlador y se agregó una función adicional en el servicio dedicado a los participantes. La lógica detrás de negocio es la siguiente: si se ingresan datos distintos en relación a la clase "Participante", estos datos se actualizan. Luego, se agrega el nuevo año a la lista de años atendidos y se crea un nuevo registro de datos anuales.

Este enfoque garantiza que se optimiza el rendimiento al evitar consultas innecesarias a la base de datos para años sin registros relevantes. De esta manera, se logra una gestión eficiente de los datos anuales y se mejora la eficacia en la generación de informes.

En el frontend, si el participante no se encuentra registrado en el año actual (figura 6.8), la interfaz mostrará una alerta para informar al usuario y le solicitará que seleccione el banner para crear un nuevo registro. Esta información se obtiene haciendo uso de la lista de años anteriormente mencionada. El motivo de este enfoque radica en que es una operación que se realiza una vez al año, por lo que carece de sentido incluirla en el menú de acciones relacionadas con el participante de forma permanente.

Además se ha logrado una reutilización efectiva de los componentes utilizados para registrar a un participante en el formulario. La única variación radica en que el formulario se completa automáticamente con la información correspondiente al último año registrado. Para lograr esto, se implementó un nuevo componente principal que carga los datos iniciales y define el nuevo endpoint que debe invocarse en el componente de confirmación del formulario.

Esta estrategia de reutilización demuestra un enfoque inteligente para optimizar el desarrollo y la mantenibilidad del código. Al aprovechar los componentes ya existentes y agregar

solo las modificaciones necesarias, se ahorra tiempo y esfuerzo, al tiempo que se mantiene la consistencia en la interfaz de usuario. Además, el hecho de cargar automáticamente los datos del último año registrado agiliza el proceso ya que mejora la experiencia del usuario al minimizar la necesidad de introducir datos repetitivos.

Figura 6.8: Registro datos año actual participante.

#### CU-04 Visualizar datos de participante para un año determinado:

El sistema genera un informe en formato PDF con los datos relativos al participante en el año seleccionado, para ello en el backend se crea un nuevo endpoint que solicita id del participante y el año del que se quiere obtener el participante. El DTO que se devuelve en la petición resulta de ser la combinación entre los datos de Participant y los del datos anuales del año seleccionado.

Para llevar a cabo la transición entre estas dos clases y el DTO, se empleó MapStruct, una herramienta que simplifica el proceso de mapeo entre clases. Sin embargo, a pesar de la ayuda proporcionada por MapStruct, el código resultante sigue siendo de una cierta complejidad debido a las transformaciones de atributos requeridas.

El uso de la anotación `@Mapping` se utiliza en aquellos atributos que presentan un formato distinto y necesitan ser convertidos. Por ejemplo, en lugar de devolver las entidades completas para atributos como "country", el DTO solo incluye el identificador ("id"). Por otro lado, los mapeos que utilizan la anotación `qualifiedByName` denotan la aplicación de una función personalizada con ese nombre específico, la cual se encarga de realizar la conversión requerida.

```

1  @Mapping(source = "p.gender", target = "sex", qualifiedByName =
    "toSex")
2  @Mapping(source = "p.country.id", target = "country")
3  @Mapping(source = "p.id", target = "idParticipant")
4  @Mapping(source = "a.id", target = "idAnnualData")
5  @Mapping(source = "a.nationalities", target = "nationalities",
    qualifiedByName = "toNationalities")

```

```
6      @Mapping(source = "a.municipality.id", target = "municipality")
7      @Mapping(source = "a.province.id", target = "province")
8      @Mapping(source = "a.situation", target = "situation",
9              qualifiedByName = "toSituation")
10     @Mapping(source = "a.studies.id", target = "studies")
11     @Mapping(source = "a.languages", target = "languages",
12             qualifiedByName = "toLanguages")
13     @Mapping(source = "a.cohabitation.id", target = "cohabitation")
14     @Mapping(source = "a.maritalStatus.id", target =
15             "maritalStatus")
16     @Mapping(source = "a.housing.id", target = "housing")
17     @Mapping(source = "a.exclusionFactors", target =
18             "exclusionFactors", qualifiedByName = "toExclusionFactors")
19     @Mapping(source = "a.approved", target = "approved",
20             qualifiedByName = "toApproved")
21     @Mapping(source = "a.disability", target = "disability",
22             qualifiedByName = "toDisability")
23     @Mapping(source = "a.employment.id", target = "employment")
24     @Mapping(source = "a.benefit", target = "benefit",
25             qualifiedByName = "toBenefit")
26     @Mapping(source = "a.demand.id", target = "demand")
27     ParticipantDto toParticipantDto(Participant p, AnnualData a);
```

En el frontend, se ha implementado de manera sencilla un elemento seleccionable que utiliza la lista de años asociados al participante para mostrar las distintas opciones disponibles. Al seleccionar un año en particular, se muestra el informe del participante correspondiente a ese año en formato PDF. Este informe es el mismo al generado al finalizar el formulario. En la figura 6.9 se muestra esta funcionalidad.


Esta implementación refleja una solución eficiente para permitir a los usuarios acceder a los informes de los participantes de años anteriores. Al aprovechar la lista de años ya almacenada y reutilizar el formato PDF generado previamente, se optimiza la experiencia del usuario y se simplifica la navegación y visualización de los informes.



VOLVER
EDITAR

Datos Name5 Surnames5
 

Seleccionar año  
2023



**AMIGA**  
Asociación de Migrantes de Galicia

Ficha de acogida

Fecha: 03-09-2023

NOMBRE Y APELLIDOS: Name5 Surnames5		DNI: NO      NIE: NO	
FEC. DE NACIMIENTO: 01-01-1990      SEXO: H		PAS: p5	
DOMICILIO: Address		SITUACIÓN ADMINISTRATIVA: IRREGULAR	
PROVINCIA: Pontevedra	MUNICIPIO: Cambre	PAÍS DE ORIGEN: Cuba	
C.P.: Postal Address		NACIONALIDAD/ES: Arabia Saudita	
TELÉFONO/C: 123456789			

Figura 6.9: Visualizar datos de participante.

#### CU-05 Editar datos del participante para un año específico:

El sistema ofrece la capacidad de modificar los datos de cualquier año registrado. Para lograrlo, se inició la implementación de esta funcionalidad en el backend. La complejidad de este caso de uso radicaba en la manipulación de las listas de menores a cargo y de programas en los que el participante está inscrito. El desafío consistía en determinar si se habían eliminado o agregado elementos en estas listas, ya que se trataba de una lista de objetos el atributo que se recibía.

La solución adoptada para abordar este desafío se basó en la ventaja de que los objetos previamente creados ya contaban con un campo "id", mientras que los nuevos objetos carecían de este campo. Se aprovechó esta distinción para discernir si un objeto debía ser añadido o eliminado. La lógica implementada fue la siguiente:

- **Objetos Nuevos::** Si un objeto no tenía un "id", se interpretaba como un nuevo elemento y se guardaba en la base de datos.
- **Objetos Existente:** Para los objetos con "id", se realizaba una comparación con la lista original de elementos antes de la edición. Si un objeto no se encontraba en la nueva lista proporcionada, se entendía que había sido eliminado.

En el frontend, la implementación resultó ser bastante sencilla. Se creó un nuevo componente principal que se encargaba de cargar los datos en el formulario utilizado en casos de uso anteriores con la información específica de ese año. Además, se realizó la petición correspondiente al enviar el formulario.

## 6.4 Iteración 2

### 6.4.1 Análisis

Durante esta fase del desarrollo, se completaron las implementaciones restantes relacionadas con la gestión completa del ciclo de vida del participante. Estas implementaciones abarcaron dos aspectos fundamentales: la administración de las atenciones brindadas a los participantes en su rutina diaria y el registro de las inserciones laborales que lograron obtener con el apoyo proporcionado.

Estas funcionalidades añaden un nivel de detalle y seguimiento más completo al sistema. La gestión de atenciones ofrece una visión detallada de las interacciones y asistencias diarias proporcionadas a los participantes, mientras que el registro de inserciones laborales proporciona una forma de documentar el desarrollo profesional de los participantes. Los casos de uso implementados en esta fase fueron los siguientes:

- CU-06 Crear atención.
- CU-07 Editar atención.
- CU-08 Eliminar atención.
- CU-09 Buscar atenciones mediante filtros.
- CU-10 Crear inserción laboral.
- CU-11 Visualizar inserciones laborales.
- CU-12 Editar inserción laboral.
- CU-13 Eliminar inserción laboral.

### 6.4.2 Diseño e implementación

En esta etapa del desarrollo, se introdujeron las entidades "Observation" para almacenar información relacionada con las atenciones brindadas a los participantes, así como "WorkInsertion" para registrar los detalles acerca de las inserciones laborales que han logrado los participantes bajo su asistencia.

Para gestionar estas dos nuevas clases, se establecieron sus propios servicios y controladores. Esto garantiza una organización eficiente y una estructura coherente en el código, al mismo tiempo que permite un enfoque especializado en la gestión de atenciones y las inserciones laborales. Este enfoque modular y especializado mejora la claridad y mantenibilidad

del sistema en su conjunto, permitiendo un manejo eficaz de cada aspecto particular del ciclo de vida del participante.

### CU-06 Crear atención, CU-07 Editar atención, CU-08 Eliminar atención, CU-09 Ver detalles atención

En conjunto, los casos de uso ofrecen las funcionalidades CRUD (Crear, Leer, Actualizar y Eliminar) en relación a las atenciones brindadas a los participantes. La información guardada para cada atención incluye los siguientes datos:

- **Fecha:** Indica la fecha en la que se llevó a cabo la atención.
- **Tipo de Atención:** Puede ser categorizada como General, Laboral o Jurídica.
- **Título:** Proporciona un resumen de la atención realizada.
- **Descripción:** Ofrece un campo de texto donde se detalla lo que se llevó a cabo durante la atención en cuestión.

