



GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FINAL DE GRADO

Desarrollo de un frontend en ReactJS

Autor:
Pablo BERBEL MARÍN

Supervisor:
José Antonio CANO GÓMEZ
Tutor académico:
Óscar COLTELL SIMÓN

Fecha de lectura: 27 de julio de 2018 Curso académico 2017/2018

Resumen

Esta memoria explica el desarrollo del frontend del proyecto Constructor, una aplicación web que ofrece certificaciones a sus usuarios en base a sus conocimientos y a su experiencia laboral. Contiene tanto la metodología seguida durante la realización del proyecto, como el análisis y diseño. Sirve para realizar un acercamiento a muchas de las herramientas y tecnologías usadas en la actualidad para desarrollar software de calidad, sincronizar un equipo de desarrollo y facilitar la comunicación entre ellos.

Palabras clave

ReactJS, E-Force, formularios web, peticiones a API, SCRUM, componentes, web, diseño, SQL

Keywords

ReactJS, E-Force, web forms, API requests, SCRUM, components, web, design, SQL

A mis compañeros que han estado conmigo a lo largo de la carrera, sobre todo Albert, Adrián y Jose, a mis amigos, a mi supervisor, a Pau, a mi tutor y a toda mi familia, por su acompañamiento y dedicación, ya fuera durante la etapa académica o durante la estancia en prácticas.

Índice general

1. Introducción	7
1.1. Contexto y motivación del proyecto	7
1.1.1. E-Force	7
1.1.2. Proyecto Constructor	8
1.1.3. Proyecto propuesto por la empresa	9
1.1.4. Necesidad y utilidad del proyecto	9
1.1.5. ¿Por qué elegí este proyecto?	9
1.2. Objetivos del proyecto	10
1.2.1. Objetivos técnicos del proyecto Constructor	10
1.2.2. Objetivos técnicos de mi proyecto	10
1.2.3. Objetivos personales	11
1.3. Estructura de la memoria	11
2. Descripción del proyecto	13
2.1. Tecnologías y herramientas usadas en el proyecto	13
2.1.1. React	13
2.1.2. Bootstrap	14
2.1.3. Visual Studio Code	14
2.1.4. Postman	14

2.1.5.	Consola de Chrome	15
2.1.6.	Sourcetree	15
2.1.7.	Xampp	15
2.1.8.	React developer tools	15
2.1.9.	Bitbucket	16
2.1.10.	Slack	16
2.1.11.	OneDrive	17
2.1.12.	Escritorio remoto de Chrome	17
2.1.13.	.NET Core	17
2.1.14.	SQL Server	17
2.2.	Situación inicial del proyecto	18
3.	Planificación del proyecto	19
3.1.	Metodología	19
3.1.1.	Scrum	19
3.2.	Planificación	20
3.3.	Estimación de recursos y costes del proyecto	26
3.3.1.	Estimación de recursos	26
3.3.2.	Estimación de costes	27
3.4.	Seguimiento del proyecto	28
3.4.1.	Control del proyecto	28
3.4.2.	Desviaciones de la planificación	30
4.	Análisis y diseño del sistema	33
4.1.	Análisis del sistema	33
4.1.1.	Requisitos	33

4.1.2.	Diseño lógico de la base de datos	34
4.2.	Diseño de la arquitectura del sistema	36
4.2.1.	Arquitectura del sistema	37
4.2.2.	Migración de la base de datos	38
4.3.	Diseño de la interfaz	38
5.	Implementación y pruebas	41
5.1.	Detalles de implementación	41
5.1.1.	Uso de la API REST	41
5.1.2.	Control de usuarios	43
5.1.3.	Ciclo de vida de un componente	44
5.1.4.	Jerarquía de componentes	45
5.1.5.	Programación funcional	45
5.2.	Verificación y validación	46
6.	Conclusiones	47
6.1.	Ámbito formativo	47
6.2.	Ámbito profesional	47
6.3.	Ámbito personal	47
A.	Peticiones a la API	51
A.1.	Imágenes detalladas	51
A.2.	Función fetch en ReactJS	52
A.3.	Ciclo de vida de un componente en ReactJS	52

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

En este apartado voy a realizar una breve introducción del contexto del proyecto, junto con los motivos que me han llevado a escogerlo para realizar mi estancia en prácticas. Creo que es muy importante también, introducir la empresa donde he estado desarrollando el proyecto durante estos meses. En cuanto al proyecto, destacaré su necesidad y utilidad en la vida real, así como mi decisión de embarcarme en él.

1.1.1. E-Force

La empresa donde he realizado mi estancia de prácticas es E-Force. Su sede oficial se encuentra en Holanda, exactamente en Havik, un pueblo situado en la ciudad de Amersfoort. También disponen de otra sede en España, en el Espaitec de la Universitat Jaume I de Castellón, que es donde he estado yo trabajando durante estos meses. El nombre de la empresa española es E-Force Spain y mi supervisor, Jose Antonio Cano, es el CTO y cofundador. Actualmente, la empresa trabaja exclusivamente para clientes en Holanda y desarrolla en .NET Core y ReactJS.

Durante todo este tiempo, tanto la sede oficial holandesa como la sede española, han estado dedicándose al desarrollo de aplicaciones web y al reclutamiento de personal de las tecnologías de la información y la comunicación. ¿Qué tiene esta empresa que no tengan otras? El conocimiento, la experiencia y la pasión por hacer soluciones técnicamente avanzadas. Todos los proyectos asociados a la empresa surgieron de una pregunta o un problema, cuya solución parecía ser de un alcance muy elevado [3]. La filosofía de E-Force es trabajar duro, junto con el cliente, para acabar construyendo una infraestructura de gran valor, actualizada y de la cual se sientan orgullosos tanto los clientes como la propia empresa. Todo esto se consigue también, gracias a que en la empresa siempre se intenta estar a la última en cuanto a tecnologías se refiere. Estar al día es clave para el éxito y esto es uno de los valores añadidos principales de E-Force, que forma a sus trabajadores para que conozcan en profundidad las tecnologías más punteras dentro del sector.

1.1.2. Proyecto Constructor

El proyecto Constructor ha sido encargado a E-Force por parte de una organización holandesa, llamada Constructor Register RC/RO/RT. Esta organización monitoriza la calidad de los constructores y evaluadores constructivos, promoviendo la experiencia. Los constructores afiliados son profesionales del sector civil, de infraestructura y construcción. La finalidad de este proyecto es la de desarrollar una página web donde los usuarios (constructores afiliados a la organización) podrán obtener certificaciones en base a sus conocimientos y a su experiencia laboral. El proceso general que cualquier usuario debe seguir para convertirse en usuario certificado, es el siguiente (figura 1.1):

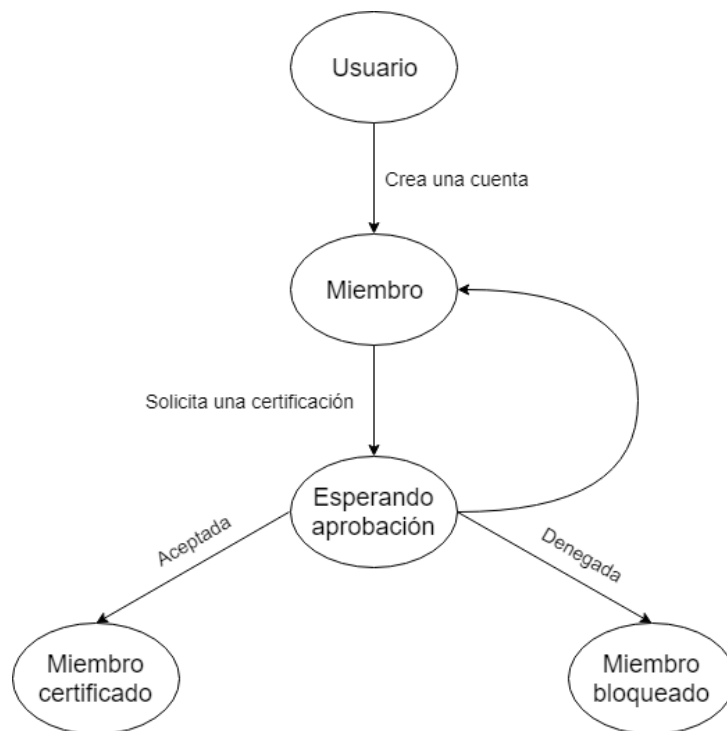


Figura 1.1: Proceso mediante el cual un usuario se convierte en miembro certificado

Realizar un proyecto de tal envergadura solo, conlleva más tiempo del disponible en la estancia de prácticas. Es por ello que este proyecto está dividido en 2 partes: el backend y el frontend. La propuesta que yo he escogido es la de desarrollar el frontend, donde más adelante veremos que me divido la faena con Pieter, programador frontend en E-Force ICT (Holanda). En cuanto al backend y la base de datos, se ocupará mi supervisor.

Para entender un poco la estructura que va a tener la página web, voy a explicar brevemente las 3 áreas que la componen:

- Área pública: cualquier usuario tiene acceso a esta área. En ella podemos encontrar noticias, anuncios, información sobre los cursos, eventos y una pantalla de login.
- Área del miembro: solo los miembros tienen acceso a esta área. Es aquí donde el miembro tendrá la opción de editar su perfil, añadir conocimientos y experiencia laboral, solicitar una certificación, etc.

- Área de administración: solo tendrán acceso los usuarios con roles de administración. En ella se podrá editar toda la información visible en el área pública; manejar cuentas, compañías y conocimientos; y obtener y editar información sobre los miembros, entre otras cosas.

1.1.3. Proyecto propuesto por la empresa

El proyecto propuesto por la empresa y el cual acabé escogiendo, ha sido el de desarrollar el frontend de la aplicación web de Constructor. Existía otra propuesta para Constructor en la parte backend, que debía coordinarse con la mía, pero en apartados posteriores veremos como esta propuesta no fue elegida por ningún otro estudiante.

1.1.4. Necesidad y utilidad del proyecto

Debido a que actualmente en la empresa solo son dos programadores (contando a mi supervisor Jose Antonio Cano), les surgió la necesidad de encontrar a alguien que se encargara de la parte frontend de este proyecto. El tiempo que lleva utilizándose ReactJS dentro de la empresa es muy poco y no disponían del tiempo suficiente como para desarrollar un frontend de calidad. Es por ello que decidieron lanzar esta propuesta para los estudiantes del grado, los cuales tendrían tiempo de sobra para aprender a usar ReactJS y programar código fiable y de calidad. Respecto al backend, no necesitaban con tanta urgencia un desarrollador, ya que mi supervisor domina a la perfección .NET y ha sido capaz de cumplir con los plazos dedicándole el tiempo mínimo.

1.1.5. ¿Por qué elegí este proyecto?

Durante todos estos años de grado, he trabajado constantemente con lenguajes como Java o Python. Las prácticas eran una oportunidad muy grande para probar lenguajes de programación nuevos y aprender nuevas formas de trabajo. Hace unos años que estoy aprendiendo y programando aplicaciones móviles en mi tiempo libre, ya que considero que tienen una gran utilidad y que son el presente y futuro. Muchas de las propuestas que habían para realizar la estancia, eran para desarrollar una aplicación móvil, ya fuera en Java, React Native o Xamarin. Muchas otras eran para desarrollar una aplicación web, tanto en Javascript como en PHP. Debido a que ya me había metido en el mundo móvil hace un tiempo, decidí escoger esta vez algún proyecto relacionado con el desarrollo web. Consideré que era una oportunidad muy buena para aprender un nuevo paradigma y enfrentarme a problemas que no había tenido anteriormente. Ya hacía tiempo que quería probar a desarrollar una aplicación web y entre mis opciones se encontraban las siguientes: Angular.js, React.js o Vue.js. Terminé escogiendo el proyecto que ofrecía E-Force, ya que cumplía con los requisitos que yo necesitaba y además, la empresa está situada en un lugar óptimo como es el Españtec.

1.2. Objetivos del proyecto

En este apartado voy a explicar los objetivos del proyecto, separándolos en objetivos técnicos del proyecto Constructor, objetivos técnicos dentro del proyecto propuesto por la empresa y objetivos personales.

