Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Tarea 3

Prototipo de Implementación

Integrantes: Pedro Belmonte
Ricardo Cordova
Israel Peña
Joaquín Pérez
Profesora: Jocelyn Simmonds
Auxiliar: Constanza Escobar
Ayudantes: Sergio Leiva
Pablo Miranda
Santiago, Chile

1. Introducción

En la tercera iteración del trabajo realizado, se deben desarrollar las interfaces diseñadas para el sistema, por el grupo seleccionado en la tarea anterior.
Las interfaces solicitadas corresponden al landing page para personas naturales, el landing page para administradores, el perfil del usuario (visto por el dueño del perfil) y la ficha de un artículo. Para cada uno de ellos, se utiliza el diseño realizado por Soft-of-War (según lo votado por el curso). En este informe, se presenta las metodologías utilizadas por el equipo de Soft-Where para realizar estas implementaciones, el modelo de datos utilizado y su relación con las interfaces gráficas implementadas; finalmente, se presenta como las interfaces implementadas satisfacen los requisitos funcionales establecidos para esta entrega. Se concluye este informe con reflexiones sobre el trabajo y la metodología utilizada.

2. Metodologías de Equipo

El equipo de trabajo utilizó una metodología de trabajo tipo Scrum: esto es, con trabajo diario de avance en "features", divididos para los distintos miembros del equipo, necesarios para la implementación de los requisitos, con revisión constante de los logros conseguidos.
Sin embargo, hay ciertas limitantes que no permiten realizar un trabajo Scrum a cabalidad: no es posible hacer reuniones diarias dada la distancia entre los miembros, así como los horarios de la universidad, además de que no se dispone de los espacios necesarios como para poder llevar una tabla de avances como lo requiere esta metodología de trabajo. Herramientas como Git y Git Flow ayudan a suplir algunas de estas falencias, ya que permiten ir revisando los avances que realizan los otros miembros del equipo y los "features"sobre los que trabajan.
El desarrollo de las implementaciones se realiza utilizando el Framework de Django, que permite levantar el funcionamiento de un servicio web rápidamente con backend basado en Python y frontend basado en html y sintaxis de DTL que separa el funcionamiento del diseño, permitiendo que cada parte pueda trabajar de forma independiente sin tener que repetir información de uno en otro. Otros elementos que se añaden para mejorar el trabajo desarrollado son Bootstrap, para tener un diseño de web más responsivo, y Pillow, para facilitar el manejo de imágenes en la base de datos.
Dada esta facilidad de separar las partes, y las restricciones a las que nos enfrentamos, se decide que los miembros del equipo deberían trabajar en las áreas que consideren son las más fuertes para ellos mismos. De esta forma, se aprovechan las habilidades y los intereses de cada uno de los miembros del equipo.
Por ejemplo, Ricardo trabajó principalmente en Backend, en lo referido al modelo de datos y el sistema de login y creación de usuarios; mientras que Pedro trabajó principalmente en el Frontend, pues conocía el funcionamiento de Bootstrap y tiene buenas nociones de diseño. De está forma, cada uno de los miembros del equipo pudo aportar lo mejor de cada uno.

Miembro Equipo	Participación
Pedro Belmonte	• Principalmente Front-End • Grilla de Espacios • Landing Page User
Ricardo Cordova	• Principalmente Back-End • Sistema de Registros/Login • Permisos de Adminis
Israel Peña	• Pedidos de Artículos • Ficha de Artículo • Admin Landing Page
Joaquín Pérez	• Perfil de Usuario • Diferenciación entre user para el Landing Page • Realizar Merging

Gracias a esta metodología de trabajo, se pudieron implementar los requerimientos indicados por lo solicitado en las instrucciones de la tarea n° 3: los requisitos 1-2, 4-5, 7-8, 12, 14-18, 23, 25-26, 28, 37-40, 51 y 54-56. En la sección 4 de este informe se especifíca como se satisfacen estos requisitos mediante las interfaces implementadas. En la sección 3 se observara el modelo de datos que subyace a las interfaces implementadas, y como estas se relacionan con los modelos para poder implementar sin tener que re-escribir código.