En la capa del backend, se ha desarrollado un servicio y un controlador específicos para llevar a cabo estas tareas. Por otro lado, en la interfaz del usuario, el proceso de registro se realiza a través de un formulario que requiere de que se inserten todos los datos previamente mencionados (figura 6.10).

El formulario se encuentra en la parte superior de la interfaz, con el título "Nueva atención Name1 Surnames1" a la izquierda y un botón "CANCELAR" a la derecha. El formulario está dividido en dos secciones principales. La primera sección contiene tres campos: "Fecha \*" con el valor "26/08/2023", un campo "Tipo" que muestra un desplegable con las opciones "General", "Jurídico" y "Laboral", y un campo "Título \*". La segunda sección contiene un campo "Texto \*" para la descripción. En la parte inferior del formulario, hay un botón "ENVIAR".

Figura 6.10: Crear atención.

Una vez realizado el proceso de ver los datos de una atención, el sistema ofrece la posibilidad de ejecutar las acciones de edición y eliminación. En el caso de elegir la opción de eliminación, se despliega una modal solicitando la confirmación de la acción. En el escenario de edición, se presenta un desplegable que habilita la modificación de los campos que componen la atención.

Estas funcionalidades permiten registrar y gestionar todas las atenciones proporcionadas a los participantes. A través de la fecha y tipo de atención, el sistema proporciona un mecanismo para buscar y administrar el historial de atenciones, lo que resulta de gran utilidad para la gestión y seguimiento de los participantes en el programa.

[CANCELAR](#)

Fecha: 20-08-2023

Tipo: General

Título: Título atención

Texto atención

[EDITAR](#)[ELIMINAR](#)

Figura 6.11: Ver atención.

**CU-10 Buscar atenciones mediante filtros:**

Tal como se mencionó previamente, el sistema ofrece la funcionalidad de búsqueda paginada basada en el tipo de atención y un rango de fechas que abarca la fecha de inicio, fecha final o un intervalo de fechas. Para llevar a cabo esta implementación, se requirió la creación de consultas personalizadas en el DAO de las atenciones. Estas consultas personalizadas se crean utilizando la anotación `@Query`, como se ilustra a continuación:

```
1      @Query("SELECT o FROM Observation o
2      WHERE o.date >= COALESCE(:startDate, o.date)
3      AND o.date <= COALESCE(:endDate, o.date)
4      AND o.participant = :participant
5      ORDER BY o.date DESC")
6      Slice<Observation> findByDate(
7      @Param("startDate") LocalDate startDate,
8      @Param("endDate") LocalDate endDate,
9      @Param("participant") Participant participant,
10     Pageable pageable);
11
12     @Query("SELECT o FROM Observation o
13     WHERE o.observationType IN :types
14     AND o.date >= COALESCE(:startDate, o.date)
15     AND o.date <= COALESCE(:endDate, o.date)
16     AND o.participant = :participant
17     ORDER BY o.date DESC")
18     Slice<Observation> findByTypeAndDate(
19     @Param("types") List<ObservationType> types,
20     @Param("startDate") LocalDate startDate,
21     @Param("endDate") LocalDate endDate,
22     @Param("participant") Participant participant,
23     Pageable pageable);
```

La función `COALESCE` en SQL se emplea para recuperar el primer valor no nulo de una serie de expresiones. En este caso, se utiliza para manejar la situación en la que una fecha puede estar a nulo. En dicho caso, se recurre a la fecha correspondiente a la atención en sí misma, evitando así que esta condición afecte a la petición. En el servicio, la elección entre las dos consultas a la base de datos se determina en función de si se proporciona o no la lista de `ObservationType`.

Se optó por estructurar las páginas con un conjunto de cinco elementos, ya que una cantidad mayor dificultaría su visualización de manera rápida. Cada elemento está formado solamente por la fecha en que fue realizada y su título, permitiendo seleccionar cada uno ejecutando CU-09 Ver detalles atención. En cada búsqueda, las atenciones son cargadas según lo

recientes que se realizaron. En cuanto a la aplicación de filtros, se han dispuesto dos inputs destinados a ingresar la fecha de inicio y finalización respectivamente. Además, se ha incorporado un tercer campo en forma de lista desplegable, que permite la opción de seleccionar uno o varios tipos de atención que se desean obtener.

The screenshot shows a web interface for viewing attention records. At the top left, a red arrow labeled 'Menu' points to a hamburger menu icon. Next to it is the title 'Listado atenciones Miguel'. An orange arrow labeled 'Observations' points to the list of records. To the right, there are three filter fields: 'Fecha inicio' (dd/mm/aaaa), 'Fecha fin' (dd/mm/aaaa), and 'Tipo' (Juridico). A blue 'BUSCAR' button is on the far right. The main area contains a list of five records, each in a yellow box. The last record, '12-08-2023 Obtención papeles', is highlighted with a purple border. At the bottom, there is a pagination section with 'Anterior', 'Siguiente', and 'Pager' buttons, and an 'ObservationItem' label with an upward arrow.

Fecha	Descripción
26-06-2020	Primera observación revisada
26-04-2020	Primera revision
26-01-2020	Segunda revision
10-08-2023	Rutinaria
12-08-2023	Obtención papeles

Figura 6.12: Búsqueda de atenciones.

### CU-12 Visualizar inserciones laborales:

Dado que en este caso se prevé tener pocas inserciones por participante, se optó por mostrar todos los datos de las inserciones en una tabla, ordenados según la fecha más reciente. A partir de esta tabla, los usuarios tendrán la opción de realizar modificaciones o eliminar una inserción específica de manera eficiente.

### CU-11 Crear inserción laboral, CU-13 Editar inserción laboral, CU-14 Eliminar inserción laboral:

En conjunto, los casos de uso ofrecen las funcionalidades de registrar, editar y eliminar información sobre las inserciones laborales realizadas por los participantes, incluyendo los si-

## Inserciones laborales Name1 Surnames1

CANCELAR

Fecha	Tipo contrato	Sector	Jornada	Acciones
16-08-2023	Contrato Indefinido	Agricultura	Completa	<a href="#">EDITAR</a> <a href="#">ELIMINAR</a>
20-08-2023	Contrato Formativo para la Obtención de la Práctica Profesional	Sector para participante 1	Parcial	<a href="#">EDITAR</a> <a href="#">ELIMINAR</a>
<a href="#">NUEVA INSERCIÓN</a>				

Figura 6.13: Visualizar inserciones laborales.

guientes datos:

- **Fecha:** Indica la fecha en la que se llevó a cabo la inserción.
- **Tipo de contrato:** Que son opciones almacenados en la entidad Contract, en caso de escoger el tipo de contrato "otros" se habilita un campo para introducirlo manualmente.
- **Sector:** Texto libre a introducir por el trabajador.
- **Tipo de jornada:** Parcial o completa.

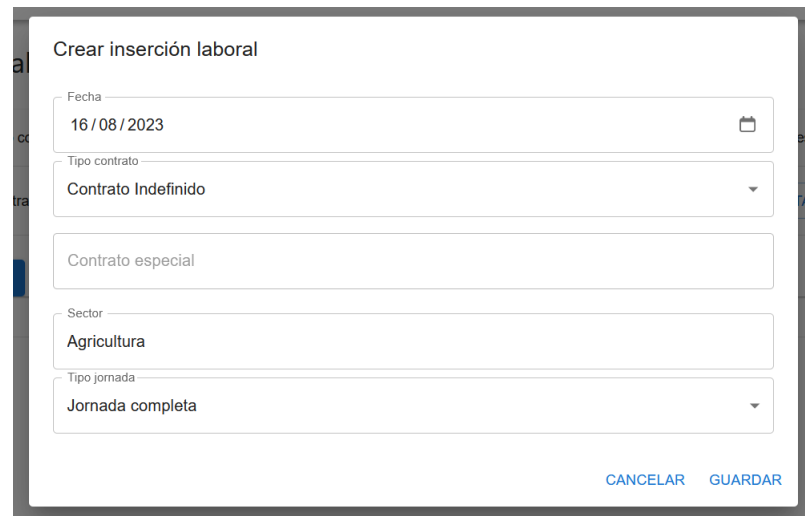
Siguiendo el mismo enfoque seguido para las atenciones, se ha procedido a crear en el backend el correspondiente servicio y controlador para implementar estas funcionalidades. En el frontend se ofrece la posibilidad de editar y de eliminar cada inserción mostrada en la tabla del caso de uso **Visualizar inserciones laborales**. Siguiendo una metodología similar a la aplicada en las atenciones, cada una de estas operaciones desencadena la apertura de una modal específica que facilita el cumplimiento de la funcionalidad requerida.

Este enfoque fue adoptado con el objetivo de evitar repeticiones constantes de redireccionamientos entre distintas páginas, las cuales podrían generar una experiencia menos satisfactoria para el usuario.

## 6.5 Iteración 3

### 6.5.1 Análisis

Durante esta fase, se abordó la tarea de generar estadísticas, la cual presentó un desafío considerable debido a la necesidad de aprender a utilizar Apache POI, una herramienta de



Crear inserción laboral

Fecha  
16/08/2023

Tipo contrato  
Contrato Indefinido

Contrato especial

Sector  
Agricultura

Tipo jornada  
Jornada completa

CANCELAR GUARDAR

Figura 6.14: Registrar inserciones laborales.

Java para la generación de archivos Excel, y Recharts, una biblioteca para crear gráficos en el frontend.