1.2.1. Objetivos técnicos del proyecto Constructor

Una vez finalizado el proyecto Constructor, estos son los objetivos que la empresa espera haber logrado:

- Desarrollo de un backend de calidad, que contenga toda la funcionalidad especificada por el cliente.
- Desarrollo de un frontend de calidad: buen diseño de interfaces, fácil navegación a través de la web, gestión correcta de errores y excepciones, etc.
- Diseño de una base de datos que albergue toda la información que se encuentra en la base de datos del antiguo proyecto Constructor.
- Mejora de las metodologías utilizadas en técnicas de colaboración y comunicación: intra equipos, para mejorar la colaboración entre los integrantes de los equipos de desarrollo; inter equipos, para mejorar la colaboración y el intercambio entre equipos físicamente distantes entre sí (E-Force Spain y E-Force ICT).
- Aprendizaje de una nueva tecnología: ReactJS.

1.2.2. Objetivos técnicos de mi proyecto

Mi proyecto consta de desarrollar parte del frontend de la aplicación web. En esta sección explicaré con más detalle qué objetivos técnicos espera la empresa que se cumplan en este proceso.

- Utilizar las mejores prácticas de ReactJS para el desarrollo de la página web.
- Mejorar el estilo de las interfaces de la página web antigua.
- Mejorar la funcionalidad y navegación de la página web antigua.
- Aplicar buenas prácticas de colaboración y comunicación, tanto con mis compañeros dentro de la empresa, como con el personal trabajando en Holanda.

1.2.3. Objetivos personales

Los objetivos personales del proyecto son los que se citan a continuación:

- Aprender la biblioteca ReactJS.
- Aprender a desarrollar una página web de calidad utilizando esta biblioteca.
- Colaborar con los compañeros que trabajan sobre el mismo proyecto e integrarme.

Teniendo en cuenta los objetivos anteriores, los resultados esperados una vez acabada la estancia en la empresa, son los siguientes:

- Comprender el funcionamiento de ReactJS.
- Tener fluidez tanto en ReactJS como en el IDE que se va a utilizar.
- Interiorizar las mejores prácticas de colaboración y sincronización entre desarrolladores de software.

Una vez acabadas las prácticas, puedo decir que he logrado casi al completo los resultados mencionados. He aprendido mucho sobre ReactJS y las aplicaciones web en general, gracias en gran parte a mis compañeros de trabajo, tanto aquí como en Holanda. También he aprendido buenas prácticas a la hora de desarrollar un proyecto de tal envergadura.

1.3. Estructura de la memoria

Este apartado puede servir para tener una visión general de la estructura de la memoria. En el capítulo 2 se presenta una descripción a fondo de las principales tecnologías utilizadas dentro del proyecto, así como la situación inicial del proyecto. El capítulo 3 muestra la metodología, planificación y seguimiento que se ha realizado en el proyecto. En el capítulo 4 hablaré del análisis completo del sistema, dónde especificaré los requisitos principales y el modelado del sistema; el diseño de la arquitectura del sistema, que incluye los componentes del sistema y su interrelación; y el diseño de las interfaces. El capítulo 5 trata sobre los detalles de la implementación, la verificación y la validación. Por último, el capítulo 6 incluye las conclusiones obtenidas.

Capítulo 2

Descripción del proyecto

2.1. Tecnologías y herramientas usadas en el proyecto

Normalmente, cuando desarrollamos un proyecto de media o gran envergadura, utilizamos gran cantidad de tecnologías y herramienta que nos facilitan el trabajo. Actualmente, existen gran cantidad de frameworks o bibliotecas para Javascript que nos ayudan a crear interfaces web de una manera menos tediosa, los cuales es aconsejable utilizar. También es muy recomendable utilizar un IDE¹ o un editor de código fuente, los cuales amenizan el desarrollo de software gracias a sus editores de código fuente, herramientas de construcción automáticas y técnicas de depuración. Como veremos a continuación, ciertos programas o plugins son también muy útiles para resolver problemas técnicos concretos. Dicho esto, voy a explicar detalladamente cuales han sido las tecnologías y herramientas escogidas para el desarrollo del proyecto. Además, he incluido .NET Core (sección 2.1.13) y SQL Server (sección 2.1.14), ya que son dos pilares fundamentales dentro del proyecto Constructor.

2.1.1. React

React es una biblioteca JavaScript de código abierto utilizada para construir interfaces de usuario. Es mantenida por Facebook, Instagram y una comunidad de desarrolladores independientes. React fue elegida por E-Force para el desarrollo de interfaces web, ya que encajaba perfectamente con el estilo de la empresa: es simple, declarativa, fácil de combinar y utiliza el patrón de diseño MVC². Las características por las que React se ha vuelto tan utilizada en el mundo web, son las siguientes [5]:

- Permite crear interfaces de usuario interactivas de una manera menos tediosa.
- Actualiza y renderiza los componentes adecuados cuando la información cambia, gracias al diseño de vistas simples para cada estado de la aplicación.

¹Entorno de desarrollo integrado

²MVC o Model View Controller es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica en tres componentes distintos.

- Permite escribir código más predecible y fácil de depurar.
- Permite construir componentes encapsulados que manejan su propio estado, dejándolo fuera del DOM³.

2.1.2. Bootstrap

Bootstrap es el framework más popular del mundo a la hora de crear páginas web responsive⁴. Proporciona formularios, botones, menús de navegación y muchos otros elementos con un estilo predefinido muy visual y atractivo. En el proyecto estamos utilizando React-Bootstrap, una biblioteca que ha reconstruido Bootstrap para ser utilizado en React. Esta biblioteca utiliza Bootstrap 3 a día de hoy, pero están trabajando activamente para actualizar a Bootstrap 4.

2.1.3. Visual Studio Code

Visual Studio Code es un editor de código fuente ligero pero muy potente. Está disponible para Windows, Linux y MacOS. Viene con soporte integrado para Javascript, que es lo que nos interesa en este proyecto [1]. Todos los programadores que estamos en el mismo proyecto, hemos usado Visual Studio como editor de código fuente (ya sea para el frontend en React, como el backend en .NET).

2.1.4. Postman

Postman surgió como una extensión para el navegador Google Chrome. Debido al gran éxito que tuvo, se lanzaron aplicaciones de escritorio tanto para Windows como para Mac. Esta herramienta permite realizar gran variedad de tareas dentro del mundo de las API, de las cuales me gustaría destacar las siguientes:

- Crear peticiones a APIs internas o de terceros.
- Crear tests para validar el comportamiento de las APIs.
- Crear diferentes entornos de trabajo.
- Desarrollar colecciones para APIs de manera colaborativa.

³Document Object Model.

⁴Técnica de diseño web que busca la correcta visualización de una misma página en diferentes dispositivos.

2.1.5. Consola de Chrome

La consola de Chrome es una herramienta accesible desde el navegador de Google Chrome, muy útil para el desarrollo de aplicaciones web. Permite inspeccionar los elementos html de la página web y visualizar los logs y excepciones del código implementado. También permite simular diferentes tipos de red como pueden ser offline, 3G rápido o 3G lento. Esta simulación es de mucha utilidad para comprobar el correcto funcionamiento de la aplicación ante diversas conexiones de red.

2.1.6. Sourcetree

Sourcetree es un cliente gratuito de Mercurial y Git para Windows y Mac que ofrece una interfaz gráfica para los repositorios de Hg⁵ y Git. Este software simplifica la forma en la que se interactúa con los repositorios [6]. Algunas de las ventajas de usarlo en grandes equipos de desarrollo son las siguientes:

- Es sencillo para los principiantes, lo que permite poner al día a todo el equipo de desarrollo con rapidez.
- Es potente para los expertos, lo que conlleva una mayor productividad.
- Puedes visualizar el código de manera sencilla.

2.1.7. Xampp

Xampp es una distribución de Apache completamente gratuita que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache, PHP y Perl. También incluye una herramienta llamada phpMyAdmin, que sirve para administrar bases de datos MySQL a través de una página web. Instalé Xampp básicamente porque necesitaba MySQL y tener acceso a la base de datos de la antigua página web del proyecto a desarrollar. Me dieron las credenciales para la base de datos en producción y con phpMyAdmin pude visualizar las tablas y las filas que las componían.

2.1.8. React developer tools

React developer tools es una extensión de Chrome DevTools creada por Facebook (al igual que React). Permite inspeccionar las jerarquías de componentes de un proyecto en React. Me ha servido bastante, sobre todo al principio, para observar el flujo de trabajo de la aplicación componente a componente.

⁵Mercurial

2.1.9. Bitbucket

Bitbucket es un sistema de control de versiones distribuido que facilita la colaboración con el equipo. Bitbucket es considerada como la solución Git para equipos profesionales, ya que dispone de todo lo mencionado a continuación [2]:

- Permite aprobar revisiones de código de otros usuarios mediante pull requests con total eficacia (incluyendo conversaciones directamente en el código fuente).
- Permite disponer de repositorios privados y públicos ilimitados para equipos pequeños de 5 miembros.
- Permite controlar las acciones que pueden realizar los usuarios sobre los proyectos, repositorios o ramas.

2.1.10. Slack

Slack es una herramienta de comunicación y colaboración en equipo en un mismo lugar. Slack es una gran herramienta para trabajar en equipo, ya que está diseñada exclusivamente para ello. Ofrece una serie de características que la hacen imprescindible en cualquier equipo:

- Los canales pueden ser ordenados por equipo, proyecto, cliente o según las necesidades de la organización.
- Los hilos permiten tener conversaciones paralelas para no entorpecer el tema o el proyecto principal.
- Permite la colaboración con miembros externos a tu equipo, como por ejemplo, empresas con las que trabajas habitualmente.
- Dispone de llamadas de voz o vídeo directamente desde la aplicación, permitiendo compartir pantalla.
- Permite compartir archivos como PDFs, imágenes o vídeos sin salir de Slack.

2.1.11. OneDrive

OneDrive es un servicio de alojamiento de archivos en la nube. Permite acceder a los archivos desde cualquier dispositivo, sin conexión y siempre están seguros. Una de las grandes ventajas de OneDrive y la razón por la cual lo hemos utilizado en el proyecto, es la posibilidad de compartir archivos y colaborar.

2.1.12. Escritorio remoto de Chrome

Escritorio remoto de Chrome es una aplicación que te permite acceder de manera remota a otro ordenador, o que otra persona acceda a tu ordenador. Todo esto se consigue de forma segura a través de Internet.

2.1.13. .NET Core

.NET Core es una implementación del framework .NET para uso general, modular, multi-plataforma y de código abierto. Contiene solo un conjunto de las API de .NET, consiguiendo de esta manera una carga de trabajo menor [?]. Esta opción ha sido la elegida por Jose, mi supervisor, para desarrollar el backend del proyecto.

2.1.14. SQL Server

SQL Server es un SGBD⁶ desarrollado por Microsoft, que utiliza el modelo relacional. Ha sido elegido para el proyecto por las siguientes razones:

- Soporta transacciones.
- Es escalable, estable y seguro.
- Soporta procedimientos almacenados.
- Incluye un entorno gráfico de administración muy potente.

Aunque no he trabajado en SQL Server durante el proyecto, he estado ayudando en temas de bases de datos bastantes tiempo. En la sección 4.2.2 explico el proceso que se ha seguido para migrar la base de datos MySQL del antiguo proyecto, a SQL Server, ya que considero que es un punto fundamental.

⁶Sistema de gestión de bases de datos

2.2. Situación inicial del proyecto

La situación inicial en la que se encontraba el proyecto cuando yo empecé la estancia en prácticas, era la siguiente:

- El desarrollo del backend aún no se había empezado.
- El diseño y construcción de la base de datos aún no se habían empezado.
- El desarrollo del frontend se encontraba en el siguiente estado:
 - Diseño de las interfaces del área pública finalizado.
 - Diseño de las interfaces del área del miembro casi finalizado.
 - Diseño de las interfaces del área de administración sin empezar.

