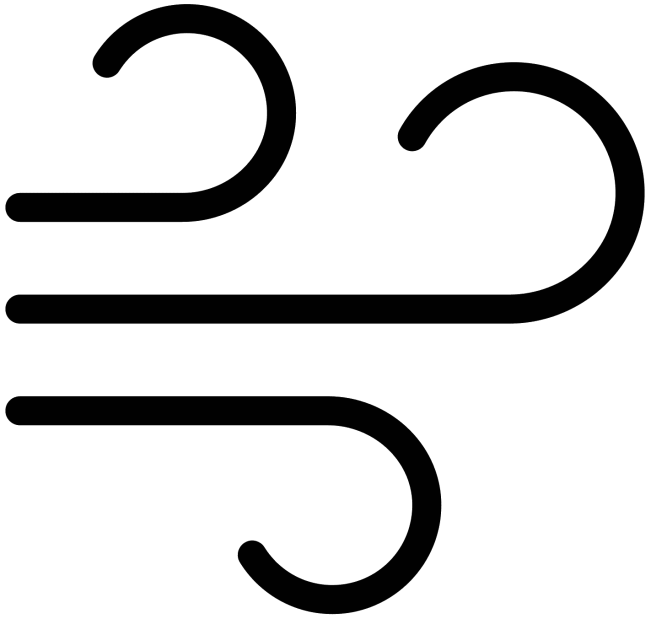
SISTEMAS Y TECNOLOGÍAS WEB

CIERZO



Virus Five:

Sergio Costa - 735317
Alejandro Gutiérrez - 735089
Daniel Revillo - 738460
Jorge Terreu - 735556
Juan Vallejo - 738435

Resumen del proyecto

Resumen

Va a tratarse de una aplicación para mostrar parámetros con una puntuación sobre conectividad, contaminación, nivel económico, nivel educativo de cada zona/barrio de Zaragoza. Además también va a elaborar un ranking de zonas ordenadas según los parámetros anteriores. Esto va a permitir consultar información sobre cada barrio de Zaragoza que los usuarios pueden emplear con diferentes propósitos.

Repositorio Github: https://github.com/sergiocostamoreno/VirusFive

Parámetros y Rankings

- Contaminación: de aire y acústica
- Conectividad: Se buscarán los siguientes datos: paradas de bus cercanas, paradas de *bizi zaragoza*, carril bici, cercanía al tranvía, plazas de aparcamiento. Existe también una base de datos con tiempos de recorrido entre los paneles informativos de la ciudad.
- Demografía: Media de edad de la gente por zonas, cantidad de gente jóven, niños, ancianos, etc, por cada zona.
- Nivel económico: Hipotecas, renta, contratos de trabajo
- Nivel educativo: Notas colegio de la zona...
- Cultura y ocio: Número de museos, cines, establecimientos deportivos, discotecas...
- Calificación de los usuarios: Basado en las valoraciones que realizan los propios usuarios, votarán del 1 al 10 y podrán añadir un comentario.

Fuentes de datos públicas a consultar

Explicación uso de API: https://www.zaragoza.es/sede/portal/datos-abiertos/api

API ZARAGOZA: https://www.zaragoza.es/docs-api_sede/#/

http://www.zaragoza.es/api/catalogo

Barrios de Zaragoza: GET http://www.zaragoza.es/api/recurso/sector-publico/territorio/barrio.json

Delimitación de barrios: GET http://www.zaragoza.es/demografia/juntas_polig_2017.json

Visualización de delimitación de barrios:
https://www.zaragoza.es/sede/portal/idezar/mapa/demografico/?id=juntas_Renta_per_capita#

- **Conectividad:** Indicadores de tráfico y movilidad del ayuntamiento de Zaragoza: https://zaragoza.es/sede/servicio/catalogo/2300
  Aparcamientos Públicos: https://zaragoza.es/sede/portal/datos-abiertos/servicio/catalogo/55
  Estaciones y uso bizi: https://zaragoza.es/sede/portal/datos-abiertos/servicio/catalogo/70
  Zonas de estacionamiento regulado: https://zaragoza.es/sede/portal/datos-abiertos/servicio/catalogo/332
- **Demografía:** Indicadores Demográficos: Infancia y Juventud: https://zaragoza.es/sede/servicio/catalogo/2140
  Distribución de la población: https://zaragoza.es/sede/portal/datos-abiertos/servicio/catalogo/1460
  Estructura de edad: porcentaje de la población de distintas edades: https://zaragoza.es/sede/portal/datos-abiertos/servicio/catalogo/2000

  Edad media: http://www.zaragoza.es/demografia/juntas_point_Edad_media_2017.json

Número de hogares por barrio:

GET http://www.zaragoza.es/demografia/juntas_point_Num_hogares_2016.json

- **Contaminacion:** BBDD de contaminación del aire por zonas: http://www.zaragoza.es/ciudad/medioambiente/atmosfera/redconta/horas_CalidadAire
- **Economía:** BBDD de rentas y coste de vivienda:
  BD de edificación y vivienda para extraer coste medio de la vivienda cada trimestre: https://zaragoza.es/sede/servicio/catalogo/313

  BD con la renta media per cápita en cada distrito de Zaragoza: https://zaragoza.es/sede/servicio/catalogo/1620

GET http://www.zaragoza.es/demografia/juntas_points.json

- **Cultura y Ocio:** BBDD de negocios y ocio
  - Museos: https://opendata.aragon.es/datos/catalogo/dataset/apertura-de-museos-y-colecciones-museograficas
  - Equipamientos culturales, deportivos, educativos (colegios, institutos...), servicios sociales... https://zaragoza.es/sede/servicio/catalogo/300

- Centros de salud: http://idezar.zaragoza.es/wfss/wfss?request=GetFeature&featureType=PuntosDeInteres&propertyNames=posicion,url,nombre,icono_grande&subtema=Centros%20de%20Salud
- Farmacias: https://zaragoza.es/sede/servicio/catalogo/87

Apis externas a utilizar

- Mapas (Open Street Maps): Overpass: https://wiki.openstreetmap.org/wiki/API_v0.6 utilizando https://leafletjs.com/
- Verificación cuentas google: Usando Node.js
- Analítica de la app: chart.js

Utilidad

- A la hora de mudarse a una zona nueva.
- Para analizar donde es más necesario aplicar medidas legislativas.
- A la hora de abrir un nuevo negocio en una zona u otra.
- Buscar relaciones entre nivel de ocio y nivel económico u otros parámetros.

Público al que va dirigida

Público entre 18 y 60 años en búsqueda de información relevante sobre diferentes zonas en Zaragoza con múltiples propósitos.
Por ejemplo puede ser útil para usuarios cuyo objetivo es comenzar a vivir en Zaragoza, pues pueden hacerse una idea de cómo es cada zona; o usuarios que quieren abrir un negocio, pues pueden ver un poco en qué zona puede tener más éxito su idea de negocio. Incluso para el ayuntamiento y órganos de gestión puede ser útil para saber qué mejorar en cada zona, pues tendrán comentarios de los habitantes y sugerencias.

Funcionalidad

Mapa de la zona con puntuaciones en cada parámetro.

Ranking de zonas ordenadas por parámetros (Ejemplo lista zonas menos peligrosas, zonas con más nivel de ocio).