La generación de estas estadísticas surgió como uno de los pilares fundamentales del desarrollo de la aplicación, lo cual condujo a la necesidad de un enfoque especialmente minucioso en el diseño de la estructura de datos. Antes de la existencia de esta herramienta, los trabajadores se veían obligados a escribir manualmente los datos recopilados de los participantes en una hoja de cálculo Excel, a partir de la cual creaban los gráficos necesarios para elaborar los informes de actividad que se deben enviar a la Xunta. Gracias a la implementación de esta aplicación, dicho archivo Excel se genera automáticamente ahorrando una inmensa cantidad de tiempo. Además también crea los gráficos más frecuentemente requeridos. La confección de estas estadísticas constituyó uno de los factores determinantes que impulsaron la necesidad de un refinamiento preciso en la modelación de datos.

Para utilizar esta funcionalidad, el usuario debe ingresar una fecha de inicio y una fecha de fin. Estas fechas determinarán qué participantes se incluirán en la estadística. Solo se tomarán en cuenta aquellas personas que hayan recibido al menos una atención durante ese período.

### 6.5.2 Diseño e implementación

Para desarrollar este caso de uso, se nos presentaron las estadísticas y se nos mostró el archivo Excel que se utilizaba actualmente. Esto nos brindó una comprensión clara de sus necesidades. Generaban un archivo Excel en el que cada fila representaba los datos de un participante. Para obtener este documento se implementó un proceso iterativo que recorría



cada participante y asignaba los datos correspondientes a las celdas adecuadas.

Sin embargo, surgieron desafíos al querer obtener estadísticas basadas en campos en los que un participante podía tener múltiples valores. Para abordar este problema, se diseñó una solución que consiste en la creación de dos páginas dentro del archivo Excel. En la primera página, se visualizan las filas de los participantes. Mientras que en la segunda página, se recopilan datos numéricos específicos, los cuales se agrupan de acuerdo a las siguientes categorías:

- Nacionalidades según el género.
- Factores de exclusión según el género.
- Tipos de contratos clasificados por género.
- Tipos de jornada también divididos por género.

Este enfoque propuesto no solo mantendrá la funcionalidad que ya están utilizando, sino que cuando surjan nuevas demandas por parte de la Xunta, podrán agregar los nuevos campos a las filas de los participantes o a las estadísticas en la segunda página sin afectar la funcionalidad existente. Esto significa que podrán adaptarse a los requisitos cambiantes de manera más eficiente y rápida, sin tener que modificar lo que está ya implementado.

En la capa del backend, se han desarrollado dos endpoints. El primero de ellos permite obtener el archivo Excel generado haciendo uso de la herramienta Apache POI. El segundo endpoint proporciona los datos necesarios para poder generar los gráficos en la interfaz.

En la parte del frontend, se ha creado una interfaz de usuario donde se incluyen dos campos de entrada para especificar la fecha de inicio y la fecha de fin. Asimismo, se ha incorporado un botón con la función de generar estadísticas. Una vez que el usuario envía la solicitud haciendo clic en este botón, el sistema realiza automáticamente dos acciones:

- Descarga del Archivo Excel (6.15): Se genera el archivo Excel con los datos y las estadísticas correspondientes utilizando la biblioteca Apache POI. Luego, este archivo se descarga automáticamente en el dispositivo del usuario.
- Visualización de Gráficos (6.16): El usuario es redirigido a una página donde se presentan los gráficos basados en los datos obtenidos. Estos gráficos proporcionan una representación visual de las estadísticas generadas.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Fecha	Programa	Situación	Retornado	Pi	dni/nie/pas	Nombre	Apellidos	Edad	Fecha nacimiento	Genero	Pais origen	Municipio	Provincia
20-08-2023	Acogida	IRREGULAR	NO	SI	p1	Name1	Surnames1	33	01-01-1990	H	Bolivia	Boiro	Pontevedra
20-08-2023	Sociolaboral	IRREGULAR	NO	SI	p2	Name2	Surnames2	33	01-01-1990	H	Jordania	Camarifias	Pontevedra
20-08-2023	Jurídico	REGULAR	NO	SI	p3	Name3	Surnames3	33	01-01-1990	M	Nauru	Arzúa	Lugo
20-08-2023	Sociolaboral-Jurídico	IRREGULAR	NO	SI	p4	Name4	Surnames4	33	01-01-1990	M	Luxemburgo	Baña (A)	A Coruna
20-08-2023	Jurídico	IRREGULAR	NO	SI	p5	Name5	Surnames5	33	01-01-1990	H	Cuba	Cambre	Pontevedra
20-08-2023	Acogida	REGULAR	NO	SI	p6	Name6	Surnames6	33	01-01-1990	H	Argelia	Ames	A Coruna
20-08-2023	Jurídico	REGULAR	NO	SI	p7	Name7	Surnames7	33	01-01-1990	H	Camboya	Cerdido	Lugo
20-08-2023	Acogida	IRREGULAR	NO	SI	p8	Name8	Surnames8	33	01-01-1990	H	Noruega	Corcubión	Ourense
20-08-2023	Sociolaboral	IRREGULAR	NO	SI	p9	Name9	Surnames9	33	01-01-1990	M	Mauricio	Cerdido	Pontevedra
20-08-2023	Jurídico	REGULAR	NO	SI	p10	Name10	Surnames10	33	01-01-1990	H	Marruecos	Cerdido	Lugo
20-08-2023	Sociolaboral-Jurídico	IRREGULAR	NO	SI	p11	Name11	Surnames11	33	01-01-1990	H	Senegal	Cesuras	Lugo
20-08-2023	Sociolaboral-Jurídico	REGULAR	NO	SI	p12	Name12	Surnames12	33	01-01-1990	M	Tayikistán	Cabana de Bergantiños	A Coruna
20-08-2023	Acogida	IRREGULAR	NO	SI	p13	Name13	Surnames13	33	01-01-1990	M	Nigeria	Arzúa	Ourense
20-08-2023	Jurídico	IRREGULAR	NO	SI	p14	Name14	Surnames14	33	01-01-1990	M	Malawi	Coirós	Ourense
20-08-2023	Sociolaboral	IRREGULAR	NO	SI	p15	Name15	Surnames15	33	01-01-1990	H	Islas Cocos (Keeling)	Brión	Lugo
20-08-2023	Sociolaboral-Jurídico	REGULAR	NO	SI	p16	Name16	Surnames16	33	01-01-1990	M	Eritrea	Boqueixón	A Coruna
20-08-2023	Sociolaboral	IRREGULAR	NO	SI	p17	Name17	Surnames17	33	01-01-1990	M	Estonia	Arteixo	A Coruna
20-08-2023	Acogida	IRREGULAR	NO	SI	p18	Name18	Surnames18	33	01-01-1990	H	Eritrea	Boiro	Lugo
20-08-2023	Sociolaboral	REGULAR	NO	SI	p19	Name19	Surnames19	33	01-01-1990	H	Filipinas	Carral	A Coruna
20-08-2023	Sociolaboral	REGULAR	NO	SI	p20	Name20	Surnames20	33	01-01-1990	H	Angola	Cabana de Bergantiños	A Coruna
20-08-2023	Acogida	REGULAR	NO	SI	p21	Name21	Surnames21	33	01-01-1990	M	Isla de Man	Ames	Lugo
20-08-2023	Sociolaboral	IRREGULAR	NO	SI	p22	Name22	Surnames22	33	01-01-1990	H	Uganda	Coruña (A)	Ourense
20-08-2023	Acogida	REGULAR	NO	SI	p23	Name23	Surnames23	33	01-01-1990	M	Emiratos Árabes Unidos	Ares	A Coruna
20-08-2023	Sociolaboral	REGULAR	NO	SI	p24	Name24	Surnames24	33	01-01-1990	M	Hong kong	Ares	Pontevedra
20-08-2023	Sociolaboral-Jurídico	IRREGULAR	NO	SI	p25	Name25	Surnames25	33	01-01-1990	M	Bahamas	Ares	A Coruna
20-08-2023	Sociolaboral	REGULAR	NO	SI	p26	Name26	Surnames26	33	01-01-1990	H	Seychelles	Abegondo	Lugo
20-08-2023	Sociolaboral-Jurídico	REGULAR	NO	SI	p27	Name27	Surnames27	33	01-01-1990	M	Curazao	Boiro	Lugo
20-08-2023	Sociolaboral	IRREGULAR	NO	SI	p28	Name28	Surnames28	33	01-01-1990	M	Cuba	Carballo	A Coruna
20-08-2023	Jurídico	IRREGULAR	NO	SI	p29	Name29	Surnames29	33	01-01-1990	M	Mali	Cerceda	Ourense
20-08-2023	Sociolaboral-Jurídico	REGULAR	NO	SI	p30	Name30	Surnames30	33	01-01-1990	H	Antigua y Barbuda	Baña (A)	Pontevedra

Figura 6.15: Excel generado.

Esta implementación brinda una experiencia fluida para el usuario, permitiéndoles tanto obtener los datos en formato Excel como visualizar de manera interactiva los gráficos relacionados con las estadísticas generadas.