El frontend no tenía nada de funcionalidad implementada: no se hacía nada con la información de los formularios, la información que se mostraba no era real, no se conectaba con el backend ni la base de datos, etc. Es por esto que mi proyecto se encontraba en una situación inicial que considero como "no empezado". La parte del proyecto correspondiente al diseño de las interfaces no entraba en el alcance de mi proyecto, aunque en apartados posteriores veremos cómo acabé diseñando algunas de las interfaces de la página web.

Capítulo 3

Planificación del proyecto

3.1. Metodología

En esta sección voy a explicar con detalle cuál ha sido la metodología que hemos empleado en el desarrollo del proyecto. Esto incluye tanto la forma de trabajar durante la planificación previa del proyecto como durante el desarrollo del mismo.

3.1.1. Scrum

Scrum es un proceso en el cual se aplican buenas prácticas para el trabajo en equipo, para obtener el mejor resultado posible del proyecto. La figura 3.1, muestra las actividades que se realizan durante el proceso:

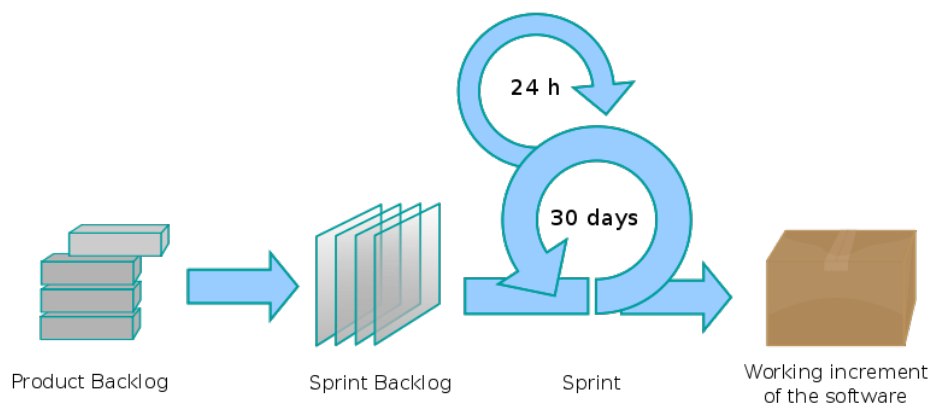


Figura 3.1: Actividades del proceso Scrum

En Scrum se realizan entregas parciales y regulares del producto final, priorizando los beneficios que puedan aportar al cliente. Se aconseja utilizar Scrum en entornos complejos, donde la

necesidad de obtener resultados pronto es vital y donde los requisitos no están del todo definidos o cambian constantemente. También se suele utilizar en situaciones donde el cliente no recibe lo que necesita, las entregas no cumplen con los plazos, los costes son elevados o la calidad no es la esperada [7].

En la empresa se ha elegido esta metodología ágil para el desarrollo del proyecto, porque es la que se lleva usando desde hace bastante tiempo y la que mejor resultados está dando. Los detalles del proceso seguido son los siguientes:

1. Cada iteración (sprint) tenía una duración de 3 semanas.
2. El primer día de cada sprint, Pieter se reunía con el cliente para realizar una planificación del sprint. El cliente le presentaba la lista de requisitos del proyecto y seleccionaban los más prioritarios, comprometiéndose a completarlos en ese sprint. Entre Pieter, Jose y yo elaborábamos la lista de tareas del sprint necesarias para cumplir los requisitos y nos asignábamos las tareas (Pieter, el diseño de las interfaces; Jose, la API; y yo, la funcionalidad del frontend).
3. Casi todos los días, Jose y yo realizábamos una reunión de sincronización de unos 10 minutos. Él me enseñaba las llamadas HTTP que había realizado y las que no, ya que yo era el encargado de conectar el frontend con el backend. Si hacía falta adaptar alguna llamada a la API, lo hablábamos en estas reuniones. Además, Jose y Pieter realizaban reuniones cada 2 o 3 días mediante llamadas telefónicas, donde se informaban de la funcionalidad cubierta hasta la fecha.
4. El último día del sprint, Pieter presentaba a los clientes los requisitos completados durante ese sprint mediante un incremento del producto. De manera objetiva, el cliente realizaba las adaptaciones necesarias en función de los resultados mostrados, para replanificar el proyecto.

Una vez detallado el proceso general de cualquiera de las iteraciones realizadas durante el desarrollo del proyecto, puedo decir que estas han sido las limitaciones que he tenido:

1. No podía elegir los requisitos a realizar durante cada iteración.
2. No podía hablar con el cliente directamente.

Cumpliendo la función de Scrum Master, mi supervisor se encargaba de eliminar los obstáculos que yo no podía resolver por mí mismo, aumentando así mi productividad y compromiso.

3.2. Planificación

Antes de comenzar a programar, es muy importante hacer una planificación inicial de todas las tareas a realizar, para tener una idea general e intentar ir cumpliendo los tiempos marcados. A continuación, podemos ver la tabla de tareas (3.1), donde especifico todas las tareas incluidas en la planificación, con su correspondiente desglose en subtareas y dedicación prevista:

Nº	Tareas	Tiempo (h)	Dependencias
1	Desarrollo de la propuesta técnica		
1.1	Inicio	9 h	
1.1.1	Definir proyecto con tutor y supervisor	1	
1.1.2	Definir metodología de trabajo	4	1.1.1
1.1.3	Definir flujo de trabajo	4	1.1.2
1.2	Interiorizar la idea del proyecto	12 h	
1.2.1	Leer documentación del proyecto	6	1.1
1.2.2	Investigar sobre las tecnologías propuestas	6	1.2.1
1.3	Planificar el proyecto	12 h	
1.3.1	Definir las tareas iniciales y estimar fechas	2	1.2
1.3.2	Crear diagrama de Gantt	2	1.3.1
1.3.3	Desarrollar la propuesta técnica	8	1.3.2
1.3.4	Entregar la propuesta técnica	0	1.3.3
2	Desarrollo técnico del proyecto		
2.1	Definir requisitos del proyecto	8 h	
2.1.1	Definir y documentar los requisitos de datos	4	1.3
2.1.2	Definir los requisitos tecnológicos	4	2.1.1
2.2	Análisis	8 h	
2.2.1	Documentar las clases	4	2.1
2.2.2	Validar el análisis	4	2.2.1

Cuadro 3.1: Desglose de tareas, junto con la planificación temporal y las dependencias entre tareas (Tabla 1 de 2)

Nº	Tareas	Tiempo (h)	Dependencias
2.3	Diseño	8 h	
2.3.1	Identificar y clasificar los diferentes usuarios	4	2.2
2.3.2	Validar el diseño	4	2.3.1
2.4	Desarrollar el producto		
2.4.1	Estudio de las tecnologías	50 h	
2.4.1.1	ReactJS	40	
2.4.1.2	Redux	10	2.4.1.1
2.4.2	Configuración	4 h	
2.4.2.1	Instalación y configuración de VS Code	2	2.4.1
2.4.2.2	Sincronización del proyecto en Bitbucket	1	2.4.2.1
2.4.2.3	Conexión con la base de datos (SSMS)	1	2.4.2.2
2.4.3	Programación	183 h	
2.4.3.1	Programación de los formularios	100	2.4.2
2.4.3.2	Conexiones con la API	40	2.4.3.1
2.4.3.3	Programación de las tareas de administración	43	2.4.3.2
2.5	Puesta en marcha	6 h	
2.5.1	Implantación	3	2.4.3
2.5.2	Formación	3	2.5.1
2.5.3	Entrega final	0	2.5.2
3	Documentación y presentación del TFG	150 h	
3.1	Redacción de los informes quincenales	12	
3.2	Redacción de la memoria técnica	100	2.5
3.3	Entrega de la memoria técnica	0	3.2
3.4	Preparación de la presentación oral	37	3.3
3.5	Presentación oral	1	3.4

Cuadro 3.2: Desglose de tareas, junto con la planificación temporal y las dependencias entre tareas (Tabla 2 de 2)

Junto con esta tabla de tareas, he generado un diagrama de Gantt. Se han adaptado los tiempos para que la estancia sea de 300 horas, para que la memoria sea entregada el día oficial de la entrega y la presentación se realice el día indicado.

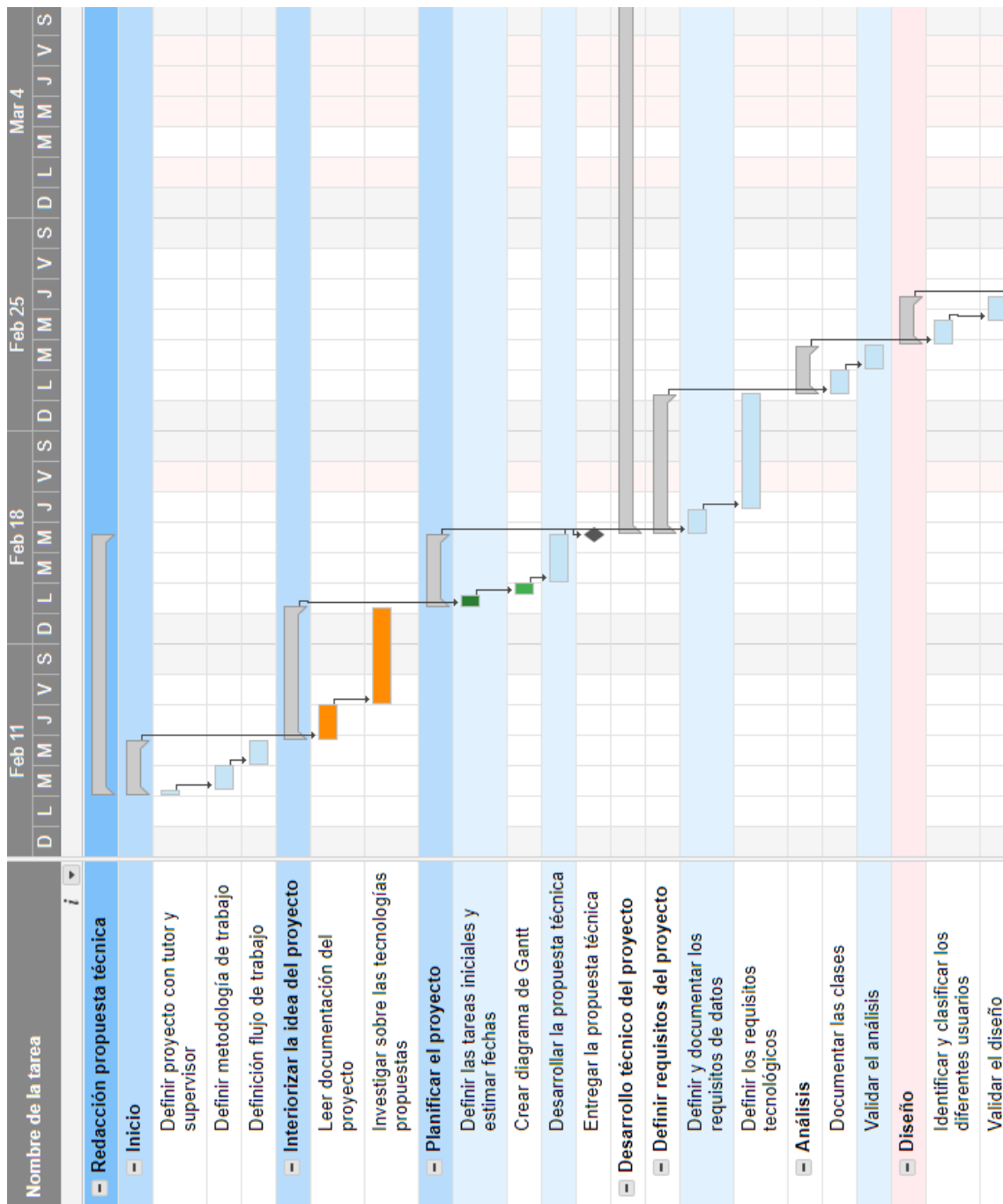


Figura 3.2: Diagrama de Gantt (parte 1 de 3)