Requisitos

RF1	El usuario podrá crearse una cuenta.
RF2	El usuario podrá iniciar sesión en su cuenta.
RF3	El usuario podrá autenticarse utilizando su cuenta de google.
RF4	El usuario podrá eliminar su cuenta.
RF5	El administrador podrá iniciar sesión en una cuenta con privilegios de administración.
RF6	El usuario podrá consultar un ranking de barrios ordenados según un parámetro de clasificación.
RF7	El usuario podrá consultar los parámetros de calificación de un barrio.
RF8	El usuario podrá consultar los parámetros de calificación de un barrio buscando por calle.
RF9	El usuario podrá filtrar los barrios especificando los valores de los parámetros de calificación (filtros personalizados).
RF10	El usuario podrá realizar una valoración de cada barrio y añadir un comentario con esta.
RF11	El usuario podrá clasificar según la valoración o por más recientes los comentarios de otros usuarios.
RF12	El usuario podrá observar analíticas de uso de la aplicación.
RF13	El administrador podrá eliminar cuentas.
RF14	El administrador podrá eliminar comentarios.
RF15	El administrador podrá desactivar los comentarios para una cuenta.
RF16	El administrador podrá observar analíticas de uso de la aplicación.
RNF1	El sistema deberá poder autenticar usuarios con los servicios de google.
RNF2	Los rankings se actualizarán en tiempo real.
RNF3	El sistema debe tener acceso a diversas fuentes de datos públicas.
RNF4	El sistema debe interactuar con APIs de servicios de terceros.

Diccionario de datos:

Parámetro de calificación: contaminación, conectividad, demografía, nivel económico, nivel educativo, cultura y ocio y valoración según usuarios.

Analíticas:

Las analíticas de uso para cada usuario serán las siguientes:

- Usuario:
    - Número accesos a la app tanto individual como global: “Has accedido a esta aplicación X veces esta semana de un total de X+Y accesos”.
    - Número de visualizaciones de un barrio en la última semana: “Este barrio ha sido consultado Z veces en la última semana”.
- Admin:
    - Número de accesos a cada fuente de datos: Accesos x fuente.
    - Número total de usuarios.
    - Número de usuarios administradores.
    - Número de usuarios baneados.
    - Número de comentarios por barrios: Comentarios x barrio.
    - Número de accesos a la aplicación.
    - Número de visualizaciones de un barrio en la última semana: “Este barrio ha sido consultado Z veces en la última semana”.
    - Número de valoraciones de un barrio.

Propuestas similares

En el mercado actual de aplicaciones encontramos diversas aplicaciones de ámbito local que tratan aspectos de la ciudad de Zaragoza.

Algunos ejemplos van desde aplicaciones creadas por fuentes públicas para consultar horarios de autobuses, estaciones de bicis, etc hasta otras privadas para buscar grupos de ocio y visitar Zaragoza. Es muy probable que algunas de estas aplicaciones utilicen datos de las bases de datos del ayuntamiento para nutrir la app.

Sin embargo, hemos encontrado pocas aplicaciones con un enfoque similar en el mercado actual. De aquellas que también tratan de barrios se pueden sacar ideas adicionales a la aplicación desarrollada que podrían complementarla en versiones posteriores. Algunos ejemplos son:

- Barrios Activos de *Kit Urbano:* permite reportar informes ciudadanos en las calles. Esto podría servir para notificar de pavimentos en mal estado, robos o agresiones, etc
- ¿Tienes sal? De *Good Good GmbH*: genera una red social alrededor de los barrios buscando que la gente se conozca
- Nextdoor de Nextdoor.com: ayuda a crear eventos en el barrio, vender cosas que no necesitas, etc

Arquitectura de alto nivel

Cierzo es una aplicación web 3-tier, que se ha desarrollado usando la metodología stack MEAN, compilación de tecnologías para desarrollar aplicaciones web usando como único lenguaje de programación javascript en el servidor, en el cliente y en la base de datos.

El desarrollo del Frontend se ha realizado con Angular, mientras que el desarrollo de la API se ha desarrollado con Node y Express. Para la persistencia de datos se ha usado una base de datos no relacional MongoDB almacenada en un cluster de Atlas.