## Nacionalidades

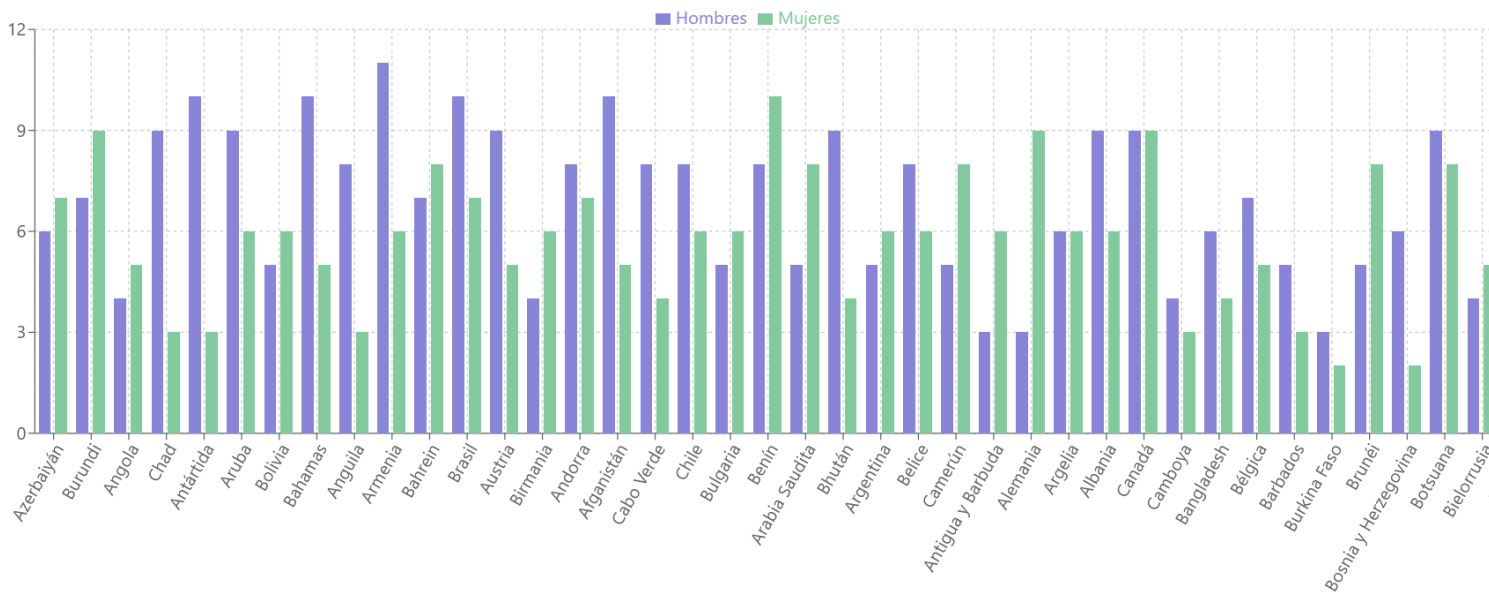


Figura 6.16: Gráfico nacionalidades.

Con el fin de verificar el funcionamiento adecuado de la aplicación, se han desarrollado scripts utilizando JavaScript para generar una cantidad significativa de datos simulados. Estos scripts tienen como objetivo crear una carga considerable en la aplicación, lo que permite evaluar su rendimiento. Aunque la generación de estadísticas con quinientos participantes es prácticamente inmediata, se ha incorporado un loader para manejar situaciones en las que la petición no se complete instantáneamente. Esta mejora proporciona una experiencia más fluida al usuario, al ofrecer una representación visual mientras la acción se encuentra en progreso. El siguiente es un ejemplo de un script que genera quinientos participantes:

```

1 const fs = require('fs');
2
3 function getRandomGender() {
4   const genders = ['H', 'M'];
5   const randomIndex = Math.floor(Math.random() * genders.length);
6   return genders[randomIndex];
7 }
8
9 function generateInserts() {
10   const totalInserts = 500;
11   const inserts = [];
12

```

```

13   for (let id = 1; id <= totalInserts; id++) {
14       const birthDate = '1990-01-01';
15       const datePi = '2023-01-01';
16       const email = `email${id}@example.com`;
17       const gender = getRandomGender();
18       const interviewPi = '2022-01-01';
19       const name = `Name${id}`;
20       const phone = '123456789';
21       const surnames = `Surnames${id}`;
22       const country_id = Math.floor(Math.random() * 240) + 1;
23       const pas = `p${id}`;
24
25       const insert = `
26           INSERT INTO Participant (birthDate, datePi, email, gender,
27           interviewPi, name, phone, surnames, country_id, pas)
28           VALUES ('${birthDate}', '${datePi}', '${email}',
29           '${gender}', '${interviewPi}', '${name}', '${phone}',
30           '${surnames}', ${country_id}, '${pas}');
31       `;
32       inserts.push(insert);
33   }
34   return inserts.join('\n');
35 }
36 const generatedInserts = generateInserts();
37
38 fs.writeFile('insertsParticipantes.sql', generatedInserts, (err)
39 => {
40     if (err) {
41         console.error('Error al escribir en el archivo:', err);
42     } else {
43         console.log('Archivo generado con éxito: inserts.sql');
44     }
45 });

```

Fue necesario llevar a cabo la simulación de datos para múltiples tablas, principalmente debido a que la tabla "AnnualData" presenta un elevado número de relaciones n:n. Estas relaciones generan tablas intermedias que deben ser completadas con información adecuada para su correcto funcionamiento.

## 6.6 Iteración 4

### 6.6.1 Análisis

En esta iteración, el enfoque se dirigió hacia la implementación de los casos de uso relacionados con la autenticación y la administración de los usuarios de la aplicación, específicamente dirigidos a los trabajadores que utilizarán dicha aplicación. Se establecieron dos roles claramente diferenciados: el rol de "técnico", que posee acceso básico a la aplicación y la capacidad de modificar su información personal, y el rol de "admin", quien cuenta con la facultad de agregar y eliminar técnicos.

### 6.6.2 Diseño e implementación

#### CU-16 Iniciar sesión

Para la autenticación de usuarios se ha usado el estándar [JWT](#) [23], de esta forma en cada petición del usuario se enviará también el token, y podrá acceder a los recursos a los que dicho token tiene acceso.

Se implementó en el controlador relativo a usuarios una operación que recibe el nombre de usuario y contraseña, una vez validados devuelve los datos del usuario y el token que el usuario enviará en las próximas peticiones que realice al [backend](#).

En el [frontend](#), este token se guarda en el almacenamiento de sesión del navegador, así al recargar la página el usuario no necesitará volver a autenticarse, pero en caso de que cierre la pestaña o abra una nueva sí deberá hacerlo.

En la figura [6.17](#), se muestra un diagrama de secuencia, donde se detalla el proceso de autorización de una petición usando [JWT](#). Como se puede apreciar, cuando llega una petición que incluye un token, se lleva a cabo un control de acceso, en el que se comprueba si el usuario tiene los roles suficientes para la petición que ha mandando.

Se desarrolló un componente en el [frontend](#) que consta de un formulario junto con sus campos respectivos para ingresar el nombre de usuario y la contraseña. En situaciones en las que se ingresen credenciales incorrectas, el [backend](#) responderá con un código NOT FOUND, el cual desencadenará la visualización de un mensaje correspondiente para informar al usuario sobre el error.

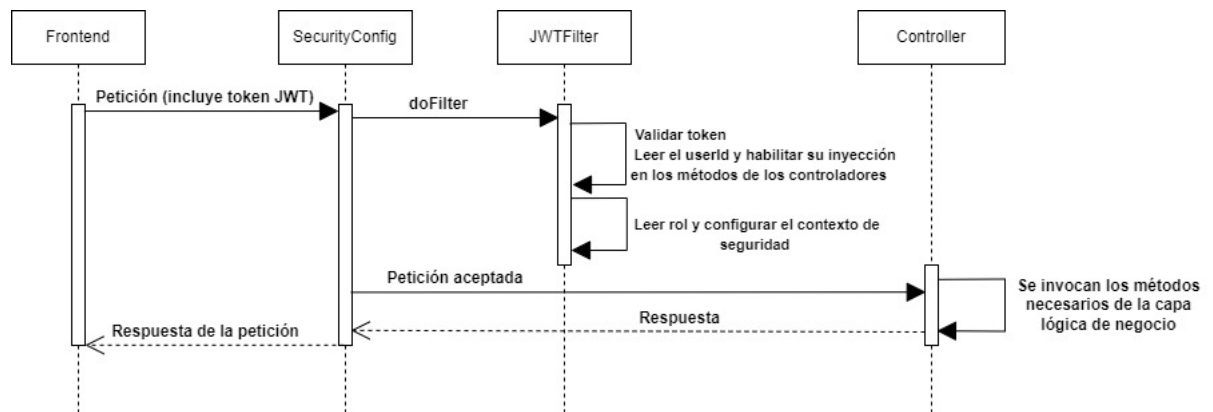


Figura 6.17: Diagrama de secuencia de autorización con JWT.

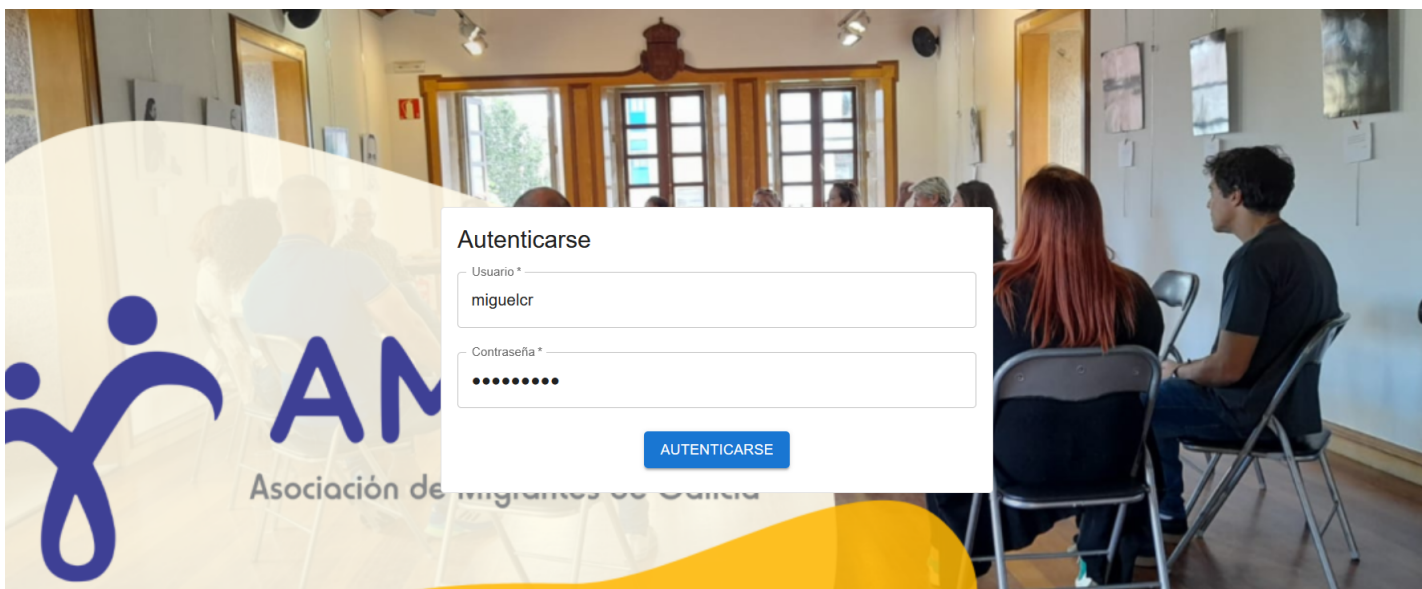


Figura 6.18: Iniciar sesión.