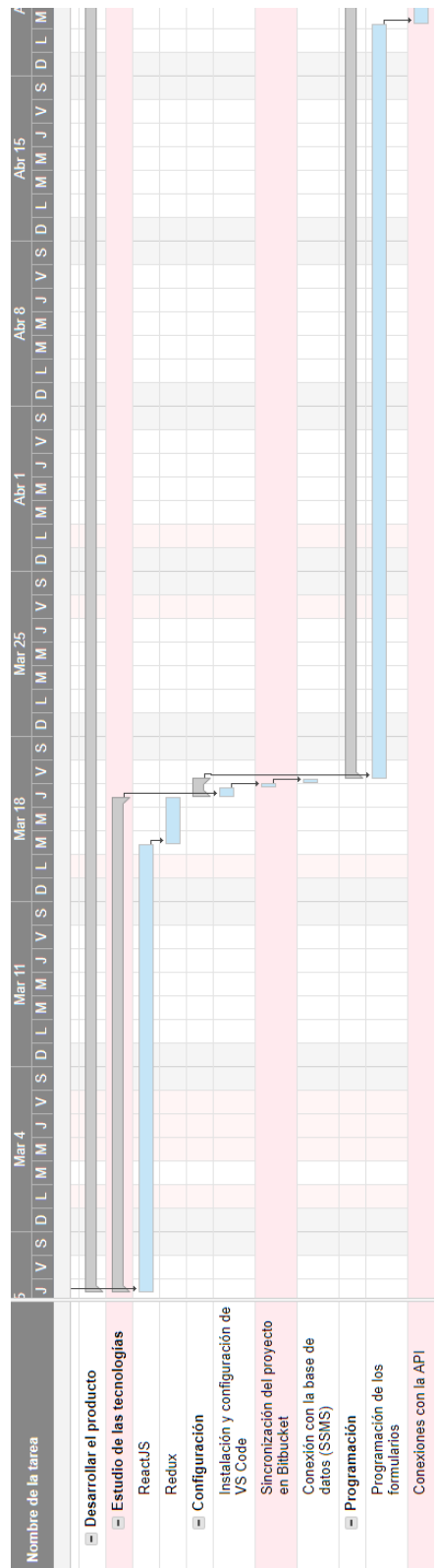


Figura 3.3: Diagrama de Gantt (parte 2 de 3)

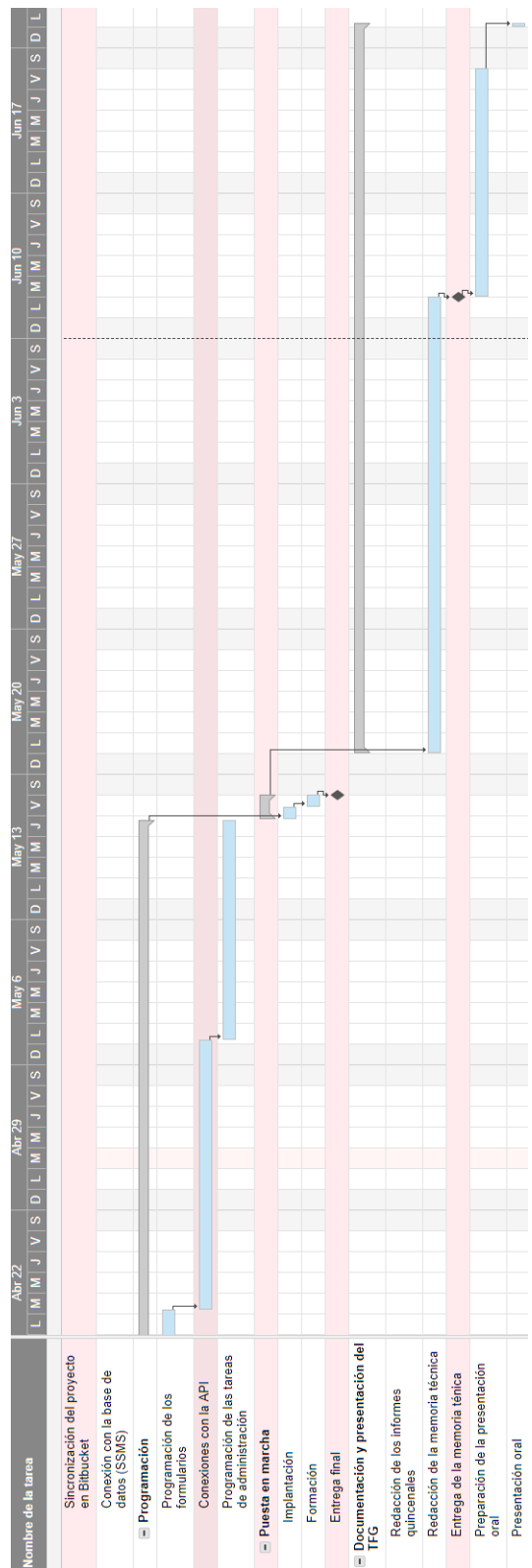


Figura 3.4: Diagrama de Gantt (parte 3 de 3)

3.3. Estimación de recursos y costes del proyecto

Para la realización del proyecto, tuve que realizar una estimación inicial de los recursos necesarios, así como de los costes del proyecto. He separado los recursos en diferentes tipos y he estimado los costes monetarios.

3.3.1. Estimación de recursos

Al principio de la estancia en E-Force, realicé una estimación inicial de los recursos software, hardware y humanos que iba a necesitar. Los recursos software son los siguientes:

- IDE Visual Studio Code: para el desarrollo del proyecto web utilizando ReactJS.
- SQL Server Management Studio (SSMS) de Microsoft: para trabajar sobre la base de datos del proyecto.
- Sourcetree: ofrece una interfaz gráfica para nuestro repositorio Git.
- Postman: ayuda a trabajar con las peticiones a la API.
- Sublime Text 3: ofrece una rápida edición y visualización de extractos de código fuente no muy extensos.

Por otro lado, estos son los recursos hardware estimados:

- HP Pavilion g6 Notebook PC con el sistema operativo Microsoft Windows 10 Education.
- Monitor de 27 pulgadas: es el monitor del supervisor pero lo podía utilizar cuando él no se encontraba en la oficina. Me ayudó a optimizar el tiempo y desarrollar de manera más amena.

Por último, los recursos humanos previstos para el proyecto son los siguientes:

- Pablo Berbel (yo), como desarrollador en la parte frontend.
- Jose Cano, como supervisor y desarrollador en la parte backend.
- Pau Segarra, como compañero y asesor.
- Pieter van Uhm, como desarrollador en la parte frontend.

3.3.2. Estimación de costes

Durante el proyecto, voy a ocupar 3 tipos de roles:

- Analista: encargado de definir los requisitos del proyecto e identificar los diferentes tipos de usuarios.
- Desarrollador: encargado de la parte de programación frontend.
- Diseñador de base de datos: encargado de estudiar la base de datos y extraer conclusiones.

Basándome en el desglose de tareas y en los sueldos medios obtenidos en la página web Indeed [4], este sería el coste del personal:

1. Analista: 24 horas. El sueldo medio anual de un analista ronda los 28.000 €. Atendiendo a jornadas laborales de 40 horas al día, el sueldo por hora aproximado es de 13.89 €. El coste total del rol de analista es de $24 \times 13.89 = 333,36$ €.
2. Desarrollador: 240 horas. El sueldo medio anual de un desarrollador junior es de aproximadamente 18.000 €. El sueldo por hora es de unos 9 €. El coste total del rol de desarrollador es de $250 \times 9 = 2.250$ €.
3. Diseñador de bases de datos: 20 horas. El sueldo medio anual de un diseñador de base de datos ronda los 25.000 €. El sueldo por hora sería de y unos 12.40 €. El coste total del rol de diseñador de bases de datos es de $20 \times 12.40 = 248$ €.

El coste total del personal es de $333,36 + 2.250 + 248 = 2.831,36$ €.

En cuanto al software utilizado (Visual Studio Code, Postman, SSMS, Sourcetree y Sublime Text 3), es completamente gratuito, por lo que el coste del software es de 0 €.

Por último, este sería el coste del hardware:

1. HP Pavilion g6 Notebook PC con el sistema operativo Microsoft Windows 10 Education: 570 €.
2. Monitor de 27 pulgadas: 200 €.

El coste total del hardware es de $570 + 200 = 770$ €.

Una vez obtenidos los costes por separado del personal, el software y el hardware, podemos calcular el coste total del proyecto. Este coste es igual a 3601,36 €

3.4. Seguimiento del proyecto

3.4.1. Control del proyecto

La dedicación realizada durante el transcurso de todo el proyecto ha sido la misma prácticamente todos los días: 5 horas diarias de lunes a viernes, empezando a las 9:00 y acabando a las 14:00. El horario era flexible, por lo que algún día entraba más tarde y salía más tarde o venía por la tarde para cogermme algún día libre. El requisito principal era cumplir con las 300 horas establecidas al inicio de las prácticas e intentar ser siempre productivo mientras me encontraba en la empresa. La estancia empezó el 13 de febrero y se fijó la fecha final para el 25 de mayo, siempre y cuando se atendiera a una dedicación de 5 horas diarias en los días lectivos. Se han usado varias técnicas para el seguimiento del proyecto, así como herramientas para la colaboración, sincronización y comunicación. Estas herramientas están explicadas con detalle en las secciones 2.1.9 (Bitbucket), 2.1.10 (Slack), 2.1.11 (OneDrive) y 2.1.12 (Escritorio remoto de Chrome).

Para el proyecto Constuctor disponemos de dos repositorios privados alojados en Bitbucket, uno con el proyecto backend y otro con el frontend. En la figura 3.5 se pueden ver ambos repositorios, pertenecientes al equipo eforceteam:

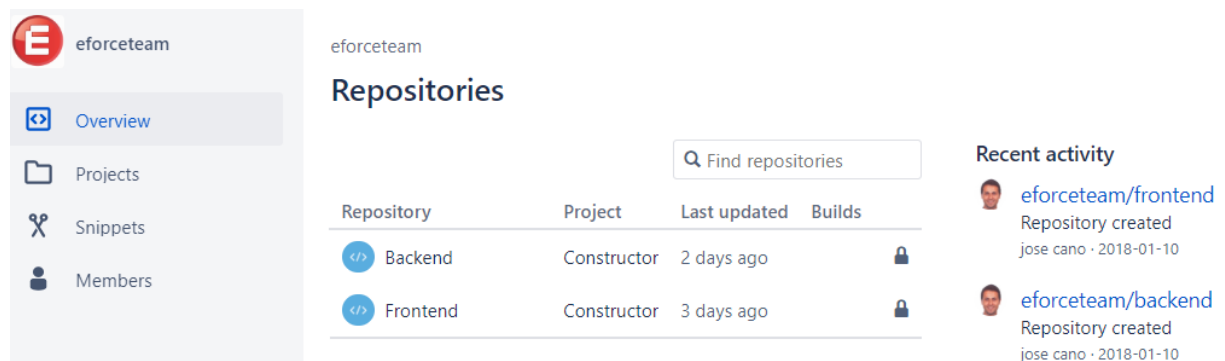


Figura 3.5: Repositorios del equipo eforceteam

Jose trabajaba sobre el repositorio Backend, mientras que Pieter y yo trabajábamos en el repositorio Frontend. Cada vez que cubríamos una nueva funcionalidad definida por los requisitos del cliente, subíamos los cambios al repositorio. Debido a que Pieter y yo trabajábamos en cosas diferentes del frontend, cada uno teníamos nuestra propia rama de desarrollo dentro del repositorio. En la figura 3.6 se pueden observar todas las ramas creadas para el proyecto Frontend:

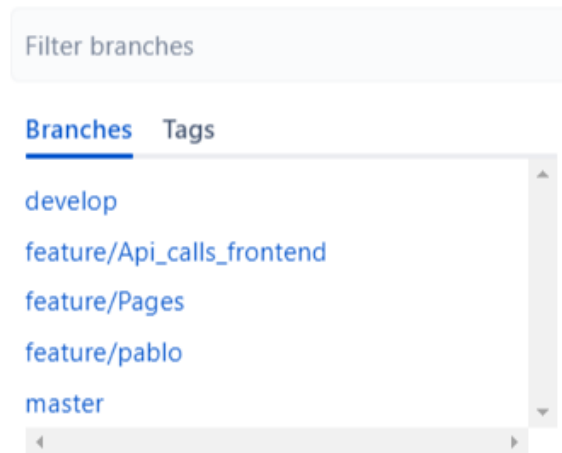


Figura 3.6: Ramas del repositorio Frontend

Bitbucket me ha servido mucho para realizar un seguimiento correcto del proyecto. Gracias a este VCS¹ he podido estar al tanto de los cambios realizados en el frontend por parte de Pieter. También he podido ir viendo como avanzaba el proyecto en la parte del backend.