Diagrama de módulos

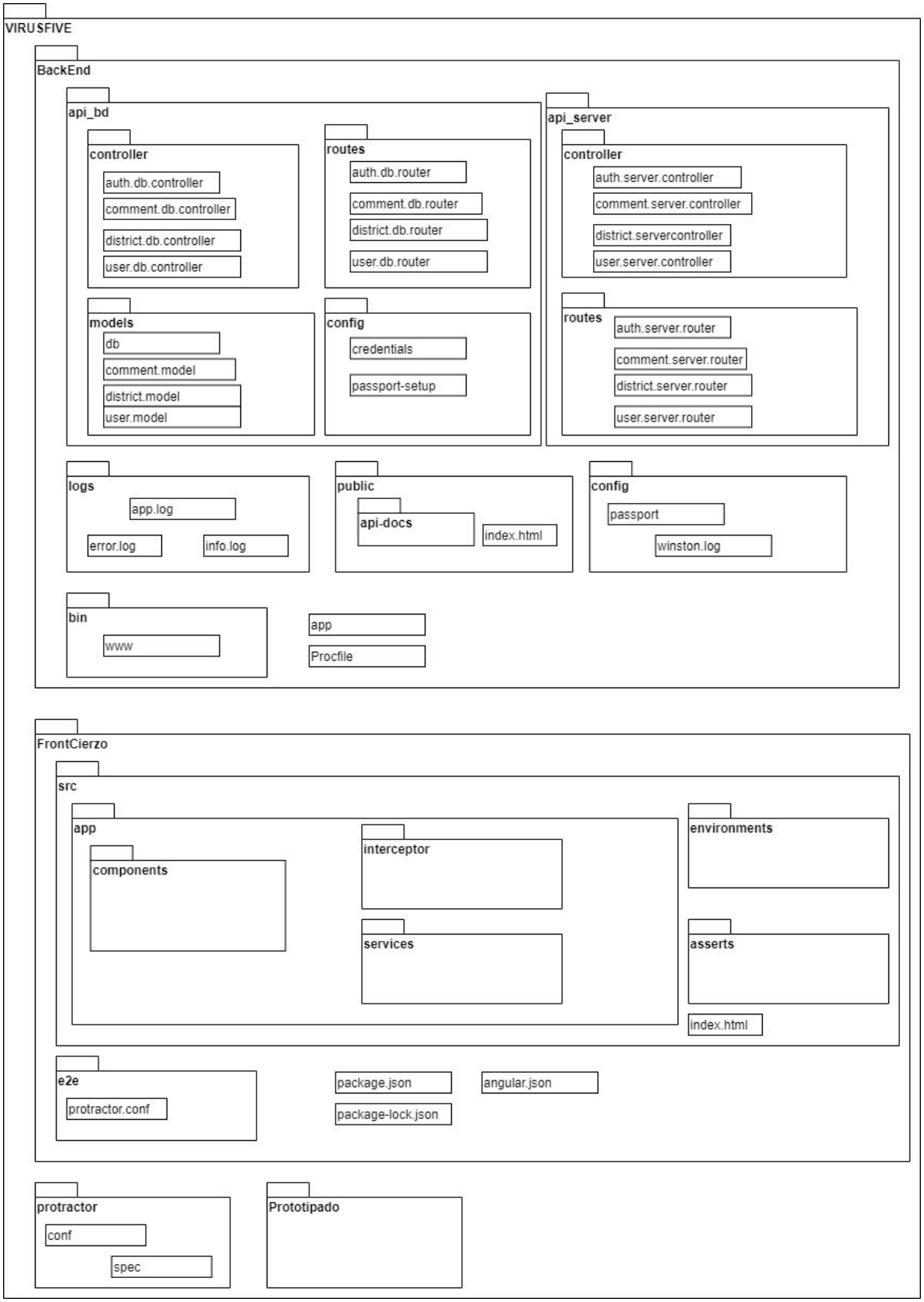


Diagrama de componentes y conectores

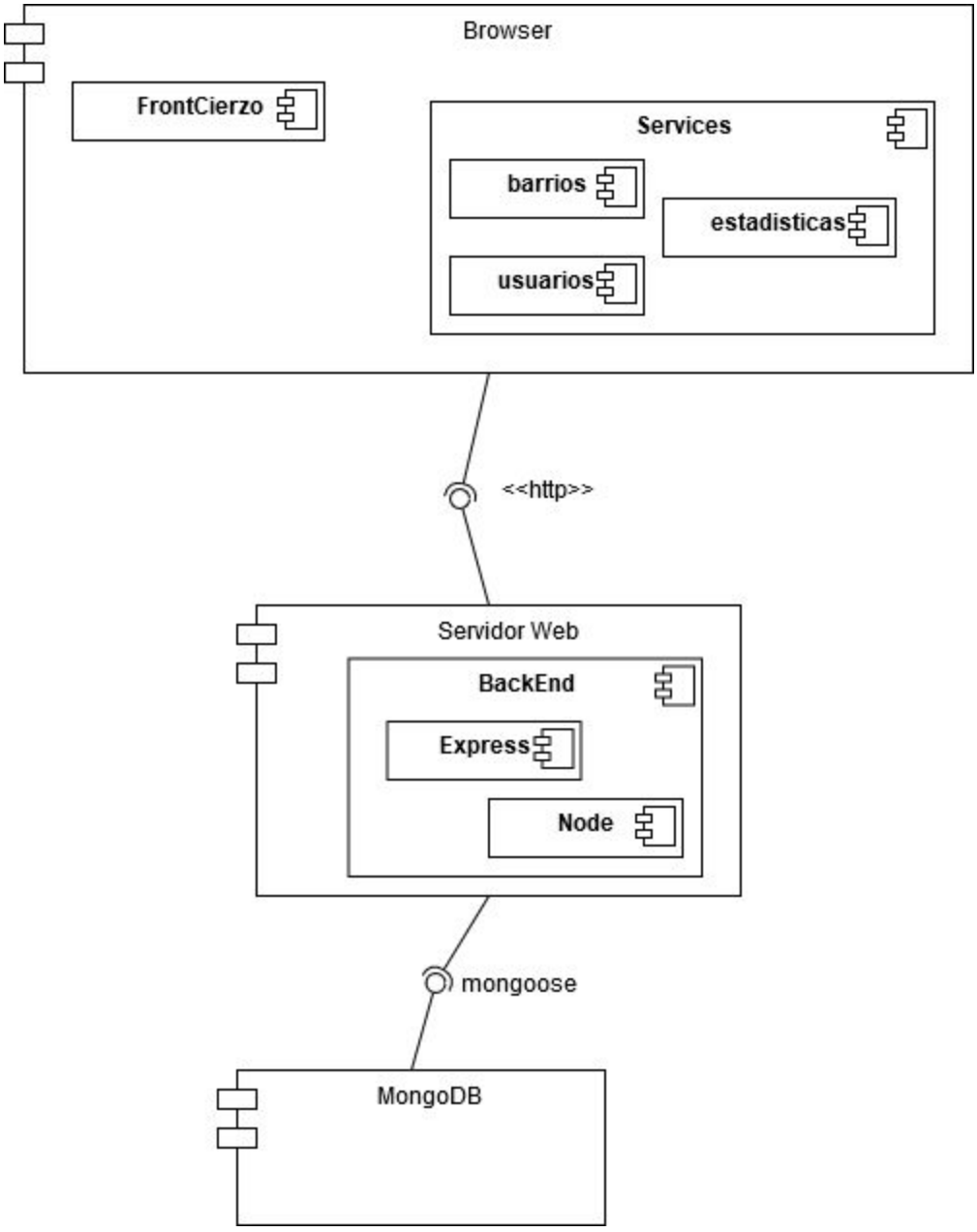
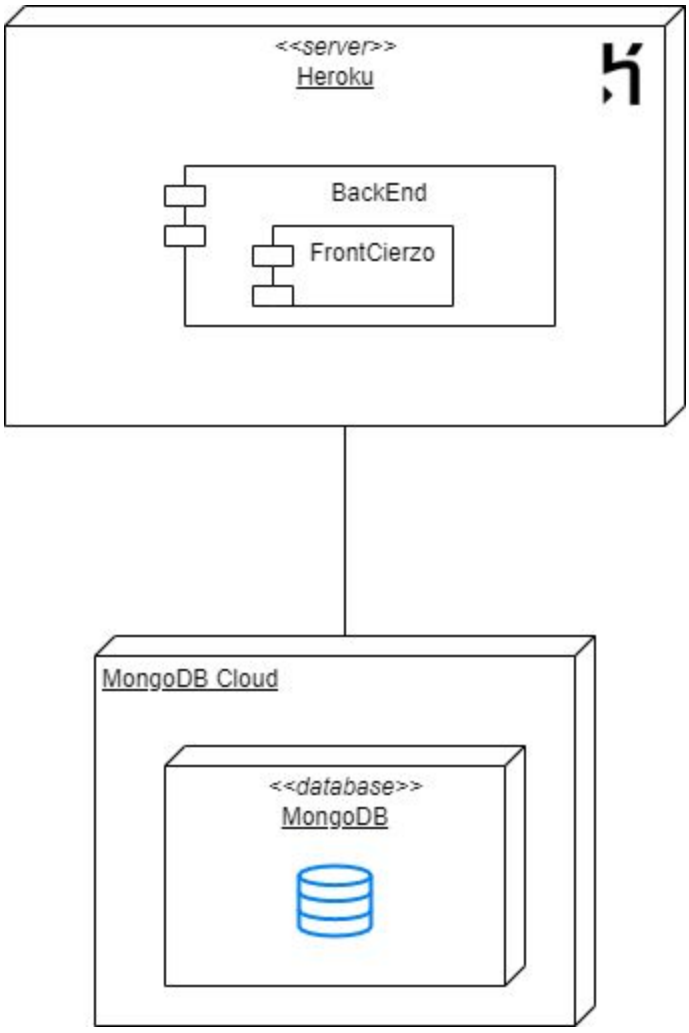


Diagrama de despliegue



Modelo de datos

La base de datos consta de tres tipos de esquemas distintos. Dos de ellos modelan los barrios, el primero contiene información sobre la puntuación subjetiva, las valoraciones de los usuarios y los comentarios; el segundo se centra más en las valoraciones objetivas que se le da a cada parámetro con la información obtenida del ayuntamiento.. El tercero modela los usuarios.
Un usuario se modela mediante un esquema que contiene:

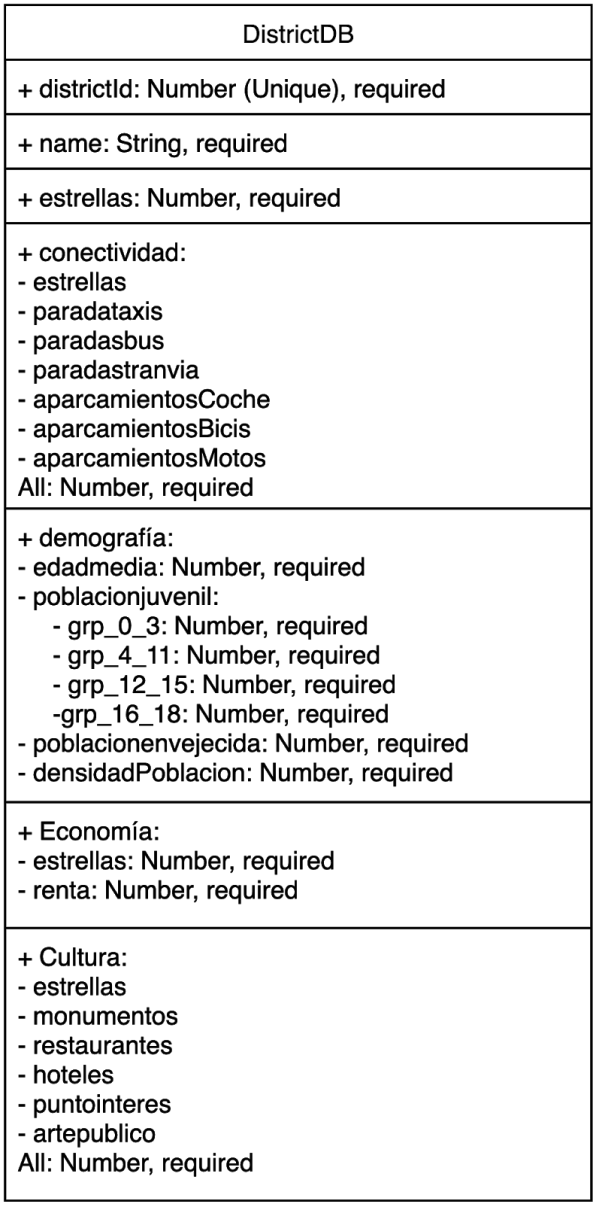
- El nombre
- La dirección de correo
- La contraseña
- El número de veces que se ha conectado
- La fecha de su creación
- El último día que se conectó
- Si está baneado o no (mediante un booleano)
- Si es o no administrador

User
+ Name: String
+ email: String (Unique), Required
+ password: String
+ logins: Number, default '0'
+ lastLoginOn: Date, default 'now'
+ createdOn: Date, default 'now'
+ banned: boolean, default 'false'
+ admin: boolean, default 'false

El barrio y su puntuación objetiva consta también de un único esquema con:

- El identificador
- El nombre del barrio
- La puntuación general contabilizada en estrellas (de 0 a 5)
- La conectividad, que a su vez contiene
    - Las puntuación en estrellas (de 0 a 5)
    - El número de paradas de taxi
    - El número de paradas de bus
    - El número de paradas de tranvía
    - El número de aparcamientos para coche
    - El número de aparcamientos para bicicletas
    - El número de aparcamientos para motos
- La demografía, que a su vez contiene

    - La edad media de los habitantes del barrio
    - La población juvenil, dividida en cuatro grupos
        - Grupo de 0 a 3 años
        - Grupo de 4 a 11 años
        - Grupo de 12 a 15 años
        - Grupo de 16 a 28 años
    - El índice de población envejecida
    - La densidad de población
- La economía, con:
    - Puntuación en estrellas (de 0 a 5)
    - La renta per cápita
- La cultura, con:
    - La puntuación en estrellas (de 0 a 5)
    - El número de monumentos
    - El número de restaurantes
    - El número de hoteles
    - El número de puntos de interés
    - El número de obras de arte público



El barrio con la puntuación de los usuarios, los comentarios, etc. se compone del siguiente esquema con sus subesquemas:

- El identificador
- El nombre
- La puntuación media de los usuarios en estrellas (de 0 a 5)
- El número de puntuaciones que ha recibido
- Un array con las valoraciones de los usuarios
- Un array con los comentarios de los usuarios
- El número de accesos que han realizado los usuarios a ese barrio

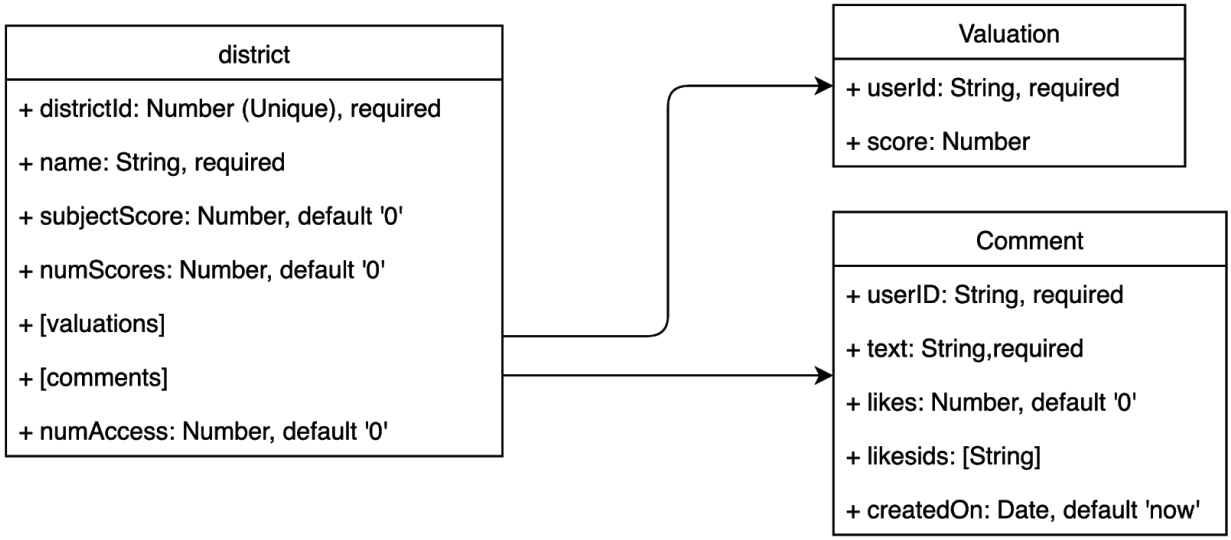
El subesquema de las valoraciones contiene:

- El identificador del usuario que ha realizado esa valoración

- La puntuación en estrellas que ha dado ese usuario

El subesquema de los comentarios contiene:

- El identificador del usuario que ha realizado ese comentario
- El texto del comentario
- La fecha de su creación
- El número de “Me gusta” que tiene
- Un array con los identificadores de todos los usuarios que han dado “Me gusta”



API REST

URL: http://cierzo.herokuapp.com/api-docs/

Se ha optado por realizar un enfoque Bottom-up:

1. Partir de la API REST que se ha implementado para la cual se quiere crear la definición Swagger.
2. Documentar cada uno de los servicios. *swagger-jsdoc* utiliza comentarios del estilo JSDoc para generar la especificación Swagger.
3. Añadir esos comentarios en YAML a los ficheros router, escribiendo su funcionalidad.