3. Modelo de Datos

El modelo de datos es el siguiente:

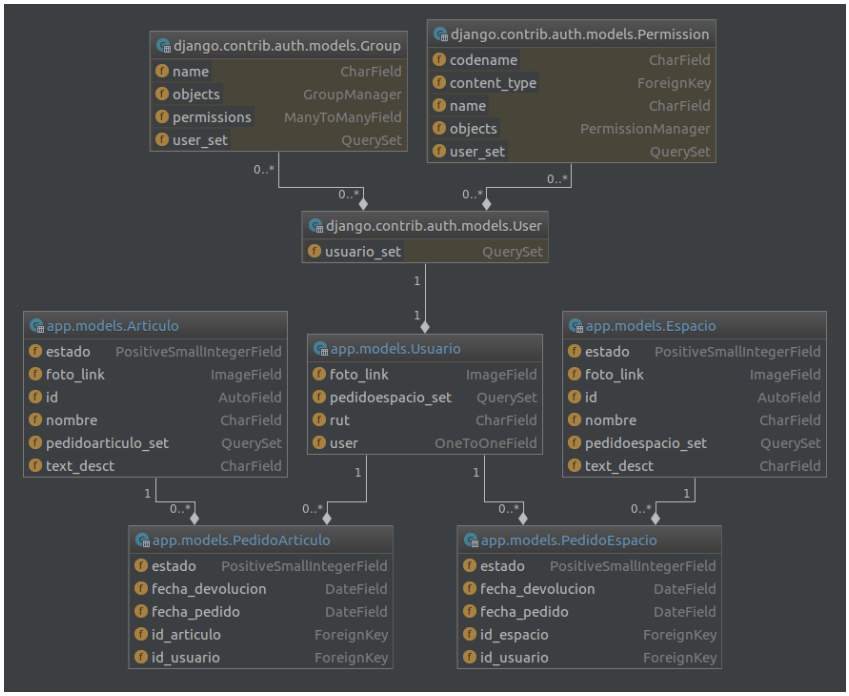

Figura 1: Diagrama del modelo de datos implementado

En este modelo se pueden distinguir tres niveles: un primer nivel generado automaticamente por Django en cuanto se utilizan sus modelos de Groups, Permission y User; un segundo nivel creado por el equipo donde se encuentran los elementos directamente relacionados con los requisitos que son los Artículos, Usuarios y Espacios; el último nivel es creado también por el equipo, pero son tablas relacionales entre los usuarios y artículos y espacios que representan los pedidos realizados por los usuarios y los resultados históricos de estas solicitudes. Los modelos del primer nivel permiten generar un sistema de creación de usuario y logín con facilidad. Estos modelos disponen de nombre, apellido, correo, clave, permisos y grupos, todo ello por defecto dentro de Django. Así, podemos diferenciar una Persona Natural de un Administrador utilizando grupos, dando permisos por separado a cada grupo según lo indicado en los requisitos, así como poder guardar la información requerida por los usuarios.
Sin embargo, se hace necesario complementar esta información con datos adicionales que no corresponden al modelo de Django. Para ello, en el segundo nivel del modelo se extiende el usuario por defecto de Django con un usuario personalizado con nueva información adicional, como el rut y la foto de perfil; esta información servira para poblar de contenido el perfil del usuario. Además, consideramos tablas adicionales para los artículos y los espacios que dispondra el sistema para prestar a sus usuarios. La información que ahí se almacena servira para poder poblar las fichas de los artículos y espacios.
El último nivel, que posee las tablas relacionales entre usuario y artículo, y usuario y espacio; estas tablas almacenan la información de los pedidos hechos por un usuario, la fecha de pedido y

devolución. Esta información se puede usar por el perfil del usuario como un registro histórico de sus pedidos, además de poder indicar al sistema que elementos se encuentran disponibles para que otros usuarios puedan pedir o no un artículo (según las fechas de los pedidos), así como informar a los administradores sobre la vigencia o caducidad de una solicitud (entre otros posibles).