E-Force dispone de un espacio de trabajo en Slack donde estábamos todos los miembros de la empresa. En la figura 3.7 se pueden observar los diferentes canales disponibles dentro del espacio de trabajo, así como las personas que pertenecíamos al mismo:

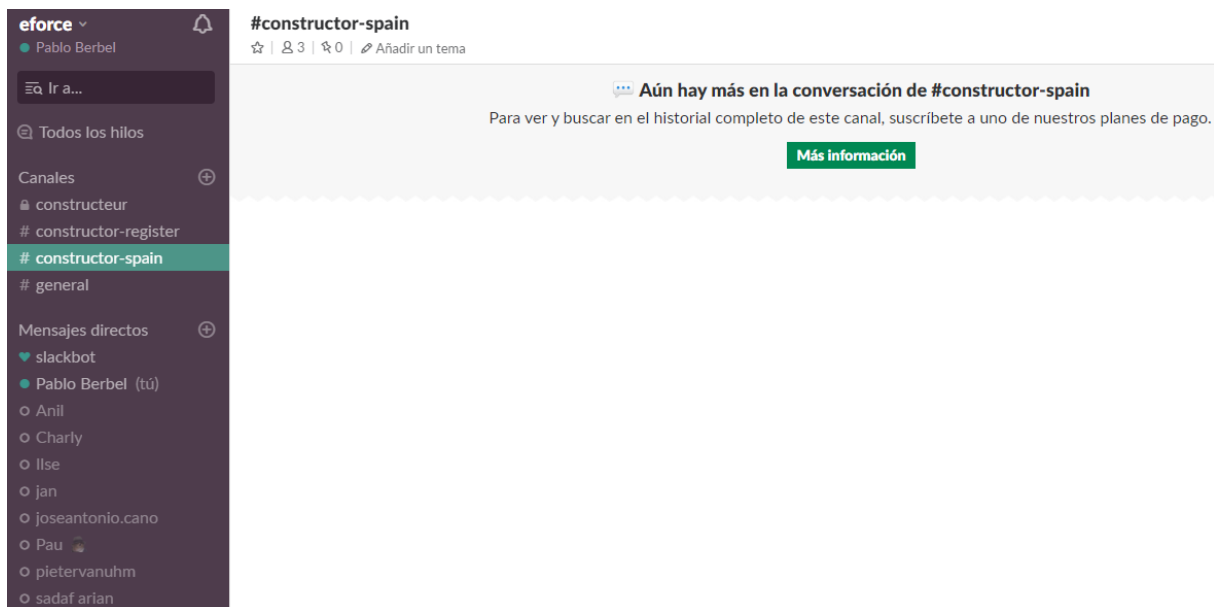


Figura 3.7: Espacio de trabajo de E-Force

¹Version Control System

Esta herramienta ha sido muy útil para la comunicación con el resto de compañeros que trabajaban sobre el proyecto Constructor. Para facilitar la comunicación por temas o partes del proyecto, se crearon diversos canales. Uno de los usos más importantes que le he dado a Slack, es el de compartir documentos o extractos de código en los canales o en mensajes directos a los compañeros.

Todas las personas que estábamos trabajando sobre el proyecto Constructor (incluido el cliente), teníamos acceso a un documento de OneDrive, que contenía lo siguiente:

- Preguntas de los desarrolladores que necesitan respuesta del cliente.
- Respuestas del cliente a preguntas realizadas por los desarrolladores.
- Comentarios técnicos de los desarrolladores.
- Errores que necesitan ser arreglados.
- Detalles de corrección de los errores.

Este documento me ha sido de gran utilidad tanto para resolver dudas durante el desarrollo del proyecto como para redactar la memoria técnica.

Durante el transcurso del proyecto, había veces que me costaba entender el funcionamiento de algún componente de la web, ya fuera por el lenguaje o por complejidad. Gracias al escritorio remoto de Chrome (otra de las herramientas utilizadas durante todo el proyecto), Pieter podía resolverme las dudas desde Holanda, mostrándome la pantalla de su ordenador.

3.4.2. Desviaciones de la planificación

Durante las dos últimas semanas de marzo, hubo una desviación sobre la planificación inicial. Yo había terminado tanto el área pública como el área del miembro de mi proyecto y no tenía más tareas asignadas. Pieter, el encargado de diseñar las interfaces de la web, no había diseñado aún las nuevas para la parte de administración. Debido a esto, tuve que dejar de lado mi proyecto y realizar otras tareas hasta que estas interfaces estuvieran disponibles. Hasta la fecha, había preparado todos formularios de estas dos áreas, pero aún no se conectaban con el API. El encargado del backend aún no tenía disponibles los endpoints² de la API, por lo que yo no podía avanzar más en el frontend.

La tarea que se me encomendó hasta que estuvieran disponibles las interfaces del área de administración o los endpoints del backend, fue la de investigar la base de datos de la página web antigua. La principal tarea fue averiguar la correspondencia entre los datos de la base de datos y los datos mostrados en la web antigua. Debido a que tanto la base de datos como la web están en holandés, ha sido complicado ver de donde procedían algunos datos y donde se almacenaban otros. Se me concedió acceso a la base de datos que se utilizaba en la anterior página web de Constructor para que pudiera investigar y hacer consultas e inserciones. Fue

²URL de un API o un backend que responde a una petición

una tarea tediosa, ya que algunas tablas contenían muchos campos, estaban en un lenguaje que desconozco (holandés) y no estaba claro con qué formularios de la web estaban relacionadas. Era vital conocer de donde procedían los datos, ya que el nuevo sistema tenía que almacenar la misma información que la alojada en la base de datos a la que se me concedió acceso. En la figura 3.8 se puede observar la base de datos de la que hablo, administrada a través de la herramienta phpMyAdmin:

The screenshot shows the phpMyAdmin interface for a database named 'cr_db' and a table named 'werkgebied'. The table contains 12 rows of data. The columns are 'id', 'code', 'omschrijving', and 'zichtbaar'. The data is as follows:

id	code	omschrijving	zichtbaar
1	NULL	Staal	1
2	NULL	Beton	1
3	NULL	Hout	1
4	NULL	Overig	1
5	NULL	Utiliteitsbouw	1
6	NULL	Industriebouw	1
7	NULL	Infra-kunstwerken	1
8	NULL	Funderingstechniek	1
9	NULL	Hoogbouw	1
10	NULL	Offshore	1
11	NULL	Natte waterbouw	1
12	NULL	Woningbouw	1

Figura 3.8: Base de datos antigua de Constructor

Capítulo 4

Análisis y diseño del sistema

En este capítulo se habla sobre el trabajo realizado durante el rol de analista y diseñador de la base de datos. Esto incluye el análisis del sistema, con los requisitos del sistema y la estructura del proyecto; el diseño de la arquitectura del sistema, con los componentes del sistema y la migración de la base de datos; el diseño de la interfaz, con unas cuantas imágenes del resultado final del proyecto.

4.1. Análisis del sistema

Esta sección describe los requisitos que debe cumplir el frontend de la página web, según el cliente. También describe la estructura del proyecto, en directorios y subdirectorios, que se ha utilizado a lo largo de todo el desarrollo.

4.1.1. Requisitos

Los requisitos indicados por el cliente para el proyecto Constructor, están expuestos en el documento compartido de OneDrive, mencionado en el apartado 3.4. Voy a basarme en este documento para extraer todos los requisitos que me fueron asignados y que he tenido que ir cumpliendo durante el desarrollo de mi proyecto.

Dentro del **área pública** de la página web, me asignaron los siguientes requisitos:

1. Footer: debe contener los contadores de los diferentes tipo de miembros. Los contadores del número total de miembros de cada tipo deben poder visualizarse en la parte inferior de la página web, denominada footer.
2. Eventos: el apartado de eventos de la página web, debe mostrar los eventos ordenados por fecha.
3. Cursos: el apartado de cursos de la página web, al pulsar sobre una institución, debe mostrar una cantidad concreta de cursos (ordenados por nombre). Para ver más cursos, se debe utilizar un sistema de paginación, en el cuál se deben incluir 10 cursos por cada página.

En cuanto al **área del miembro**, los requisitos definidos han sido los siguientes:

1. Perfil del miembro: se debe poder actualizar el perfil, así como cambiar la contraseña actual por otra nueva.
2. Cursos: tiene que haber dos tipos de búsqueda, una por número de referencia y otra por nombre de institución.
3. Experiencia laboral: al añadir experiencia laboral, el nivel debe seleccionarse a través de una lista.

Por último, dentro del **área de administración**, estos han sido los requisitos a cumplir:

1. Paginación: debe usarse un sistema de paginación para listar los miembros, las empresas, los cursos, los anuncios, los logs, etc.
2. Contenido de las páginas y anuncios: debe poderse editar el contenido de las páginas y los anuncios (texto predefinido), mediante un editor de texto enriquecido.

4.1.2. Diseño lógico de la base de datos

En esta sección voy a mostrar cuál era el diseño lógico de la base de datos. Este diseño ha sido el que he tenido que seguir durante todo el desarrollo, ayudándome a entender mejor los componentes del sistema y la relación entre ellos.