Para la documentación de la API se ha usado el módulo de Node *swagger-jsdoc*. Permite integrar Swagger utilizando comentarios del estilo JSDoc en el código. Simplemente se agrega la etiqueta @Swagger encima de su DocBlock y se declara el significado del código en YAML que cumpla con la especificación OpenAPI. Para la definición de estos comentarios se ha utilizando Swagger Editor [1]. *swagger-jsdoc* analiza el código y genera una especificación OpenAPI.

Implementación

Backend

Winston

Biblioteca de logs extremadamente versátil y la solución de registro más popular disponible para aplicaciones Node.js , basada en estadísticas de descarga de NPM. Las características de Winston[2] incluyen soporte para múltiples opciones de almacenamiento y niveles de registro, consultas de registro e incluso un generador de perfiles incorporado.

A menudo es útil mantener cualquier tipo de soporte o archivos de configuración para nuestras aplicaciones en un directorio especial, así que se ha creado una carpeta llamada config que contiene la configuración de este módulo. Dicho fichero se llama winston.log.js. Además, se ha creado una carpeta logs que contendrá sus archivos de registro.

Winston permite definir los ajustes de configuración para diferentes transportes. Los transportes son un concepto introducido por Winston que se refiere a los mecanismos de almacenamiento/salida utilizados para los registros. Winston viene con tres transportes principales: consola , archivo y HTTP . Para esta aplicación se ha utilizado transportes de consola y transporte de archivos. Cada definición de transporte puede contener sus propios ajustes de configuración, como el tamaño del archivo, los niveles de registro y el formato de registro. En este caso, se han creado tres tipos de transporte de archivo logs con diferentes

niveles. Uno con nivel error, otro con nivel info y por último un archivo sin nivel para almacenar todos los registros de la aplicación. Todos ellos tienen un tamaño máximo de 1024 * 1024 * 10. El formato usado para los mensajes ha sido el siguiente:

“'YYYY-MM-DD HH:mm:ss [info.level] : info.message”

JSON Web Token (JWT)

JSON Web Token (JWT) [3] es un estándar abierto (RFC 7519) que define una forma compacta y autónoma para transmitir información de forma segura entre las partes como un objeto JSON. Esta información se puede verificar y confiar porque está firmada digitalmente. Los JWT se pueden firmar usando un secreto (con el algoritmo HMAC) o un par de claves pública / privada usando RSA o ECDSA .

Para cada inicio de sesión se crea este token con la información del usuario y se firma con una clave privada. Una vez que el usuario haya iniciado sesión, cada solicitud posterior incluirá el JWT, lo que le permitirá acceder a rutas, servicios y recursos que están permitidos con ese token.

Cron

Cron [4] es una herramienta que permite ejecutar una tarea en un horario determinado. Esto normalmente se hace usando la sintaxis cron y permite ejecutar una función cada vez que se active el trabajo programado. Los patrones de Cron admitidos se extienden en el formato estándar de Unix para admitir dígitos de segundos, dejarlo desactivado será 0 y coincidirá con el comportamiento de Unix.

Campos necesarios:

- Segundos: 0-59
- Minutos: 0-59
- Horario: 0-23
- Día del mes: 1-31
- Meses: 0-11 (enero-diciembre)
- Día de la semana: 0-6 (dom-sáb)

Un campo puede ser un asterisco (*), que siempre significa “primero-último”'.

Se permiten rangos de números. Los rangos son dos números separado con un guión. El rango especificado es inclusive. Por ejemplo, 8-11 para una entrada de “horas”' especifica ejecución a las 8, 9, 10 y 11 horas.

Las listas están permitidas. Una lista es un conjunto de números (o rangos) separado por comas. Ejemplos: “1,2,5,9”, “0-4,8-12“.

Dicho módulo se ha utilizado para la descarga de los datos proporcionados por la API del ayuntamiento de Zaragoza. Esta información una vez se ha descargado, se almacena en la base de datos de la aplicación. La descarga se realiza todos los lunes del año a las 04:00 de la mañana. La especificación se ha realizado mediante la siguiente expresión cron: 0 0 4 * * MON.

Método evaluación de barrios

Se ha procurado emplear un método de evaluación lo más justo posible. Para ello, las magnitudes cuantitativas se han dividido entre la densidad de la población, para obtener una tasa más justa. Así, un barrio con pocas paradas de autobús pero en el que vive muy poca gente no está necesariamente peor valorado que aquel que tiene muchas paradas pero una densidad muy alta.
Se pensó que si se añadía a esta ecuación, por ejemplo, las frecuencias de los buses, tranvías, etc. se haría una valoración más justa todavía, pero complicaba la implementación y había que programar a mano todos esos datos ya que no se encuentran como datos abiertos.

Frontend

Para la parte de frontend se ha exprimido al máximo Angular Material, esto hace que la aplicación tenga un aspecto uniforme y que, una vez implementados los primeros componentes, el ritmo de trabajo no se haya detenido por problemas de implementar nuevas bibliotecas o configuraciones diferentes.

Cierzo incluye dos tipos de vistas, antes de iniciar sesión y después de iniciar sesión. Estas últimas, comparten todas una toolbar y un side navigator con los accesos al resto de pantallas. Para evitar la replicación de código, se ha implementado un componente con la toolbar y el side navigator que es padre de todas las vistas excepto del inicio de sesión y registro de nuevo usuario.

La página principal presenta un mapa implementado gracias a la biblioteca *Leaflet*, esta consume de Open Street Map y de un fichero con las geometrías de los barrios de Zaragoza. En la configuración de dicho mapa se incluyen límites para que el usuario no pueda salir del área metropolitana de Zaragoza, un buscador para localizar una calle concreta, al pasar el cursor sobre un barrio queda resaltado y al clicar sobre un barrio se hace zoom para que ocupe el centro de la pantalla.

Otro aspecto importante del frontend es el uso de interceptores, cada vez que este realiza una petición al backend se intercepta para ver si lleva el token y por tanto se ha iniciado sesión. En caso de que no se haya iniciado sesión el usuario es rebotado a la página de inicio de sesión.

Para realizar gráficas se han utilizado las librerías de node *Chart.js* y *ApexCharts.js*. Todas las gráficas están realizadas con *Chart.js* excepto una, la de las valoraciones de los barrios. Se buscaba un diagrama de barras horizontal, elemento del cual *Chart.js* no dispone, por lo que se decidió incluir la librería *ApexCharts.js* en el proyecto e implementar esta gráfica.

Para realizar la paginación se ha utilizado el módulo de node *ngx-pagination*. Esta librería sencilla permite personalizar el número de elementos a mostrar en cada página así como la barra de navegación entre las mismas. Únicamente se modificaron las etiquetas de la navegación para tenerlas en castellano y el número de elementos por página, que se estableció en 5.

Desarrollando la aplicación surgieron varias cuestiones de implementación a resolver. Una de ellas fueron la implementación de popups. Se implementan como componentes aparte del resto. Normalmente los popups tienen funcionalidad ya que suelen ser para confirmar una acción. Esto nos lleva a otro problema, se debe llamar a funciones que se encuentran en otros componentes desde el popup. La manera de solucionar esta cuestión es importando el componente padre en el componente, instanciando el mismo dentro del componente y ejecutando la función deseada.

El problema anterior viene ligado con el de actualizar elementos dinámicos dentro de una página sin tener que refrescar la página completa, como por ejemplo al publicar un comentario. Esto se consigue implementando la función de refrescar que lee del backend y vuelve a crear el objeto concreto dentro del componente.