4. Cumplimiento Requisitos

Los requisitos se dividieron en grupos a modo de facilitar el trabajo sobre cada uno de ellos, de forma tal que la funcionalidad base pudiera dar cuenta de un conjunto de ellos en lugar de tan solo uno. Por ejemplo, si el sistema requiere de la inscripción de un usuario, era necesario que el usuario pudiera crear su cuenta, pudiera inscribirse y pudiera salir de su cuenta.
De esta forma, los requisitos se dividieron en: Requisitos generales (1-2, 4) Sistema de reservas y prestamos (5, 7-8, 12, 14-15), landing page para personas naturales (16-18), perfil de usuario (23, 25-26, 28), ficha de artículo (37-40), landing page para administradores (51, 54-56).
Del primer grupo de requisitos, podemos ver como estos se cumplen en las siguientes imágenes:

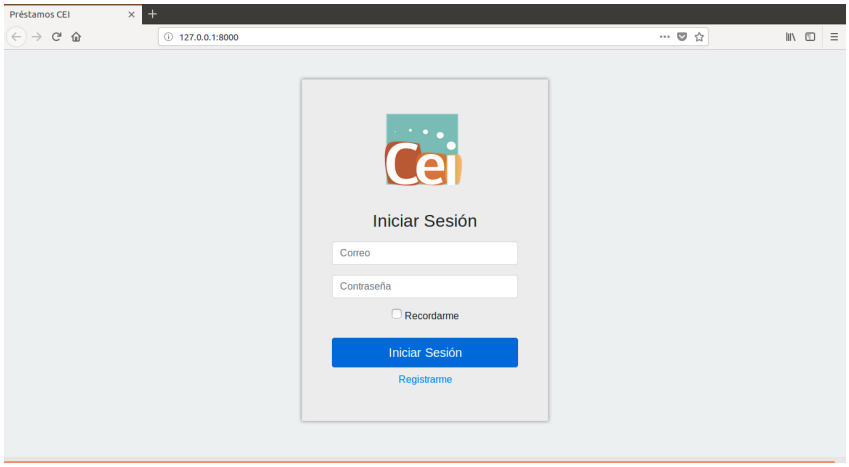

Figura 2: Página inicial de recepción a usuarios sin cuenta. Pide iniciar sesión (req 2) o crear cuenta (req 1) para poder acceder a los servicios del sistema.

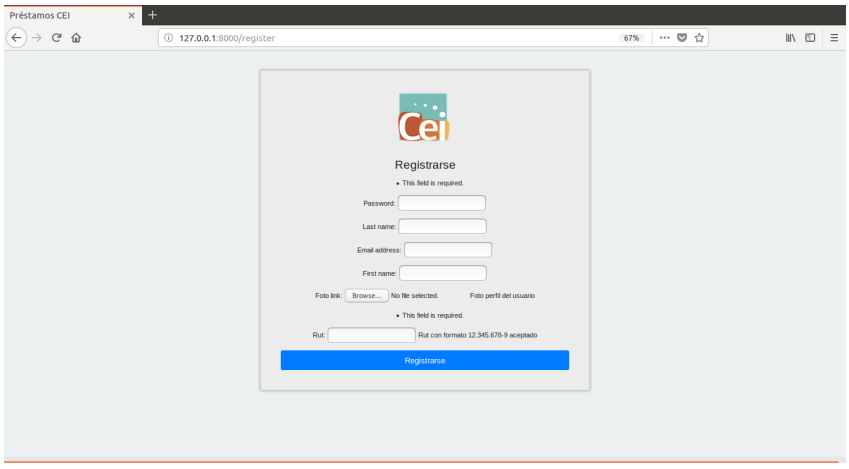

Figura 3: Página de creación de usuario (req 1). Todo usuario creado de esta forma es una persona natural, y solo los administradores pueden cambiar una cuenta a cuenta de administración (req 4).