### CU-17 Cerrar sesión

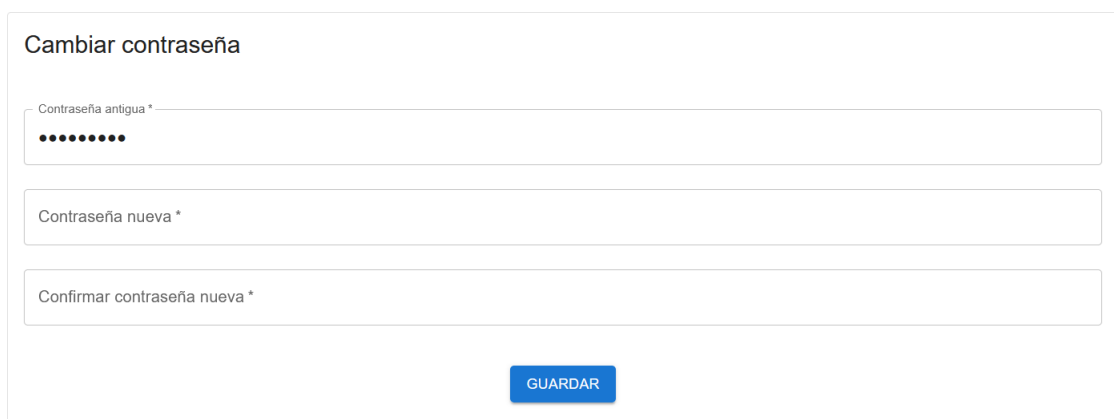
En esta instancia, se realizó una implementación en la interfaz de usuario, específicamente dentro del componente **Header**. En este componente, se introdujo una funcionalidad adicional conocida como **Toolbar**, que es básicamente una lista colapsada. A esta lista se le asignó un icono que contiene la inicial del nombre del usuario. Al hacer clic en este icono, se despliega un menú que presenta diversas opciones relacionadas con los usuarios. Es en este menú donde se ha incorporado la alternativa para llevar a cabo el cierre de sesión.

La elección de la opción cerrar sesión activa una acción específica, diseñada para eliminar el token **JWT** del almacenamiento de sesión del navegador. Esto es especialmente relevante debido a que el estado de autenticación no se guarda en el **backend**.

### CU-18 Cambiar contraseña

Para este caso de uso se implementó en el controlador una operación, que recibe la contraseña antigua y la nueva. En caso de que la contraseña antigua no sea correcta el **backend** responderá con la excepción correspondiente, en caso contrario, se cambiará la contraseña del usuario, modificando la información en la base de datos.

En el **frontend** se implementó un formulario similar a los mencionados anteriormente, en este caso los campos son la contraseña antigua, la contraseña nueva y su confirmación. Si la nueva contraseña no coincide con la confirmación, se mostrará un error y no se enviará el formulario. Para llegar al formulario, se añadió la opción en el menú mencionado en el anterior caso de uso.



El formulario, titulado "Cambiar contraseña", contiene tres campos de entrada de texto y un botón de acción. El primer campo, etiquetado "Contraseña antigua \*", muestra caracteres ocultos por puntos. El segundo campo es "Contraseña nueva \*" y el tercero es "Confirmar contraseña nueva \*". Debajo de los campos, hay un botón azul con el texto "GUARDAR".

Figura 6.19: Cambiar contraseña.

**CU-20 Buscar trabajador por palabra clave** Un administrador puede buscar empleados utilizando una palabra clave, ya sea un nombre completo o parcial. El sistema buscará coincidencias en nombres y apellidos entre los empleados registrados y mostrará una lista en la que mostrará los datos relativos a los usuarios según el criterio de búsqueda.

Para implementar esta funcionalidad, se ha establecido el siguiente enfoque: si no se introduce ninguna palabra de búsqueda, se recuperará la lista completa de usuarios de la aplicación. Por otro lado, si se introduce una palabra, se activará una consulta que aplicará un filtro en función de dicha palabra. En la interfaz del usuario, el resultado se presenta en forma de tabla, donde se muestran los datos de cada usuario obtenido, junto con la opción de llevar a cabo acciones específicas sobre cada uno de ellos.

## Buscador técnicos

Nombre	Apellidos	Email	Categoría	Dar/Quitar privilegios	Eliminar
Miguel	Crespo Rozados	m.crespo.rozados@udc.es	ADMIN	↓	
Admin	Admin	admin@udc.es	USER	↑	

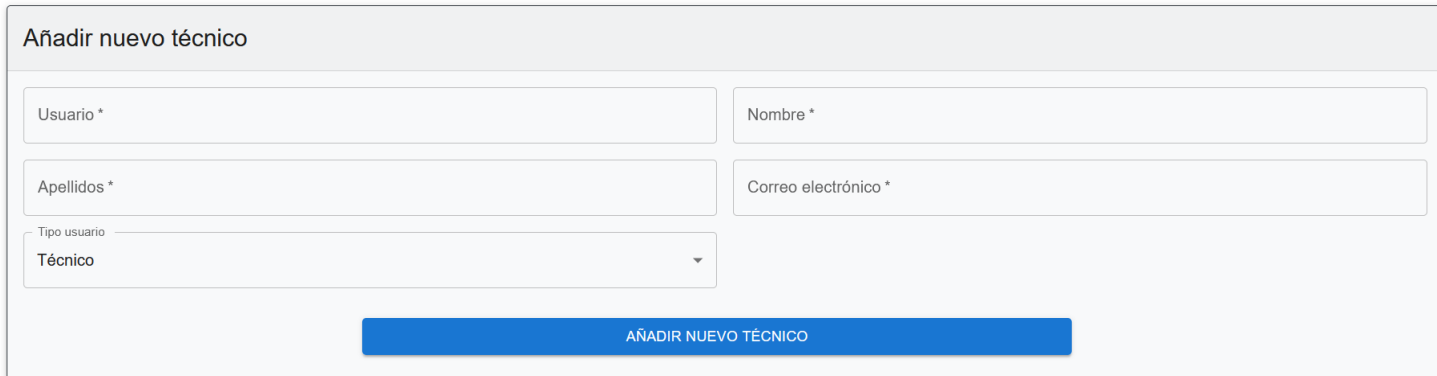
Figura 6.20: Buscar usuario.

**CU-19 Registrar nuevo trabajador, CU-21 Cambiar permisos trabajador, CU-22 Eliminar trabajador** Un administrador tiene la capacidad de dar de alta, de baja y cambiar los permisos de todos los usuarios de la aplicación. Los datos básicos del usuario son los siguientes: nombre usuario, contraseña, nombre, apellidos, correo y rol.

En el proceso de registro (figura 6.21), se recopilan los datos del usuario, a excepción de la contraseña, ya que se genera automáticamente. Esta contraseña predeterminada debe ser modificada por el propio usuario cuando inicie sesión por primera vez.

Con relación a las funcionalidades previamente mencionadas, las acciones de actualizar el rol y eliminar un usuario se llevan a cabo desde la tabla de usuarios que se despliega a través de la funcionalidad "Buscar trabajador por palabra clave" (CU-20). En el caso de la





Formulario para añadir un nuevo técnico. El formulario tiene un encabezado "Añadir nuevo técnico". Contiene los siguientes campos:

- Usuario \*
- Nombre \*
- Apellidos \*
- Correo electrónico \*
- Tipo usuario: un menú desplegable con la opción "Técnico" seleccionada.

En la parte inferior del formulario hay un botón azul con el texto "AÑADIR NUEVO TÉCNICO".

Figura 6.21: Crear usuario.

actualización de roles, se implementa una flecha seleccionable. Si el usuario es un técnico, la flecha apunta hacia arriba, permitiendo ascenderlo a la categoría de administrador. Por otro lado, si el usuario es un administrador, la flecha apunta hacia abajo, lo que posibilita su degradación a la posición de técnico.

En cuanto a la eliminación de un usuario, esta acción conlleva la revocación completa de sus accesos y permisos en la aplicación.

**CU-23 Editar datos trabajador** El trabajador cuenta con la capacidad de editar sus propios datos personales. Con este fin, se ha incorporado esta funcionalidad en la barra de herramientas de acciones del usuario. Esta característica se presenta en forma de un formulario que facilita la carga de los datos de usuario actuales y permite su modificación de manera sencilla.

## 6.7 Iteración 5

### 6.7.1 Análisis

En la última fase de desarrollo de funcionalidades, el enfoque consistió en la implementación de los casos de uso relacionados con la gestión del voluntariado. Los casos de uso abordados fueron los siguientes:

- CU-24 Registrar datos voluntario.
- CU-25 Buscar voluntario.
- CU-26 Ver detalles voluntario.
- CU-27 Editar datos voluntario.
- CU-28 Crear colaboración.
- CU-29 Editar colaboración.