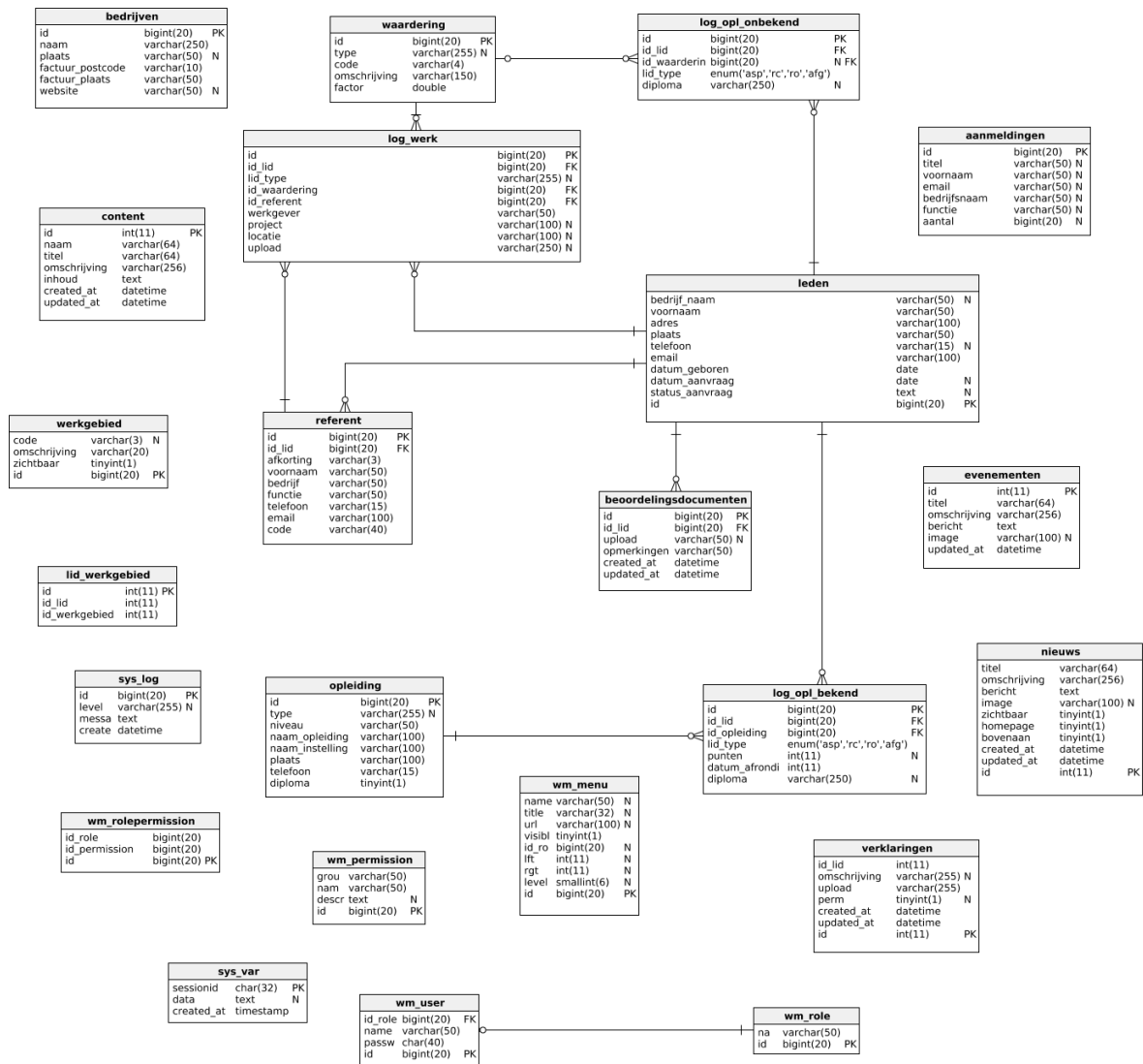


Figura 4.1: Diseño lógico de la base de datos

He intentado plasmar este esquema en la estructura del proyecto, de forma que cualquier persona que la visualizase, supiera donde encontrar cada componente. Existen más niveles dentro de la jerarquía de directorios, pero quiero hacer énfasis en los de más alto nivel. Dentro de **src** se encuentran varios directorios, siendo **components** el más importante al contener todos los componentes de la aplicación. La descripción y utilidad de cada directorio es la siguiente:

- **Admin**: contiene todos los componentes relacionados con el área de administración.
 - **Modules**: componentes que encapsulan toda la lógica correspondiente al área de administración.
 - **Pages**: componentes que contienen todos los componentes necesarios (ubicados en **Modules**) para renderizar una página completa.

- **Global:** contiene los componentes relacionados con el área pública (sin incluir las páginas) y de propósito general.
 - **Modules:** componentes que encapsulan la lógica correspondiente al área pública y acciones que necesitan realizarse desde muchos otros componentes (ubicados en diferentes directorios). Este directorio contiene componentes muy importantes como **Api.js**, que contiene funciones que son ejecutadas desde muchos componentes, evitando la duplicación de código. También contiene el componente **Login.js**, donde se realizan las operaciones necesarias para iniciar sesión en la página web; y el componente **User.js**, que almacena información sobre el usuario, como el nombre, el correo o el token para realizar posteriores peticiones.
- **Member:** contiene todos los componentes relacionados con el área del miembro.
 - **Modules:** componentes que encapsulan toda la lógica correspondiente al área del miembro.
 - **Pages:** componentes que contienen todos los componentes necesarios (ubicados en **Modules**) para renderizar una página completa.
- **NotFound:** contiene un único componente, utilizado cuando el servidor devuelve un código de respuesta 404 (Not found).
- **Public:** contiene todos los componentes relacionados con el área pública (sin incluir la lógica).
 - **Pages:** componentes que contienen todos los componentes necesarios (ubicados en **Global/Modules**) para renderizar una página completa.
- **ScrollToTop:** contiene un único componente, utilizado para realizar scroll.
- **App.js:** componente que almacena las rutas de todos los componentes, además de algunas funciones extra.

En la figura, se observan dos directorios más: **images** y **styles**. El primero contiene imágenes que se usan dentro de la aplicación, mientras que el segundo contiene los estilos que se usan en todo el proyecto. Por último, el componente **App.test.js** se utiliza para realizar pruebas de testing, y el componente **index.js** es el componente principal de la aplicación.

4.2. Diseño de la arquitectura del sistema

En esta sección se aborda el diseño de la arquitectura usada para desarrollar el sistema, el uso de la API backend por parte del frontend y la migración de la base de datos. El apartado de la arquitectura del sistema puede servir para entender mejor como están conectados todos los componentes del sistema, así como el proceso general que he realizado para conectar mi proyecto con la API. También describo cómo se ha realizado la migración de la base de datos antigua al nuevo sistema.

4.2.1. Arquitectura del sistema

El proyecto Constructor se basa en una arquitectura cliente-servidor tradicional. En la siguiente figura se puede observar la arquitectura utilizada y la interrelación entre los componentes del sistema:

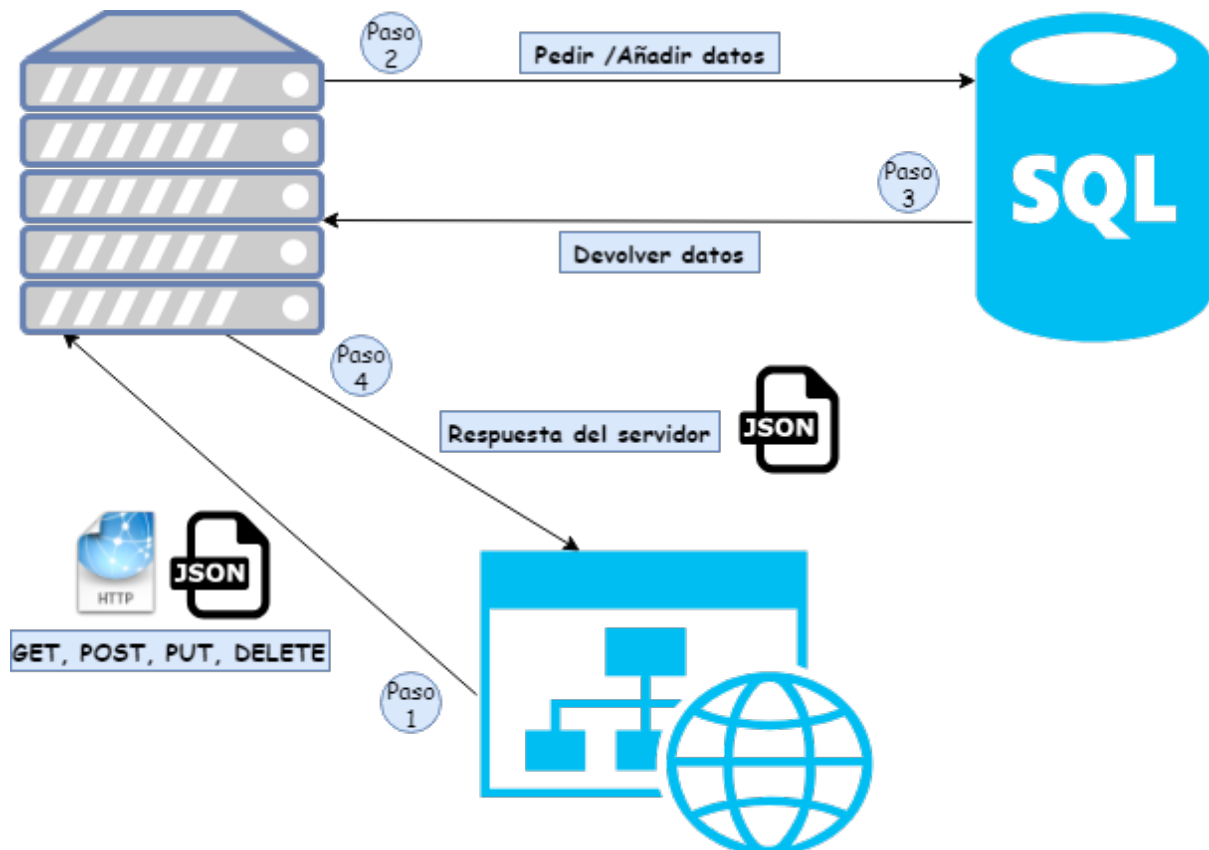


Figura 4.2: Arquitectura del sistema

El backend del sistema es una API REST: una interfaz entre sistemas que usa HTTP para obtener datos o generar operaciones sobre estos datos en cualquier formato, entre ellos XML y JSON. Las características principales de REST son las siguientes:

- Protocolo cliente/servidor sin estado: cada una de las peticiones HTTP contiene la información adecuada para poder ejecutarla.
- Las operaciones más importantes son las mencionadas a continuación:
 - POST: crear un recurso.
 - GET: consultar un recurso.
 - PUT: actualizar un recurso.
 - DELETE: eliminar un recurso.
- Los objetos siempre se manipulan a partir de la URI (identificador de recursos uniforme).

Yo he sido el encargado de conectar el frontend de la página web con el backend. Aunque el uso de la API será explicado con más detalle en la sección 5.1.1, voy a destacar ahora los pasos generales que se producen cada vez que el frontend realiza una petición a la API (figura 4.2):

1. El frontend realiza una petición HTTP a la API: POST, GET, PUT o DELETE.
2. La API ejecuta la acción necesaria sobre la base de datos, ya sea alterando su estado u obteniendo datos.
3. La base de datos puede devolver un error al ejecutar la acción o también devolver datos si la petición realizada ha sido de tipo GET.
4. La API devuelve una respuesta que debe ser interpretada por el frontend.

4.2.2. Migración de la base de datos

Debido a la desviación en la planificación que he tenido respecto a lo establecido al principio de la estancia, he llegado a trabajar sobre la base de datos. Es por ello que incluyo esta sección, ya que considero que he tenido cierto impacto sobre el diseño de la base de datos, gracias al análisis y estudio que estuve realizando durante varias semanas.

Hubo un momento durante el desarrollo de mi proyecto, en el cual había finalizado todos los requisitos del cliente que se tenían hasta la fecha y tuve que hacer un desvío en la planificación. Fue aquí donde mi supervisor me comentó que el sistema del antiguo proyecto de Constructor del cual se partía, usaba una base de datos MySQL. Por una serie de motivos, en el nuevo sistema se iba a utilizar SQL Server, por lo que había que realizar una migración de la base de datos. Estuve bastantes días investigando a fondo la base de datos en MySQL a través de phpMyAdmin, con tal de ayudar a mi supervisor a diseñar la nueva base de datos.

Para realizar la migración, Jose Antonio ha desarrollado una aplicación en .NET Core, utilizando Entity Framework para la capa de acceso a la base de datos. No voy a entrar en más detalle respecto a la migración, ya que no forma parte de mi proyecto.

4.3. Diseño de la interfaz

La mayor parte de las interfaces del frontend han sido diseñadas por Pieter, pero yo he colaborado en algunas e incluso he diseñado otras desde cero. Voy a incluir las interfaces más relevantes, explicando siempre para qué sirve cada una y mi impacto en el diseño de la misma.

Lo primero que nos encontramos al acceder a la página web, es la pantalla de login. Yo me he encargado de mostrar los errores pertinentes que pueda ocasionar el inicio de sesión. La figura 4.3 es un ejemplo de un inicio de sesión no válido, debido a que las credenciales son incorrectas. El mensaje en holandés "De inloggegevens zijn niet correct" significa "Las credenciales de inicio de sesión no son válidas".

Inloggen

E-mailadres

Wachtwoord

De inloggegevens zijn niet correct

Inloggen

Figura 4.3: Interfaz de inicio de sesión no válido

Una vez iniciada la sesión, si el usuario es un miembro, puede realizar acciones como indicar que ha asistido a un curso. En la figura 4.4 se muestra la interfaz creada para añadir esta información. Yo me he encargado de diseñar los formularios de búsqueda: el de arriba a la izquierda busca por referencia y el de arriba a la derecha busca por institución.

EEN INCOMPANY CURSUS TOEVOEGEN

Zoek op referentie nummer Regulier zoeken

Referentienummer **controleer nr**

Instituut *Instituut naam a.d.h. van het nr*

Naam *naam a.d.h. van het nr*

Instituut

Naam

Type cursus

Niveau

Afgerond in

Aantal lesuren (excl. pauzes e.d.)