Del sistema de reservas y prestamos, podemos apreciar como se resuelven los requisitos establecidos en la siguiente imagen de la interfaz que se implementa:

Figura 4: Se aprecia como funciona el sistema de solicitud de prestamos. Se puede observar que solo se pueden solicitar durante los días habiles (req 14), pero otros detalles como las restricciones de tiempo (req 12) e identificadores únicos (req 15) no se puede apreciar por su propia naturaleza interna.

De los requisitos asociados al landing page para personas naturales, podemos ver como estos se cumplen en las siguientes imágenes:

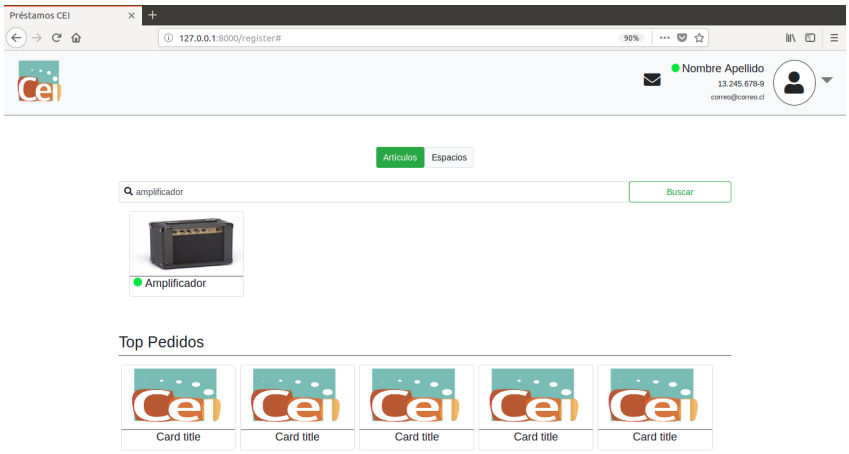

Figura 5: Página de inicio para persona natural recién conectada. Se puede apreciar como el usuario puede buscar artículos (req 16) y la barra de navegación a su perfil e historial de pedidos (req 23). Adicionalmente, se puede apreciar como la busqueda se puede restringuir según los criterios seleccionados por el usuario (req 17).

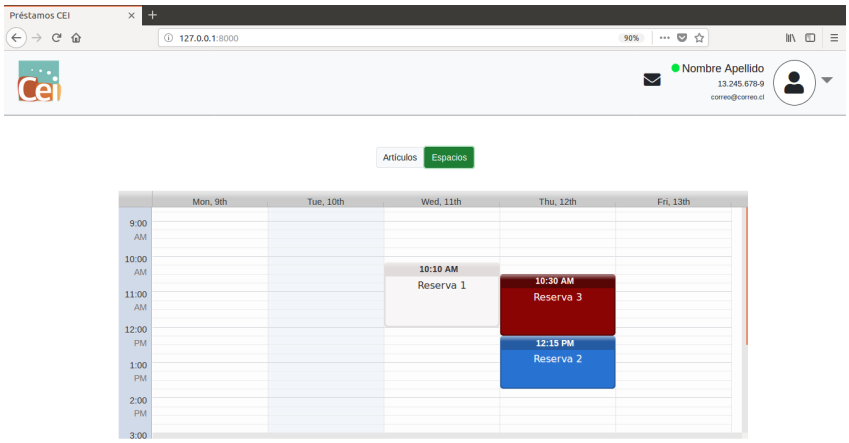

Figura 6: Página de inicio para persona natural, en pestaña de busqueda de espacios. Se aprecia la "grillaçon los horarios de espacios reservados (req 18).