### 6.7.2 Diseño e implementación

**CU-25 Buscar voluntario** El sistema facilita la búsqueda de voluntarios al permitir coincidencias en sus nombres y apellidos. Además, ofrece la capacidad de aplicar filtros para identificar exclusivamente a aquellos voluntarios que tienen colaboraciones actualmente activas. Con este fin, se diseñó un servicio y un controlador específicos para gestionar las operaciones relacionadas con los voluntarios. Las consultas personalizadas creadas para encontrar voluntarios con colaboraciones activas son las siguientes:

```
1      @Query("SELECT DISTINCT c.volunteer FROM Collaboration c
2          JOIN c.volunteer v " +
3          "WHERE (v.firstName LIKE :keyword
4          OR v.lastName LIKE :keyword) " +
5          "AND c.endDate IS NULL OR c.endDate > CURRENT_TIMESTAMP")
6      List<Volunteer> findActiveVolunteersByName
7      (@Param("keyword") String keyword);
8
9      @Query(" SELECT DISTINCT c.volunteer FROM Collaboration c
10         JOIN c.volunteer v " +
11         "WHERE c.endDate IS NULL OR c.endDate > CURRENT_TIMESTAMP")
12      List<Volunteer> findActiveVolunteers();
13
```

En las consultas, se lleva a cabo una operación de unión (join) entre las entidades Collaboration y Volunteer. A través de la aplicación de un filtro basado en la condición de que la fecha de finalización sea nula o que sea mayor al momento actual, se logra obtener únicamente a los participantes que se encuentran actualmente realizando en actividades de voluntariado.

En el frontend, se incorporó un campo de entrada de texto que permite introducir el nombre o apellido para aplicar un filtro. Además, se incluyó un checkbox que posibilita la elección de obtener únicamente a los voluntarios activos. Al realizar la búsqueda, los resultados se muestran en una tabla que permite visualizar de manera organizada la información. A partir de esta tabla, es posible acceder a los datos específicos de un voluntario en particular.

## Buscador voluntarios

☐ Solo activos

Nombre	Apellidos	Email	Teléfono
Miguel	Crespo Rozados	m.crespo.rozados@udc.es	684350303

Figura 6.22: Buscar voluntario.

**CU-26 Ver detalles voluntario** El sistema ofrece una manera conveniente para acceder a los detalles de un voluntario en particular, junto con una lista exhaustiva de las colaboraciones que ha participado, ya sea en el pasado o en la actualidad. La información asociada al voluntario incluye su nombre, apellidos, número de identificación (DNI), número de teléfono y dirección de correo electrónico. En relación a las colaboraciones, se proporciona información sobre la fecha de inicio, fecha de finalización (si aplica), cantidad de horas dedicadas y una descripción del trabajo realizado.

En la interfaz de usuario, los datos del voluntario se presentan dentro de un contenedor designado, mientras que debajo de este contenedor, se visualiza una tabla que muestra las diferentes colaboraciones en las que el voluntario ha participado. Para cada colaboración, se brinda la opción de llevar a cabo una acción específica.

[ACTUALIZAR VOLUNTARIO](#)[CREAR COLABORACIÓN](#)[CANCELAR](#)

### Información voluntario

Nombre: Miguel Crespo RozadosEmail: m.crespo.rozados@udc.es

Teléfono: 684350303DNI: 54153223E

### Colaboraciones

Fecha inicio	Fecha fin	Número horas	Descripción	Acciones
2023-08-11	2023-10-26	250	Creación aplicación	<a href="#">EDITAR</a> <a href="#">ELIMINAR</a>
2023-08-03	Activo	10	Participo en una formación a empleados	<a href="#">EDITAR</a> <a href="#">ELIMINAR</a>

Figura 6.23: Ver detalles voluntario.

**CU-24 Registrar datos voluntario, CU-27 Editar datos voluntario** Se ha implementado en el sistema la capacidad de registrar nuevos voluntarios y de editar su información. Para lograrlo, se han introducido las operaciones correspondientes en el servicio y el controlador que están dedicados a la entidad de voluntarios. En la interfaz de usuario (frontend), se ha creado un formulario para facilitar el registro de datos. En este formulario, se ingresan los datos mencionados previamente. La validación se encarga de asegurar que todos los campos estén completos y de verificar la validez del dni introducido.

Una vez se ha completado el formulario, se ofrecen dos opciones. Puedes optar por simplemente añadir al voluntario, o también tienes la opción de crear al voluntario y, al mismo tiempo, registrar su primera colaboración. Además, se ha habilitado la funcionalidad para editar la información de los voluntarios ya registrados.

Cuando se desea editar los datos de un voluntario, el proceso comienza al visualizar los detalles del participante en cuestión. Desde allí, tienes la posibilidad de abrir una modal que muestra los datos previamente guardados del voluntario. Dentro de esta modal, puedes realizar modificaciones en los datos necesarios y luego guardar los cambios.

## Registro voluntario

CANCELAR

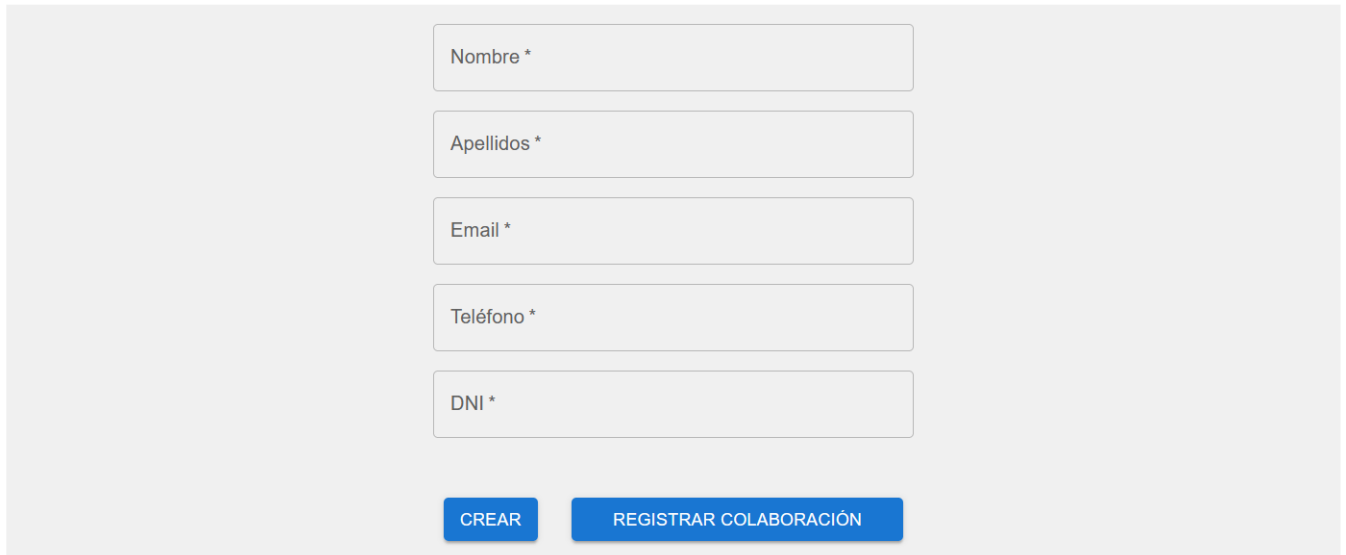
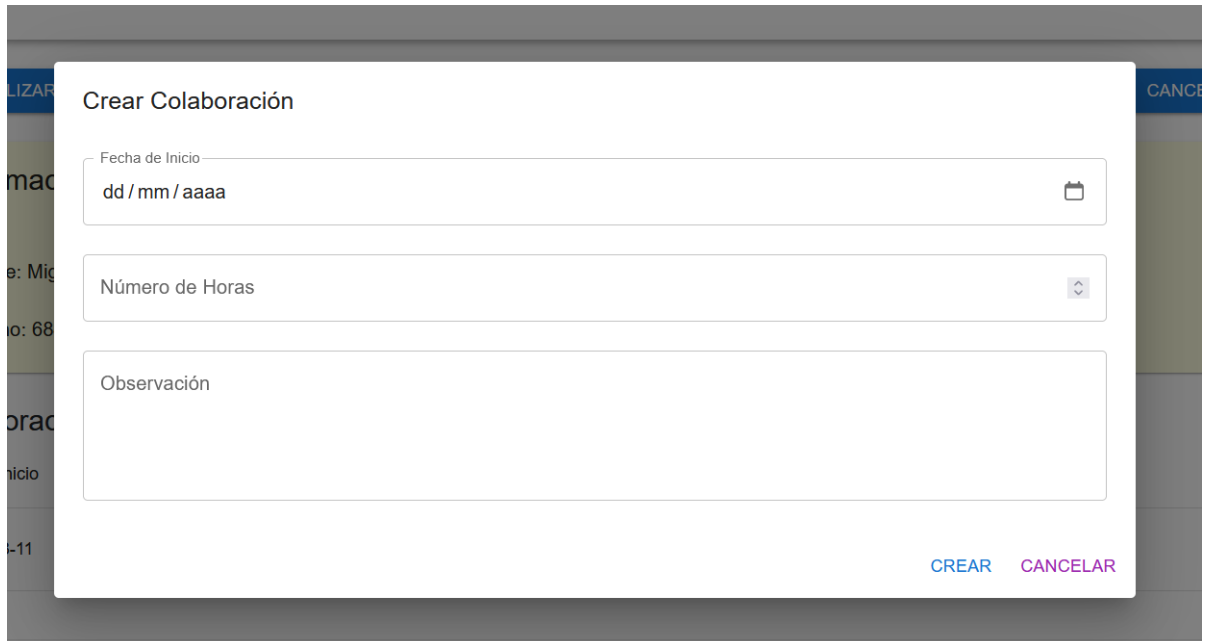
El formulario de registro de voluntario está diseñado con un fondo gris claro. En la parte superior derecha, hay un botón azul con el texto "CANCELAR" en blanco. El formulario principal contiene cinco campos de entrada de texto, cada uno con un label y un asterisco indicando que es obligatorio: "Nombre \*", "Apellidos \*", "Email \*", "Teléfono \*" y "DNI \*". Los campos están apilados verticalmente. Debajo de los campos, hay dos botones azules: "CREAR" a la izquierda y "REGISTRAR COLABORACIÓN" a la derecha.