Modelo de navegación

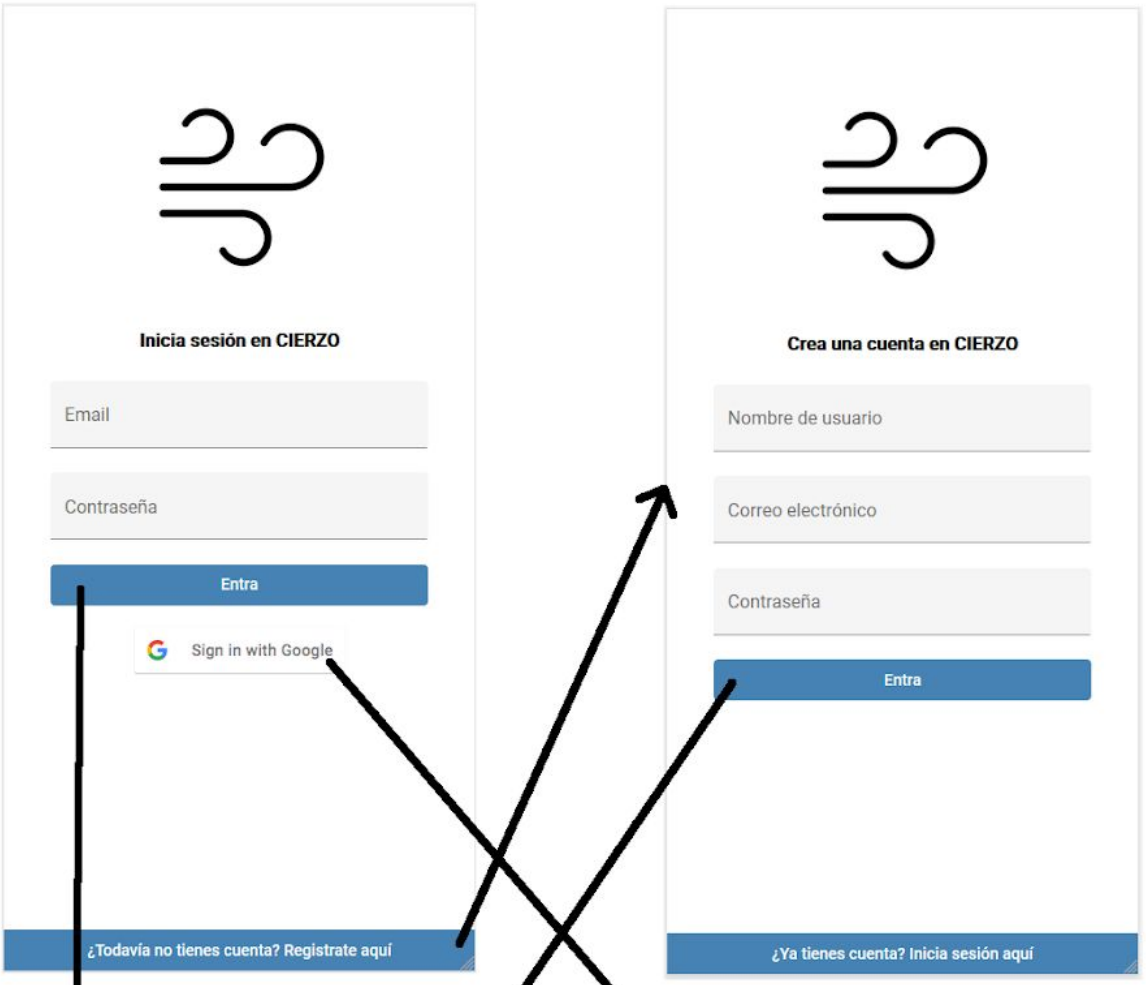
A continuación se presentarán las pantallas que integran el sistema, se incluirá una breve descripción de qué se puede hacer en ellas y de cómo funcionan por debajo.

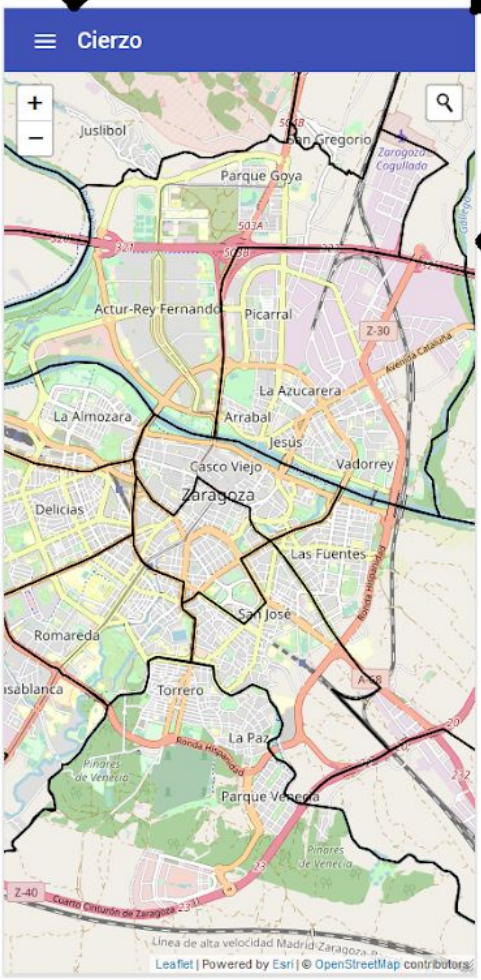
La mayoría de las pantallas realizan peticiones HTTP para obtener la información necesaria a mostrar en el frontend. También se realizan peticiones de este tipo para añadir o modificar la info en la bbdd.

En primer lugar se tiene la pantalla de iniciar sesión, la cual tiene un formulario para iniciar sesión introduciendo el correo y la contraseña y también permite el inicio de sesión con google. En el inferior de la misma encontramos un botón para ir a la pantalla de registro.

De esta pantalla se puede acceder a otras tres, en caso de iniciar sesión con correo y contraseña accederemos a la pantalla principal de la aplicación (mapa). En caso de querer registrar un nuevo usuario se pulsa sobre el botón inferior y aparecerá la pantalla de registro, la cual está compuesta de un formulario sencillo y de un botón para crear la cuenta e iniciar sesión. Si se decide iniciar sesión con google se accede a una pantalla intermedia en la que uno puede autentificarse escogiendo una cuenta de google. Si se opta por esta última opción, se conecta con una api de google con la cual se obtienen datos de la cuenta autenticada suficientes como para crear un usuario nuevo (si no existe) e iniciar sesión.

El inicio de sesión normal conecta con el servicio de autenticación creado en backend y que comprueba la contraseña introducida con la almacenada en la BBDD. El registro de un nuevo usuario también llama a un servicio implementado en backend para almacenar la cuenta nueva en la BBDD. El inicio de sesión con google conecta con el servicio passport para realizar la autenticación con los servicios de google y crear una cuenta en la BBDD con la información obtenida de esta cuenta..