Respecto del perfil de usuario, los requisitos que aquí se cumplen son los siguientes:

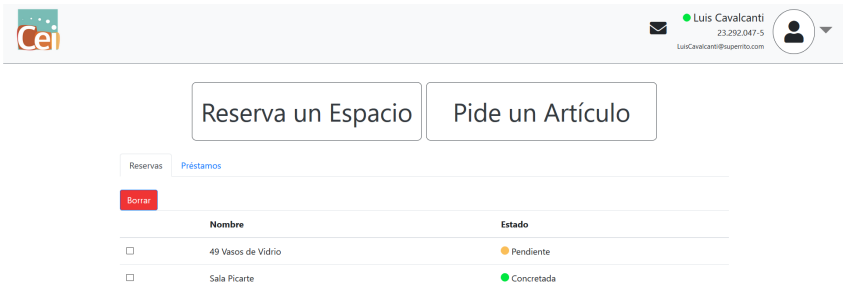

Figura 7: Página de perfil del usuario (visto por su dueño). En esta imagen se puede apreciar los elementos propios del usuario (req 23), así como la disponibilidad de su cuenta para crear y concretar prestamos y reservas (req 25). Adicionalmente, se presenta un historial con las últimas 10 reservas realizadas por el usuario (req 7 y 26) y muestra la posibilidad de eliminarlas si estas aún no se concretan (req 28).

Por otra parte, de los requisitos asociados a la ficha del artículo, las siguientes imágenes muestran como se concretan y resuelven dichos requisitos:

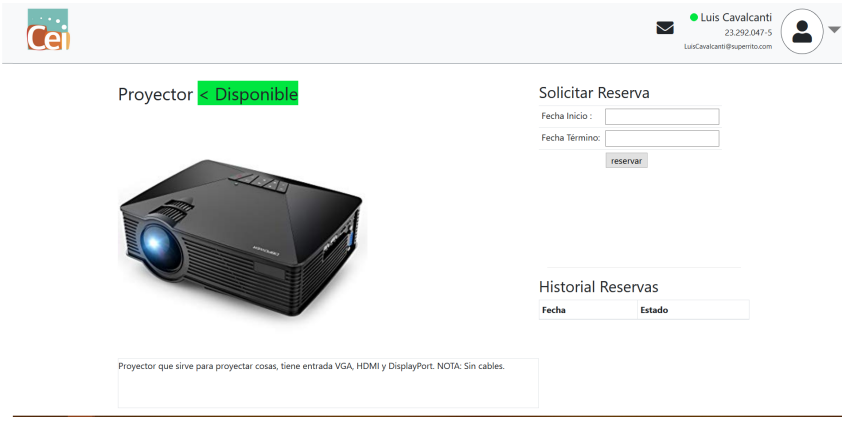

Figura 8: Página correspondiente a la ficha del artículo. Se puede apreciar como el artículo dispone de toda la información solicitada (req 37), junto con un resumen de los últimos pedidos hechos por el artículo (req 38). También se observa como el usuario, persona natural, puede hacer una solicitud de prestamo por este artículo (req 5 y 40).] [Ficha del artículo (vista admin)/ comentario: Página correspondiente a la ficha del artículo, visto por la cuenta de un administrador. Se ven los mismos elementos que en la vista por una persona natural, pero se añade la posibilidad de gestionar la información del artículo por el administrador (req 39).

Finalmente, los requisitos asociados directamente con la landing page de los administradores se pueden ver en las siguientes imágenes del sistema, vistos por una cuenta de administrador:

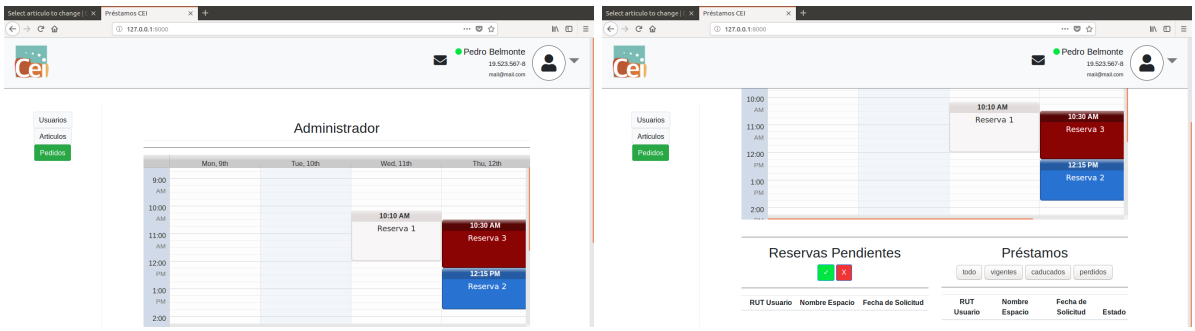

Figura 9: Página inicial de recepción para un administrador. Se aprecia el sistema de "grilla"para los horarios de reserva para los espacios administrados por el CEI (req 51). Se aprecian las reservas pendientes ordenadas por fecha (req 54), la posibilidad del administrador de marcas dichas reservas (req 55).