Figura 6.24: Registrar voluntario.

**CU-28 Crear colaboración, CU-29 Editar colaboración, CU-30 Eliminar colaboración** En conjunto, los casos de uso proporcionan las capacidades para registrar, modificar y eliminar información sobre las colaboraciones llevadas a cabo por los voluntarios. Estas operaciones se encuentran implementadas en un controlador y un servicio específicos destinados a gestionar las colaboraciones.

En la interfaz, todas estas acciones se realizan a través de la visualización de los detalles del voluntario. Desde allí, un botón permite abrir una ventana modal para crear una nueva colaboración.

En el caso de la edición y eliminación, en la tabla previamente mencionada, se brinda la opción de llevar a cabo estas funcionalidades para cada colaboración individual. Cuando se selecciona la opción de editar, se abre una ventana modal que posibilita modificar los datos actuales de dicha colaboración. Para llevar a cabo la eliminación, simplemente se solicita confirmación antes de proceder con la acción.



Crear Colaboración

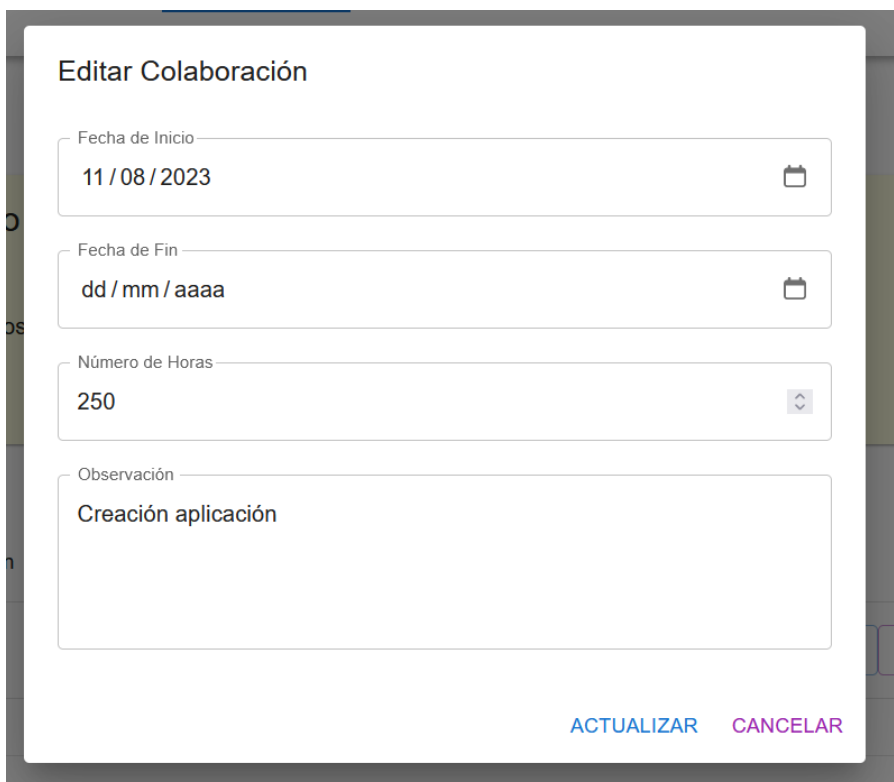
Fecha de Inicio  
dd / mm / aaaa

Número de Horas

Observación

CREAR CANCELAR

Figura 6.25: Modal creación colaboración.



Editar Colaboración

Fecha de Inicio  
11 / 08 / 2023

Fecha de Fin  
dd / mm / aaaa

Número de Horas  
250

Observación  
Creación aplicación

ACTUALIZAR CANCELAR

Figura 6.26: Modal editar colaboración.

## 6.8 Iteración 6

Como se mencionó en la fase de planificación, el enfoque de la última iteración se centra en la creación de esta memoria. Para llevar a cabo esta tarea, se hace uso de la aplicación que ya ha sido desarrollada. Adicionalmente, se ha integrado Swagger UI, una herramienta que simplifica la visualización y evaluación de las llamadas a la API REST proporcionada por el backend. Asimismo, se han elaborado las configuraciones necesarias para el despliegue de la aplicación y se ha creado su respectiva documentación.

### 6.8.1 Swagger UI

Esta incorporación está orientada principalmente a la idea de futuras expansiones del proyecto, permitiendo que nuevos desarrolladores puedan comprender de manera visual y sencilla las funcionalidades que ya han sido implementadas. Esta herramienta resulta especialmente útil al permitir, una vez que la aplicación está arrancada en el entorno local, acceder a través del navegador a la siguiente URL: <http://localhost:8080/swagger-ui/index.htm>. Esto proporciona una vista completa de la API REST del backend, proporcionando una forma conveniente de explorar los diferentes endpoints (figura 6.27).

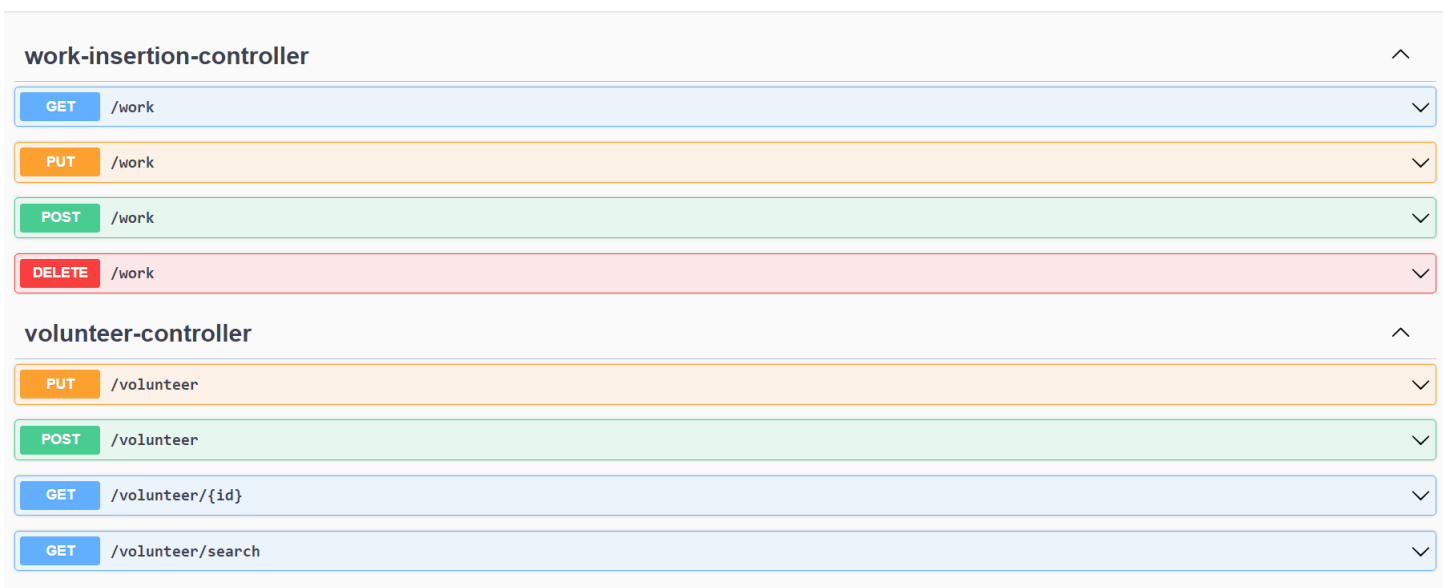


Figura 6.27: Swagger UI.

Para cada solicitud como se puede observar en 6.28, es posible expandir el menú desplegable permitiendo probar la solicitud. Esto proporciona una vista detallada del formato de la solicitud, así como las posibles respuestas generadas por la API.