Inicia sesión en CIERZO
Email
Contraseña
Entra
Sign in with Google
¿Todavía no tienes cuenta? Regístrate aquí
Crea una cuenta en CIERZO
Nombre de usuario
Correo electrónico
Contraseña
Entra
¿Ya tienes cuenta? Inicia sesión aquí


Cierzo
Juslibol
San Gregorio
Parque Goya
Actur-Rey Fernando
Picarral
La Azucarera
La Almozara
Arrabal
Jesús
Casco Viejo
Vadorrey
Zaragoza
Delicias
Las Fuentes
San José
Romareda
Torrero
La Paz
Parque Venecia
Leaflet | Powered by Esri | © OpenStreetMap contributors


Acceder con Google
Elegir una cuenta
para ir a Cierzo
JV
Usar otra cuenta
Para continuar, Google compartirá tu nombre, dirección de correo electrónico, preferencia de idioma y foto de perfil con Cierzo.
Español (Latinoamérica)
Ayuda
Privacidad
Condiciones

Una vez autentificado el usuario, ya se puede acceder a la aplicación y todas sus funciones. Estas serán diferentes según el tipo de usuario que sea. Administradores tendrán alguna funcionalidad extra, principalmente de gestión de usuarios y visualización de métricas.

En la pantalla del mapa se observa que hay a la izquierda un menú desplegable con las diferentes secciones accesibles. Se mostrará el perfil de un administrador dado que el de un usuario es igual pero con menos secciones. Desde esta pantalla podemos acceder a cualquier funcionalidad del sistema. Este menú desplegable es accesible desde cualquier pantalla una vez iniciada sesión y facilita de forma sustancial la navegación en la app. Como opciones principales encontramos “Mapa”, que es la pantalla que aparece una vez inicias sesión, “Ranking”, que presenta un listado de barrios ordenados según unos parámetros seleccionables y por último “Perfil”, donde aparece información sobre la cuenta y permite cambiar la contraseña y eliminar la cuenta. Además, en caso de ser cuenta de administrador, aparecen dos secciones más, las cuales son “Estadísticas”, donde se muestra información sobre los barrios, comentarios valoraciones y cuentas de usuario y “Gestión usuarios”, que permite ver todos los usuarios que tiene el sistema, así como da opción a eliminar cuentas o banear a las mismas (haciendo que no puedan publicar comentarios).

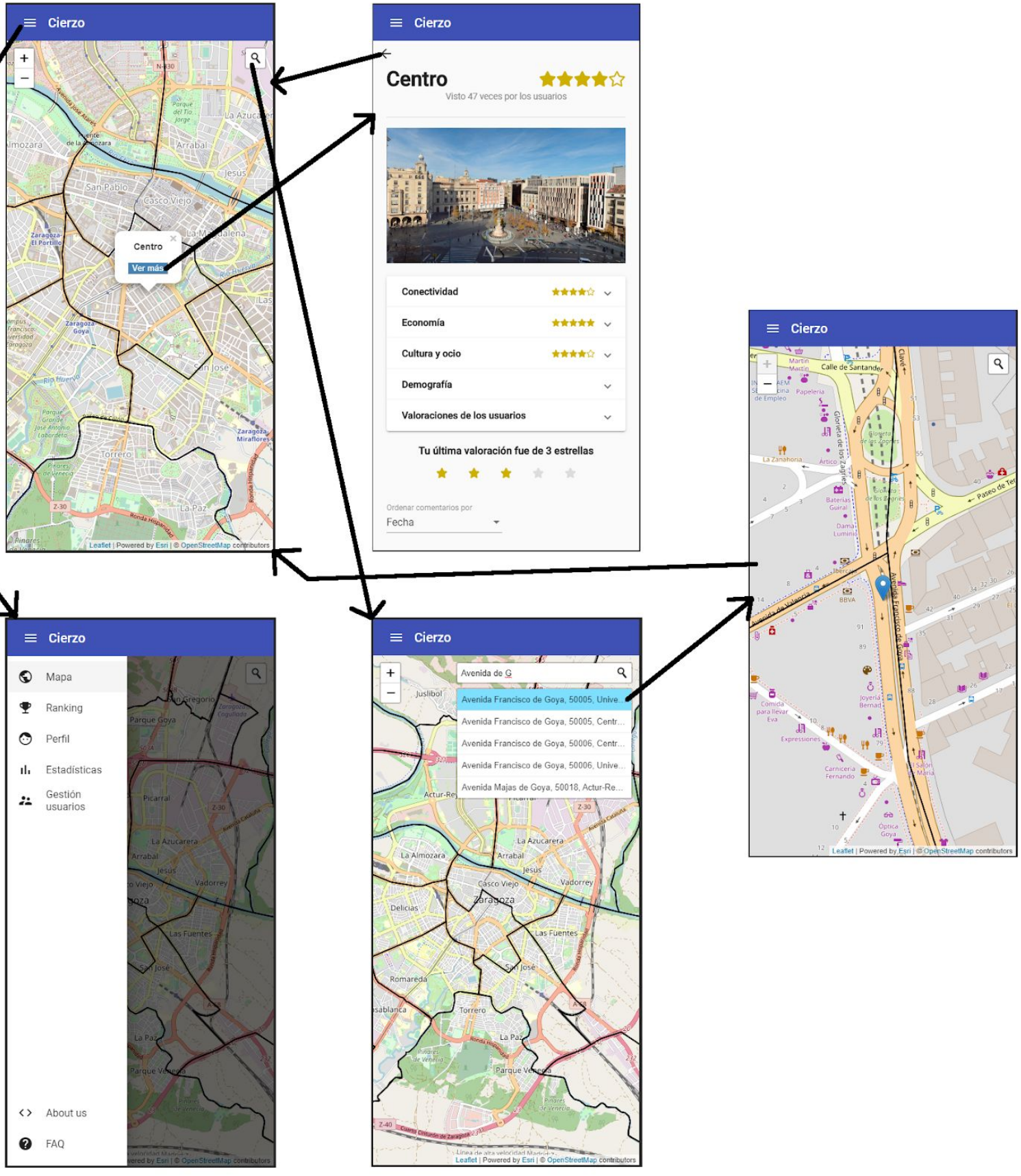
Finalmente, debajo de todas estas secciones, aparecen dos apartados más que son “About us” donde se describe brevemente los integrantes del equipo de desarrollo y “FAQ” donde aparecen preguntas frecuentes con su respuesta.

El mapa está implementado utilizando Leaflet, que permite integrar de manera sencilla un mapa utilizando OpenStreetMap. Se definieron los límites de los barrios según las coordenadas obtenidas de las fuentes de datos del ayuntamiento de zaragoza.

En primer lugar se puede hacer click sobre un barrio (delimitados por líneas negras) y pulsar sobre ver más. Haciendo esto accedemos a la pantalla de información de un barrio.

La información de los barrios se almacena en la BBDD y se actualiza semanalmente, lo cual agiliza el proceso de obtención de información ya que no es necesario obtener la información de las diferentes fuentes de datos del ayuntamiento.