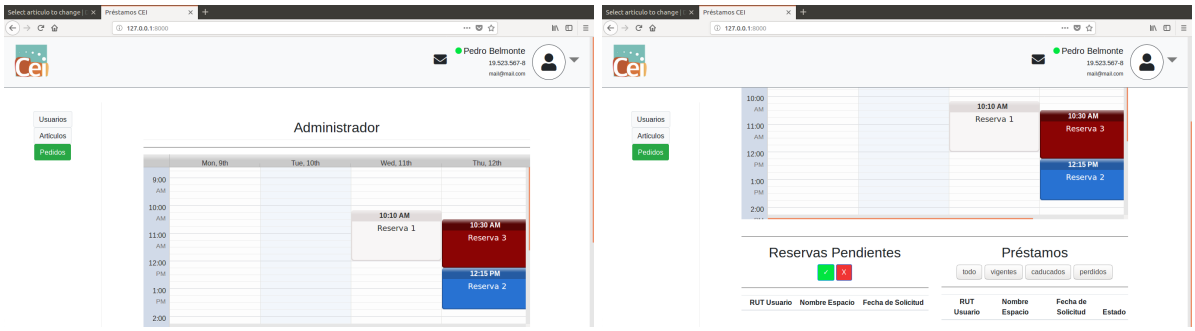

Figura 10: LP de admin con prestamos pendientes/comentarios: Página inicial de recepción para un administrador, con listado de préstamos pendientes. Se aprecia el orden de los préstamos pendientes y la posibilidad de filtrar por estado (req 8 y 56).

Cabe hacer notar que la funcionalidad desarrollada para una parte, se aprovecha también para otra parte del sistema, por lo que ciertos requisitos se cumplen más de una vez.

5. Conclusiones

Luego de haber realizado la implementación de las interfaces del sistema estudiado durante el curso, se pueden llegar a una serie de conclusiones, en base al trabajo realizado, las herramientas utilizadas y las metodologías aplicadas. En cuanto al trabajo realizado, podemos ver como la organización por requisitos ayuda a organizar los objetivos y la funcionalidad deseada para una aplicación, eliminando las distracciones y las ideas que pueden surgir a medida que se implementan otras funcionalidades. De esta forma, el trabajo se hace más organizado, más directo y más sencillo para todos los que participan. Por otro lado, muchas veces los requisitos implican otras funcionalidades, lo cual complica los límites y las responsabilidades de los trabajadores: por ejemplo, si una persona se encuentra encargada de desarrollar modelos, y otra se encuentra encargada de desarrollar un sistema de login, ambas partes se relacionan con los permisos de usuarios y la posibilidad de hacer préstamos. ¿Quién es el responsable de que el usuario pueda, efectivamente, hacer el pedido? Las herramientas utilizadas, en particular Git, ayudan a superar estos problemas y ha hacer que el trabajo sea más armonioso. Si una parte no está resuelta o si falta funcionalidad, los responsables de las partes individuales pueden compartir sus inquietudes y resolver aquello que impide que su código funcione. De la misma forma, Django ayuda a separar las responsabilidades, asegurando que gran parte del sistema pueda funcionar de forma autonoma a las otras (es decir, desarrollar un frontend independiente del desarrollo del backend, pero que asume que el otro estará funcionando para poder tener en .ªndando.el sistema). Finalmente, la metodología de trabajo tipo Scrum que se utilizó ayuda a acelerar los resultados deseados del trabajo. Poder separar el trabajo no solo en fronend y backend, si no que en "features", aseguraba el avance constante del quehacer de cada uno de los miembros del trabajo.