**POST** /volunteer

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "id": 0,
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "telephone": "string",
  "dni": "string"
}
```

Responses

Code	Description	Links
200	OK	No links
	<p>Media type</p> <p>*/*</p> <p>Controls Accept header.</p> <p>Example Value   Schema</p> <pre>{   "id": 0,   "firstName": "string",   "lastName": "string",   "email": "string",   "telephone": "string",   "dni": "string" }</pre>	
400	Bad Request	No links

Figura 6.28: Detalles endpoint swagger.



### 6.8.2 Despliegue

Finalmente, se ha documentado un manual de instalación del proyecto. Esta guía incluye en detalle las herramientas necesarias tanto para el despliegue de la aplicación como para continuar con el desarrollo.

Se han incorporado las configuraciones necesarias tanto en el backend como en el frontend, con el objetivo de facilitar despliegues sin complicaciones. En el backend, se han establecido dos perfiles de ejecución de Spring: uno para el entorno de desarrollo y otro para el de producción. Una vez que la instancia de MySQL utilizada en el backend ha sido creada y las variables de conexión detalladas en el manual configuradas, se presenta en detalle paso a paso cómo crear el archivo JAR que contiene el backend y cómo ejecutarlo.

En cuanto al frontend, el proceso de despliegue implica simplemente realizar el build del artefacto, generando así una carpeta que reúne todos los archivos esenciales para ejecutar la aplicación de manera optimizada en un servidor web.

Para agilizar la instalación del proyecto y facilitar su desarrollo, se han detallado los comandos necesarios para generar contenedores, tanto a través de Docker como de una alternativa similar denominada Podman. Asimismo, se incluyen instrucciones detalladas para crear las bases de datos utilizando los scripts correspondientes. Una vez completada la instalación, el manual explica cómo poner en marcha el proyecto, además de contener un enlace a esta documentación para obtener un contexto más amplio.

Al terminar esta fase, el proyecto fue hecho público en un repositorio Git dentro de la organización "TsolidarioFG," creada por la universidad. El proyecto se encuentra en la url <https://github.com/TsolidarioFG/2023-AMIGA>. Esta decisión permite que el proyecto sea de código abierto (open source) con licencia MIT, lo que habilita a cualquier persona a contribuir a su desarrollo en el futuro.

# Conclusiones

---

El principal objetivo de este proyecto consistió en desarrollar una aplicación web destinada a optimizar los flujos de trabajo y economizar el tiempo empleado por los trabajadores de la organización [AMIGA](#). En este sentido, se presentó una oportunidad para aplicar los conocimientos adquiridos en la mención de Ingeniería de Software. El desafío principal durante el proceso de desarrollo se centró en la identificación de los requisitos, ya que los clientes carecían de una visión clara sobre la manera de realizar a través de una aplicación informática las tareas que hacían de manera manual. Aquí, los principios de Ingeniería de Requisitos fueron cruciales para el progreso exitoso de la aplicación. La implementación de un sistema real proporcionó una gran forma de aprendizaje acerca la gestión de proyectos con aplicabilidad práctica.

Por otro lado, la etapa de implementación permitió obtener mayor experiencia en tecnologías previamente abordadas a lo largo del grado, al mismo tiempo que permitió practicar a investigar y usar nuevas herramientas como **Apache Poi** [10], utilizada para generar hojas de cálculo Excel, **React-pdf** [21] y **Recharts** [20], empleada para la elaboración de gráficos. Un elemento crucial en el proceso de desarrollo de la aplicación fue la creación del modelo de datos, poniendo en práctica los conocimientos adquiridos en bases de datos, ya que se esperan futuras expansiones de la aplicación. Debido a esto, se le dio gran importancia a la construcción de una arquitectura modular, compuesta por componentes independientes y desacoplados, con el fin de facilitar la incorporación de modificaciones y el mantenimiento del sistema.

Por último, se puso en práctica la habilidad de diseñar una interfaz de usuario intuitiva y de fácil utilización, con el propósito de que cualquier nuevo empleado pueda utilizar la aplicación sin requerir una formación previa exhaustiva.

## 7.1 Trabajo futuro

Desde su inicio, la aplicación fue concebida con la intención de proporcionar un modelo de gestión integral, incorporando las funcionalidades esenciales que capacitarían a los usuarios para emplear el producto desde su puesta en marcha y que, al mismo tiempo, contribuirían a la mejora de sus procesos de trabajo. Con todo, cualquier software es susceptible de mejoras. Algunas posibles áreas de mejora son:

- Incorporar la funcionalidad de realizar firmas directamente en un dispositivo electrónico y luego integrar estas firmas de manera automatizada en los documentos PDF generados por la aplicación.
- A lo largo de mayor parte del proyecto, las reuniones se llevaron a cabo con una de las responsables del área sociolaboral. Sin embargo, en las etapas finales, se incorporó a estas reuniones la encargada del apoyo jurídico a los participantes, quien aportó ideas innovadoras para la mejora de la aplicación. Por lo tanto, se plantea la posibilidad de considerar estas funcionalidades para futuras implementaciones.

# **Apéndices**

# Datos personales

BackLink

CANCELAR

FormPage1Part1

Nombre \*

Apellidos \*

DNI

PAS

NIE

Fecha de Nacimiento\*  
dd / mm / aaaa

Calendar icon

Sexo

Dropdown arrow

Email

Provincia

Municipio

Domicilio \*

Código Postal

Teléfono/s \*

País de origen

Nacionalidades

☐ Empadronado

Fecha de Empadronamiento  
dd / mm / aaaa

Número de personas empadronadas

Spinners

Vivienda

Estado civil

Tipo de convivencia

FormPage1

FormPage1Part2

SIGUIENTE

FormContainer

AMIGA - Asociación de Migrantes de Galicia

Figura 1: Primera página formulario acogida.

# Datos personales

RegisterMinor
BackLink
CANCELAR

Añadir menores

Sexo menor
Fecha de nacimiento \*

dd / mm / aaaa

AÑADIR MENOR

Sexo	Fecha de nacimiento	Acciones
Masculino	08 de septiembre de 2021	

☐ Protección Internacional

Fecha de Manifestación PI
dd / mm / aaaa

Fecha de 1ª Entrevista PI
dd / mm / aaaa

Seleccionar factores exclusión

E. retornado
No

Hogar sin empleo
No

Hogar 1 adulto
No

Dependientes a cargo
No

☐ Trabajador Social

Datos del Trabajador Social

☐ Cobertura Social

Datos de Cobertura Sanitaria

Dis capacidad
No

Situación administrativa

FormPage2

ANTERIOR
SIGUIENTE

Page2Data

FormContainer

Figura 2: Segunda página formulario acogida.

## Experiencia Laboral

[CANCELAR](#)

Seleccionar estudios	▼	Titulación homologada	▼
Seleccionar idiomas	▼	Formación demandada *	
Situación laboral	▼	Cualificación profesional *	
Experiencia profesional en país de origen *			
Experiencia profesional en España *			
Habilidades y limitaciones *			
Horario disponible *			
Permiso de conducir		Válido en España	
No		No	
Vehículo	▼	Inscrito a SEPE	▼
No	▼	No	▼
		Duración meses desempleo	
Tipo de prestación			
No			
Otra prestación		Fecha de prestación	
		dd / mm / aaaa	

[ANTERIOR](#)[SIGUIENTE](#)

Figura 3: Tercera página formulario acogida.

## Tipo demanda

CANCELAR

### Añadir Programas

Seleccionar programa ▼

Itinerario

No ▼

AÑADIR

Programa

Itinerario

Eliminar

Seleccionar demanda ▼

Derivación \*

Observaciones

ANTERIOR

SIGUIENTE

Figura 4: Cuarta página formulario acogida.



# Lista de acrónimos

---

**AMIGA** Asociación de Migrantes de Galicia. [1](#), [2](#), [6](#), [65](#)

**DAO** Data Access Object. [23](#), [33](#)

**DTO** Data Transfer Object. [23](#), [34](#), [36](#), [38](#)

**HTTP** Hypertext Transfer Protocol. [24](#)

**IDE** Integrated Development Environment. [20](#), [21](#)

**JRE** Java Runtime Environment. [20](#)

**JWT** JSON Web Token. [23](#), [52](#), [54](#)

**POM** Project Object Model. [20](#)

**REST** Representational State Transfer. [23](#), [24](#)

**SPA** Single Page Application. [3](#), [21](#)

**XML** eXtensible Markup Language. [24](#)

# Glosario

---

**atención** Registro de actividades realizado al llevar a cabo un seguimiento o efectuar acciones relacionadas con un participante.. 9

**backend** Parte de la aplicación que se encarga del acceso a datos y la lógica de negocio de la aplicación.. i, 3, 15, 20, 21, 23, 24, 32, 34, 52, 54

**framework** Librería que ofrece una estructura base para elaborar un proyecto con objetivos específicos. Sirve como punto de partida para la organización y desarrollo de software.. 3, 20

**frontend** Parte de la aplicación que interactúa directamente con los usuarios, y hace de intermediario entre ellos y el backend de la aplicación.. ii, 3, 15, 21, 24, 32, 34, 52, 54

**mockups** Se utilizan para visualizar el resultado del proyecto al cliente, enseñando una representación realista de las pantallas de la aplicación.. 14

**participante** Personas migrantes o en situación de vulnerabilidad social que son asesoradas por la ONG.. 6

# Bibliografía

---

- [1] Página web de scrum guides. [En línea]. Disponible en: <https://scrumguides.org/>
- [2] Página web de oracle. [En línea]. Disponible en: <https://www.oracle.com/java/>
- [3] H. Schildt, *Java the complete reference*. New York: McGraw-Hill Education, 2019.
- [4] Página web de spring. [En línea]. Disponible en: <https://spring.io/projects/spring-boot>
- [5] Página web de mysql. [En línea]. Disponible en: <https://www.mysql.com/>
- [6] Página web de docker. [En línea]. Disponible en: <https://www.docker.com/>
- [7] Página web de maven. [En línea]. Disponible en: <https://maven.apache.org/>
- [8] Página web de intellij idea. [En línea]. Disponible en: <https://www.jetbrains.com/idea>
- [9] Página web de postman. [En línea]. Disponible en: <https://www.postman.com/>
- [10] Página de web de apache poi. [En línea]. Disponible en: <https://poi.apache.org/>
- [11] Página web de mapstruct. [En línea]. Disponible en: <https://mapstruct.org/>
- [12] Página web de swagger. [En línea]. Disponible en: <https://swagger.io/>
- [13] Página web de node.js. [En línea]. Disponible en: <https://nodejs.org/es>
- [14] Página web de npm. [En línea]. Disponible en: <https://www.npmjs.com/>
- [15] Página web de javascript. [En línea]. Disponible en: <https://www.javascript.com/>
- [16] Página web de react. [En línea]. Disponible en: <https://es.reactjs.org/>
- [17] A. Banks, *Learning React : modern patterns for developing React apps*. Sebastopol, CA : O'Reilly, 2020.
- [18] Página web de redux. [En línea]. Disponible en: <https://es.redux.js.org/>

- [19] Página web de material-ui. [En línea]. Disponible en: <https://mui.com/>
- [20] Página de web de recharts. [En línea]. Disponible en: <https://recharts.org/en-US/>
- [21] Página web de react-pdf. [En línea]. Disponible en: <https://react-pdf.org/>
- [22] Página web de git. [En línea]. Disponible en: <https://git-scm.com/>
- [23] Página web de json web tokens. [En línea]. Disponible en: <https://jwt.io/>