Afgerond met toets

Toevoegen

* Na controle van de invoer wordt het aantal punten definitief vastgesteld. Dit kan verschillen met het puntenaantal dat automatisch gegenereerd wordt.

Staat jouw cursus er niet bij? voeg er een toe!

Figura 4.4: Interfaz para añadir asistencia a un curso

En todas las páginas en las que se podían editar o eliminar elementos (cursos, experiencia laboral, empresas, etc.), he añadido unos iconos de editar y borrar como los que se aprecian en la figura 4.5. Las interfaces ya estaban diseñadas cuando empecé el proyecto pero no tenían la opción de editar y eliminar. Pieter fue el que me asignó la tarea de modificar las interfaces en las que hubiera que realizar estas acciones.

MEDEDELING OVERZICHT

Mededeling toevoegen





titel	datum	
Toetsing 9 April	26-02-2018 11:00	 
Toetsing 2017/2018	23-02-2018 09:54	 

Figura 4.5: Interfaz para eliminar o modificar un anuncio

Estas serían algunas de las interfaces de la aplicación web. He incluido solamente capturas de interfaces en las que he colaborado en cuanto a diseño se refiere. Hay que tener en cuenta que muchas interfaces de crear, listar o editar son prácticamente idénticas, razón por la cual no las he incluido.

Capítulo 5

Implementación y pruebas

En este capítulo se van a explicar los detalles de implementación del proyecto, incluyendo las pruebas de verificación y validación que se han realizado durante el desarrollo. Debido a que algunos temas pueden ser bastante extensos, la explicación detallada se encuentra en el Anexo A.

5.1. Detalles de implementación

Esta sección sirve para conocer en profundidad cómo se ha realizado la implementación del proyecto. Se han usado técnicas y filosofías de React apropiadas, explicadas en las siguientes secciones. Los porcentajes de esfuerzo en el proyecto son los siguientes, incluyendo la preparación previa antes del comienzo del desarrollo. Están calculados sobre el tiempo total de desarrollo (no se incluyen el resto de tareas).

1. Formación ReactJS: 22 % (50 horas).
2. Programación en la parte frontend: 61 % (143 horas).
3. Diseño de las interfaces: 4 % (10 horas).
4. Estudio de la base de datos de la antigua web: 13 % (30 horas).

5.1.1. Uso de la API REST

Todo el funcionamiento del proyecto Constructor se basa en la conexión de la parte frontend con la parte backend. Gran parte del trabajo realizado en mi proyecto, ha sido el de realizar peticiones a esta API REST, con el fin de obtener una respuesta y mostrar los datos en el frontend. Una de las herramientas que más he utilizado durante todo el proyecto y que me ha facilitado entender la API e interpretar sus respuestas, ha sido Postman (sección 2.1.4). En Postman, tenía las credenciales de la cuenta de prácticas de E-Force, donde disponíamos de un

workspace para el proyecto Constructor. Este workspace tenía 3 colecciones (Admin, Member y Public) con todas las peticiones que se podían realizar a la API. En la siguiente figura se pueden ver estas colecciones, así como una lista desplegada de las peticiones disponibles para el área pública.

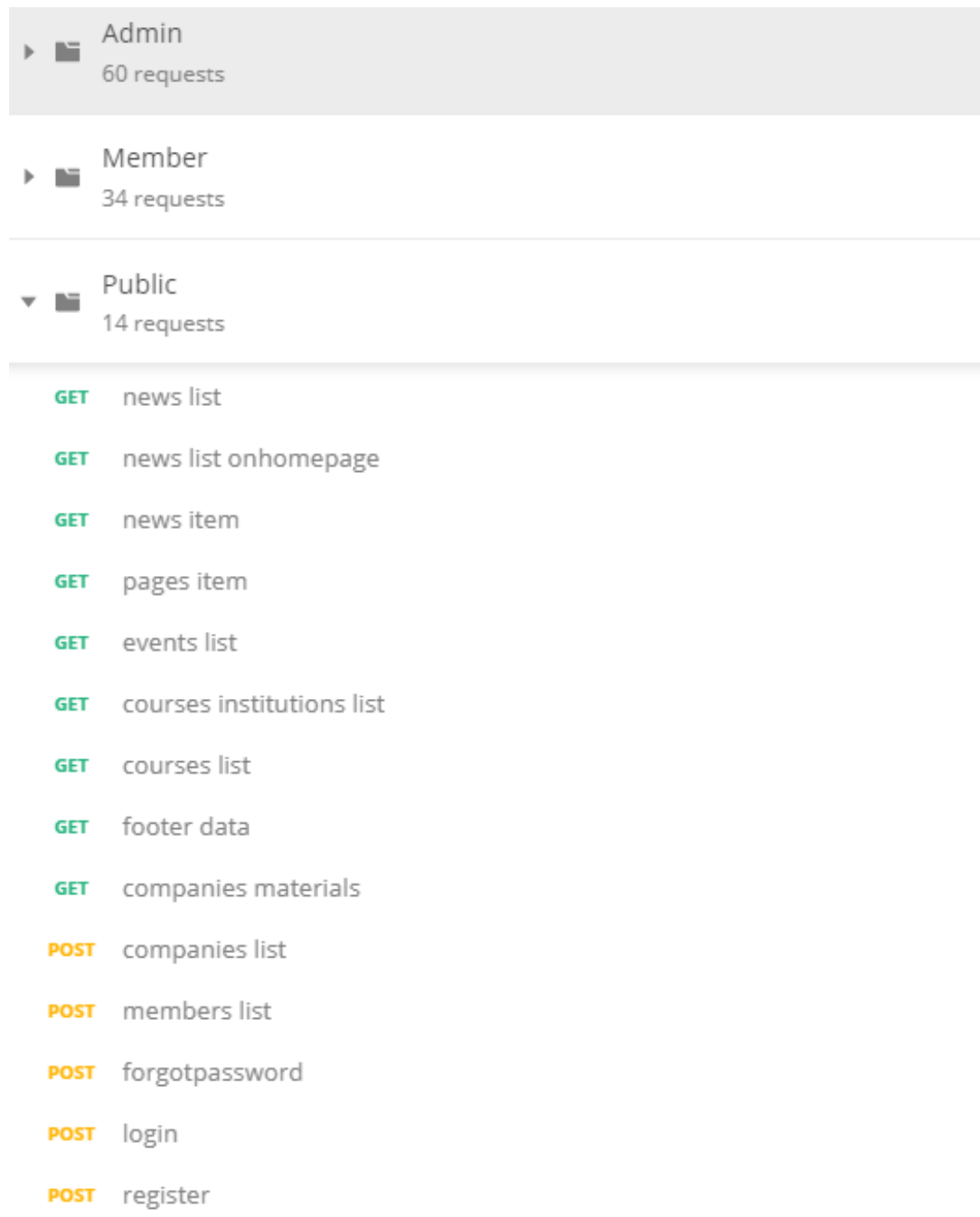


Figura 5.1: Colecciones de Constructor en Postman

Para poder trabajar de una manera más cómoda, también añadimos un entorno en Postman, con variables predefinidas. Esto era necesario para no tener que repetir la url y el token en cada petición. Los pasos a seguir para realizar peticiones desde Postman son los siguientes:

1. Realizar una petición a `{{url}}/api/auth/login` con las credenciales correctas (Figura A.2 del Anexo A).
2. Añadir el token recibido en la respuesta de la petición, en las variables del entorno en el que estamos trabajando (Figura A.1 del Anexo A).
3. Realizar una petición añadiendo la variable `{{token}}` en el header (Figuras A.3 y A.4 del Anexo A).

Gracias a Postman, he podido testear todas las peticiones sin necesidad de hacerlo manualmente a través de la interfaz de la página web. De esta manera, también he observado qué datos devolvía la API en los diferentes tipos de peticiones, así como los distintos códigos de respuesta. La manera de realizar una petición desde código en ReactJS era la siguiente:

1. Lo primero que hice fue crear un método general en un componente de alto nivel como es **Api.js**. Este método era el encargado de realizar la petición a la API y gestionar la respuesta recibida (en la sección A.2 del Anexo A se explica como se realiza una petición en ReactJS). Gracias a esto, evitaba repetir código en todos los componentes que necesitaban hacer llamadas a la API, simplemente importaba el componente **Api.js** y utilizaba el método.
2. Este método devolvía una respuesta en JSON que era interpretada por el componente que había realizado la llamada al método.
3. El componente mostraba la información recibida, actualizaba la página o realizaba la operación pertinente en base a la respuesta recibida.

5.1.2. Control de usuarios

Existen diferentes tipos o roles de usuarios dentro de la página web. Dependiendo del rol que tenga el usuario, tendrá acceso a unos recursos de la página web u otros. En la figura A.2 se pueden ver los diferentes tipos de roles disponibles, en la respuesta de la petición de login. Cuando el inicio de sesión es satisfactorio, en esta respuesta nos llega información relevante acerca del rol del usuario. He implementado una serie de funciones en el componente **Api.js** que almacenan en localStorage esta información, para que pueda ser consultada a lo largo de la navegación. La propiedad localStorage permite almacenar datos que no tienen fecha de expiración. Las funciones más importantes que he implementado y que usan localStorage, son las siguientes:

- `setIsAdmin()`: almacena una variable que indica si el usuario es administrador o no.
- `setRoles(roles)`: recibe un JSON indicando que roles tiene el usuario y los almacena en diversas variables.

- `isAdmin()`: indica si el usuario tiene el rol de administrador o no.
- `hasMemberRole()`: indica si el usuario tiene el rol de miembro o no.

En los componentes que muestran información que solo pueden ver miembros con ciertos roles, se utilizan estas funciones para diferenciarlos y ocultar información. La siguiente imagen muestra la información que se ha almacenado en la variable `localStorage` del navegador Chrome, tras un inicio de sesión de un miembro:

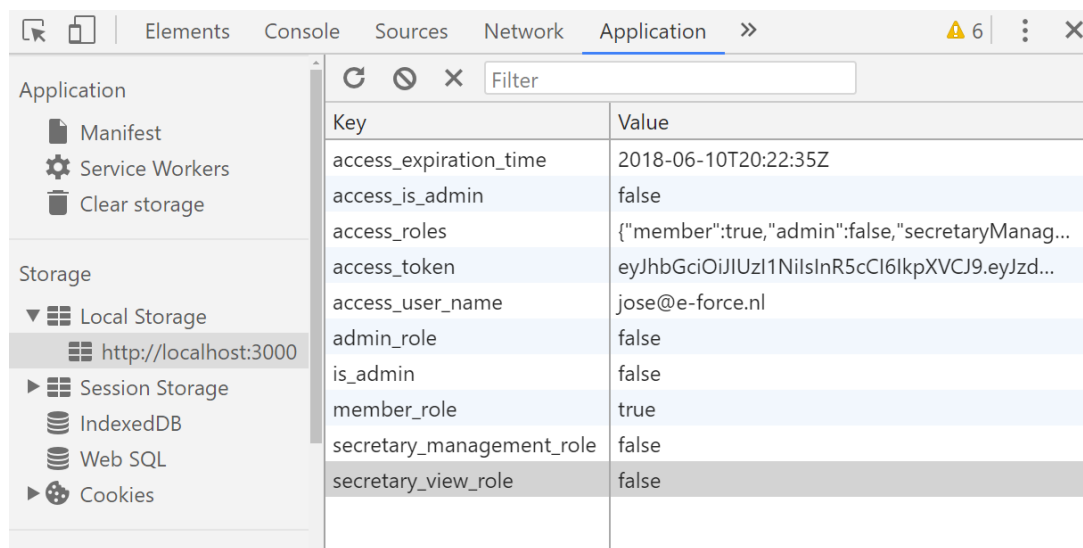


Figura 5.2: LocalStorage del navegador Chrome

5.1.3. Ciclo de vida de un componente

En React, los componentes tienen un ciclo de vida que se repite cada vez que se crea (en la sección A.3 del Anexo A se explica el ciclo de vida de componente en ReactJS). Todos los componentes que he implementado en el proyecto, siguen unas pautas comunes durante su ciclo de vida:

- `constructor`: es el método que se ejecuta antes de montar un componente. Es aquí donde inicializo el estado del componente.
- `componentDidMount()`: es el método que se ejecuta inmediatamente después de que el componente se haya montado. Es el mejor lugar para cargar información desde un punto remoto, por lo que lo he utilizado para realizar las peticiones a la API relacionadas con listar una serie de elementos (que deben mostrarse al cargar la página).
- `render`: es el método que renderiza un componente, mostrándolo a través de la interfaz. Es aquí donde examino los valores y propiedades del componente, para saber qué mostrar por pantalla.

5.1.4. Jerarquía de componentes

Durante el desarrollo del proyecto, había muchas ocasiones en las que necesitaba que un componente se comunicara con un hijo o viceversa. También necesitaba que un componente padre llamara a funciones de uno de sus componentes hijos para actualizar otro de sus componentes hijos. He implementado las siguientes funciones, que me han servido para realizar las acciones necesarias en todos los componentes del sistema:

Componente hijo que requiere información de su componente padre

El componente hijo recibe la información necesaria a través de las propiedades. Estas propiedades son pasadas desde el componente padre al componente hijo en el momento de crear la instancia. El componente hijo utilizaba estas propiedades llamando a `this.props.nombreDeLaPropiedad` o las almacenaba en el estado y accedía a ellas a través de `this.state.nombreDeLaPropiedad`.

Componente padre que requiere información de su componente hijo

El componente padre almacenaba una referencia al componente hijo a la hora de instanciarlo, tal que así: `ref={instance => { this.child = instance }}`. Posteriormente, puede llamar a funciones que se encuentran en el componente hijo de la siguiente manera: `this.child.nombreDeLaFuncion`.

5.1.5. Programación funcional

He utilizado operadores funcionales para la mayoría de funciones del proyecto. Al contrario que la programación imperativa, estos operadores se basan en la programación funcional. Dos ejemplos muy útiles y que he usado con frecuencia han sido `filter` y `map`. El más interesante y que más me ha servido ha sido el operador `map`. A continuación se puede observar un fragmento de código donde lo utilizo:

```
1 let events = objectData.events.map(function (item, index) {
2   return <tr key={item.id}>
3     <td>{item.dateCreated.slice(0, 10)}</td>
4     <td>{item.organization}</td>
5     <td>{item.name}</td>
6     <td>{item.points}</td>
7     <td><a href={item.website}>website</a></td>
8   </tr>
9 })
```

En este caso, el operador `map` se utiliza para realizar una acción sobre una lista de eventos. Cada evento tiene unos atributos y se pueden acceder dentro del operador `map`. Por cada evento se genera una fila en html con los datos de ese evento.

También he utilizado las arrow functions, una nueva forma de expresar las funciones, de forma más breve y con algunas características nuevas. Un ejemplo de uso en mi proyecto podría ser el siguiente:

```
1 getInitialState = () => {
2     return {
3         data: {
4             id: "",
5             title: "",
6             content: RichTextEditor.createEmptyValue(),
7         },
8         errors: {}
9     }
10 }
```

5.2. Verificación y validación

Debido al poco tiempo disponible en la estancia de prácticas, las pruebas de testing que se han realizado han sido mínimas. También es cierto que como el sistema aún no está disponible para ser lanzado a producción, no se han realizado todas las pruebas pertinentes. Durante el desarrollo del proyecto, he ido realizando pruebas alpha, que son pruebas del software realizadas cuando el sistema está en desarrollo, asegurando que lo se está desarrollando es correcto y útil para el cliente. Las pruebas las he realizado sobre prototipos que habían sido desarrollados rápidamente y con poco coste. Si las pruebas alpha fallan, no se desperdicia demasiado tiempo ni dinero. Me encargaba de comprobar que todos mis componentes funcionaran a la perfección, realizando pruebas sobre ellos en diferentes escenarios como los siguientes:

- Realizar peticiones a la API con el servidor apagado.
- Realizar peticiones con datos no válidos.
- Realizar peticiones con datos válidos.

Capítulo 6

Conclusiones

6.1. **Ámbito formativo**

Durante estos meses que he estado realizando las prácticas, he aprendido muchas cosas. Partía del desconocimiento total de la biblioteca ReactJS, así como del paradigma web y he acabado adquiriendo gran cantidad de conocimientos. Gracias en gran parte a los compañeros, he aprendido técnicas nuevas de desarrollo web, así como herramientas que facilitan el trabajo. Quiero destacar el uso de algunas herramientas como Postman o Sourcetree, ya que nunca había trabajado con ellas y ahora las considero una parte importante en cualquier proyecto.

6.2. **Ámbito profesional**

La experiencia en la empresa ha sido muy buena. La forma de trabajar, la buena actitud y predisposición y las ganas de aprender, han sido factores clave para sentirme cómodo durante la estancia. Enseguida he conectado con los compañeros y siempre tenía el apoyo necesario para resolver cualquier problema que me pudiera surgir. Aunque he estado prácticamente desarrollando solo mi proyecto todo este tiempo, siempre me he sentido arropado.

6.3. **Ámbito personal**

Considero que mi estancia en E-Force ha sido una experiencia personal positiva. Necesitaba trabajar sobre un proyecto serio para demostrar los conocimientos adquiridos durante el grado, y creo que aquí he podido hacerlo. El tener una rutina todos los días y estar en un entorno empresarial, ha despertado mis ganas de salir al mundo laboral en busca de una empresa donde crecer como persona y desarrollador.

Bibliografía

- [1] Documentación de microsoft. <https://docs.microsoft.com/es-es/dotnet/framework/get-started/net-core-and-open-source>. [Introducción a .NET Core].
- [2] Página web de bitbucket. <https://es.atlassian.com/software/bitbucket>. [Bitbucket homepage].
- [3] Página web de e-force ict. <http://www.e-force.nl/>. [E-Force ICT home page].
- [4] Página web de indeed. <https://www.indeed.com/salaries>. [Salarios].
- [5] Página web de reactjs. <https://reactjs.org/>. [ReactJS home page].
- [6] Página web de sourcetree. <https://es.atlassian.com/software/sourcetree>. [Sourcetree home page].
- [7] ¿qué es scrum? <https://proyectosagiles.org/que-es-scrum/>. [Qué es SCRUM].

Anexo A

Peticiones a la API

A.1. Imágenes detalladas

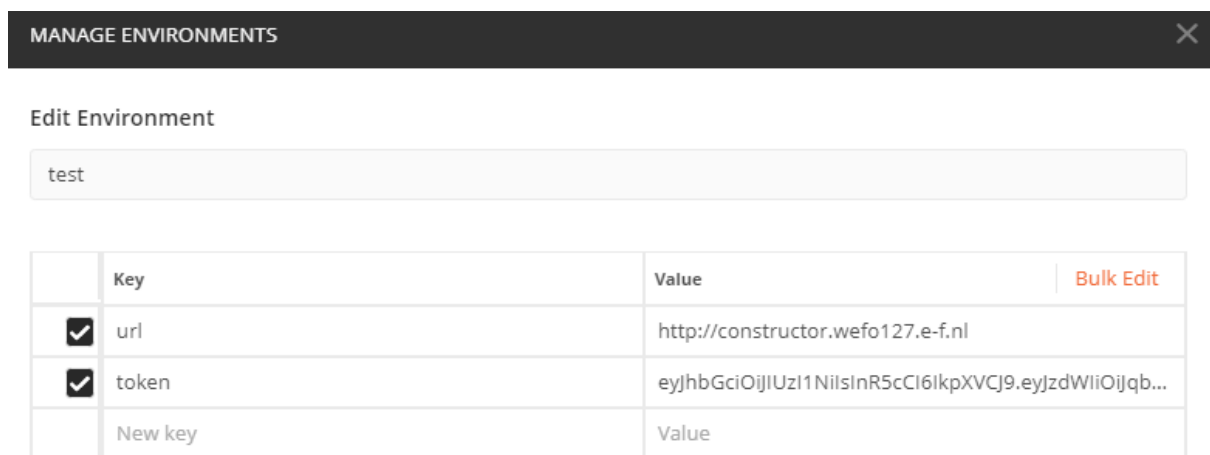


Figura A.1: Entorno test en Postman

A.2. Función fetch en ReactJS

La función **fetch** proporciona una interfaz para recuperar recursos (incluyendo recursos remotos a través de redes). Le resultará familiar a cualquiera que haya usado XMLHttpRequest, pero esta nueva función ofrece un conjunto de características más potente y flexible.

Fetch ofrece una definición genérica de los objetos Request y Response (y otras cosas relacionados con las solicitudes de red). Esto permitirá que sean utilizados donde sea necesario en el futuro. También proporciona una definición de conceptos relacionados, como CORS y la semántica de encabezado HTTP origen, suplantando sus definiciones separadas en otros lugares.

La función `fetch()` toma un argumento obligatorio, la ruta de acceso al recurso que desea recuperar. Devuelve una Promise que resuelve en Response a esa petición, sea o no correcta. También puede pasar opcionalmente un objeto de opciones como segundo argumento. Una vez que Response es recuperada, hay varios métodos disponibles para definir cuál es el contenido del cuerpo y como se debe manejar.

A.3. Ciclo de vida de un componente en ReactJS

Cuando una instancia de un componente se está creando y se inserta en el DOM, se ejecutan los siguiente métodos:

- `constructor()`
- `static getDerivedStateFromProps()`
- `componentWillMount()`
- `render()`
- `componentDidMount()`

Cuando se actualizan las propiedades o el estado de un componente, se produce una actualización. En este momento, se produce una re-renderización del componente y se ejecutan los siguientes métodos:

- `componentWillReceiveProps()`
- `static getDerivedStateFromProps()`
- `shouldComponentUpdate()`
- `componentWillUpdate()`
- `render()`
- `getSnapshotBeforeUpdate()`

- `componentDidUpdate()`

Cuando un componente se elimina del DOM, se ejecuta el método `componentWillUnmount()`.

references list [DONE] Examples (0) ▾

GET ▾ `{{url}}/api/references/` Params Save ▾

Authorization Headers (2) Body Pre-request Script Tests Cookies Code

Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> Authorization	Bearer {{token}}	
New key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 337 ms Size: 816 B

Pretty Raw Preview JSON ▾ Save Response

```

1 {
2   "references": [
3     {
4       "abbreviation": "3",
5       "firstName": "Pablo00000",
6       "middleName": "Berbel",
7       "lastName": "Marin",
8       "company": "E-force",
9       "function": "Becario",
10      "phone": "1234567",
11      "email": "pabloberbel.1993@gmail.com",
12      "id": 2737,
13      "rowVersion": "AAAAAAAAABfXE="
14    },
15    {
16      "abbreviation": "4",
17      "firstName": "Pedro",
18      "middleName": "Martinez",
19      "lastName": "Martinez",
20      "company": "E-force",
21      "function": "Programador",
22      "phone": "987654321",
23      "email": "pabloberbel.1993@gmail.com",
24      "id": 2740,
25      "rowVersion": "AAAAAAAAABbXs="
26    }
27  ]
28 }

```

Figura A.3: Petición para obtener las referencias

▶ references edit and new [DONE] Examples (0) ▾

POST ▾ `{{url}}/api/references/` Params Send ▾ Save ▾

Authorization Headers (2) Body Pre-request Script Tests JSON (application/json) ▾ Cookies Code

form-data x-www-form-urlencoded raw binary

```

1 {
2   "abbreviation": "1",
3   "firstName": "test",
4   "middleName": "test",
5   "lastName": "test",
6   "company": "test",
7   "function": "test",
8   "phone": "123123123",
9   "email": "pieter@force.nl",
10  "id": 906
11 }

```

Status: 200 OK Time: 180 ms Size: 369 B

Body Cookies Headers (5) Test Results Save Response

Pretty Raw Preview JSON ▾

```

1 {
2   "abbreviation": "1",
3   "firstName": "test",
4   "middleName": "test",
5   "lastName": "test",
6   "company": "test",
7   "function": "test",
8   "phone": "123123123",
9   "email": "pieter@force.nl",
10  "id": 2741,
11  "rowVersion": "AAAAAAAAAB3JU="
12 }

```

Figura A.4: Petición para crear una referencia