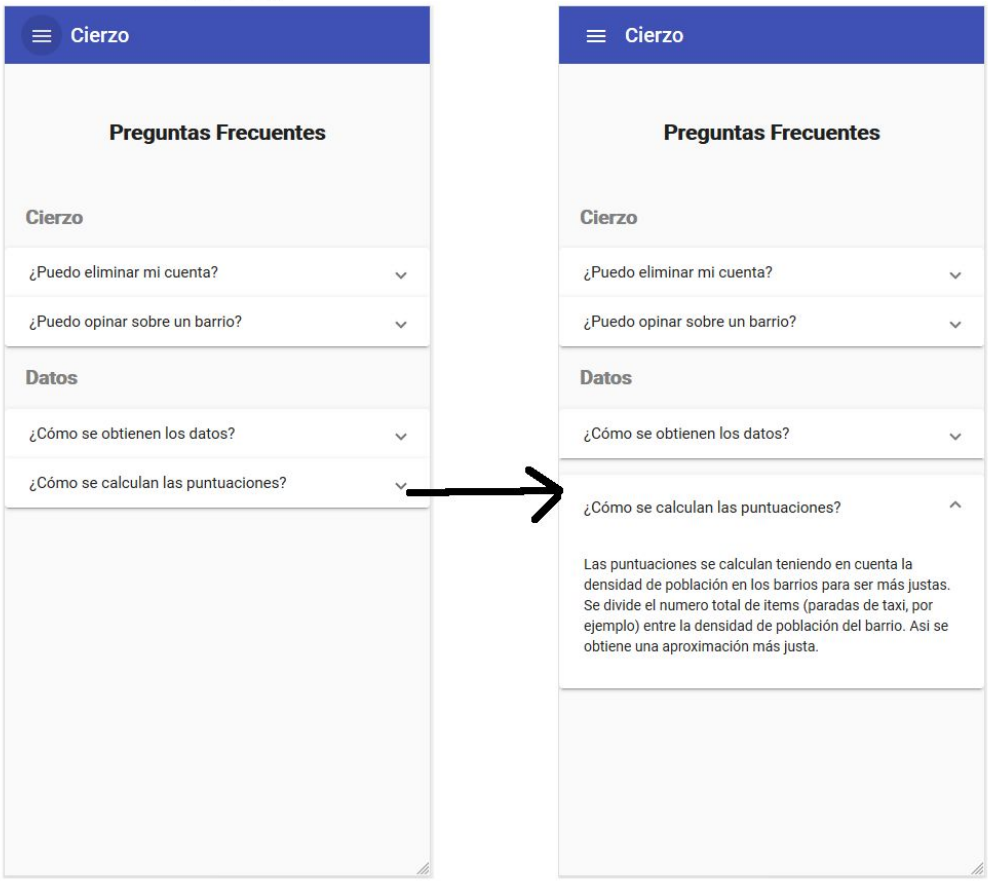
Al cambiar a la vista de información del barrio se realizan peticiones a la API REST para obtener la información relativa al ese barrio.


Cierzo
Centro
Ver más
Cierzo
Centro
Visto 47 veces por los usuarios
Conectividad
Economía
Cultura y ocio
Demografía
Valoraciones de los usuarios
Tu última valoración fue de 3 estrellas
Ordenar comentarios por
Fecha
Cierzo
Mapa
Ranking
Perfil
Estadísticas
Gestión usuarios
About us
FAQ
Cierzo
Avenida de G
Avenida Francisco de Goya, 50005, Unive...
Avenida Francisco de Goya, 50005, Centr...
Avenida Francisco de Goya, 50006, Centr...
Avenida Francisco de Goya, 50006, Unive...
Avenida Majas de Goya, 50018, Actur-Re...
Cierzo
Leaflet | Powered by Esri | © OpenStreetMap contributors

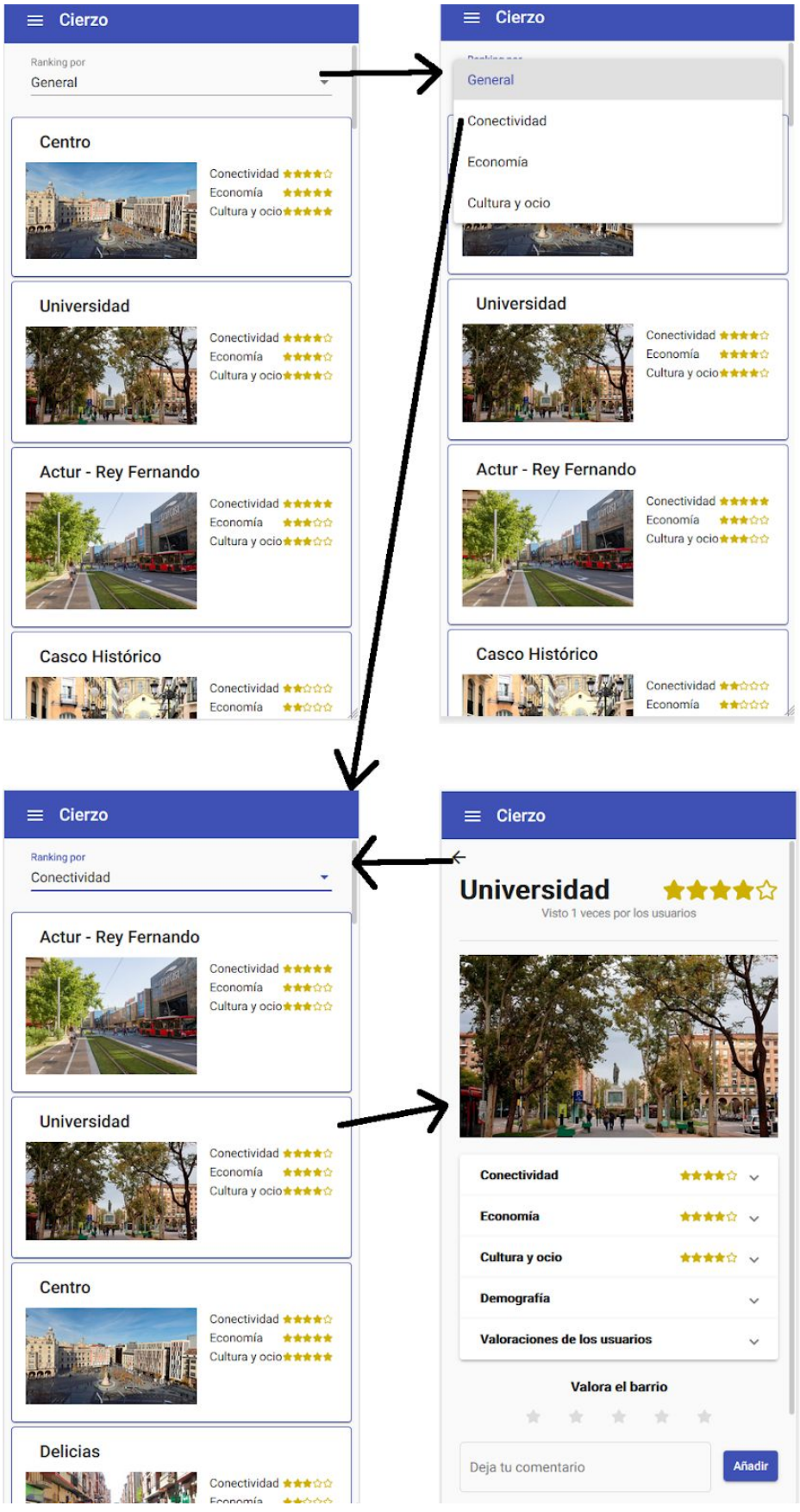
Las pantallas de “About us” y “FAQ” son páginas simples y estáticas donde se muestra información sobre el equipo y la aplicación. Se accede a ellas desde cualquier pantalla pulsando sobre la sección correspondiente en el menú lateral desplegable. En ellas también existe el menú desplegable lateral. La pantalla de FAQ está organizada mediante elementos desplegables, lo cual evita sobrecargar la página.

About us

FAQ



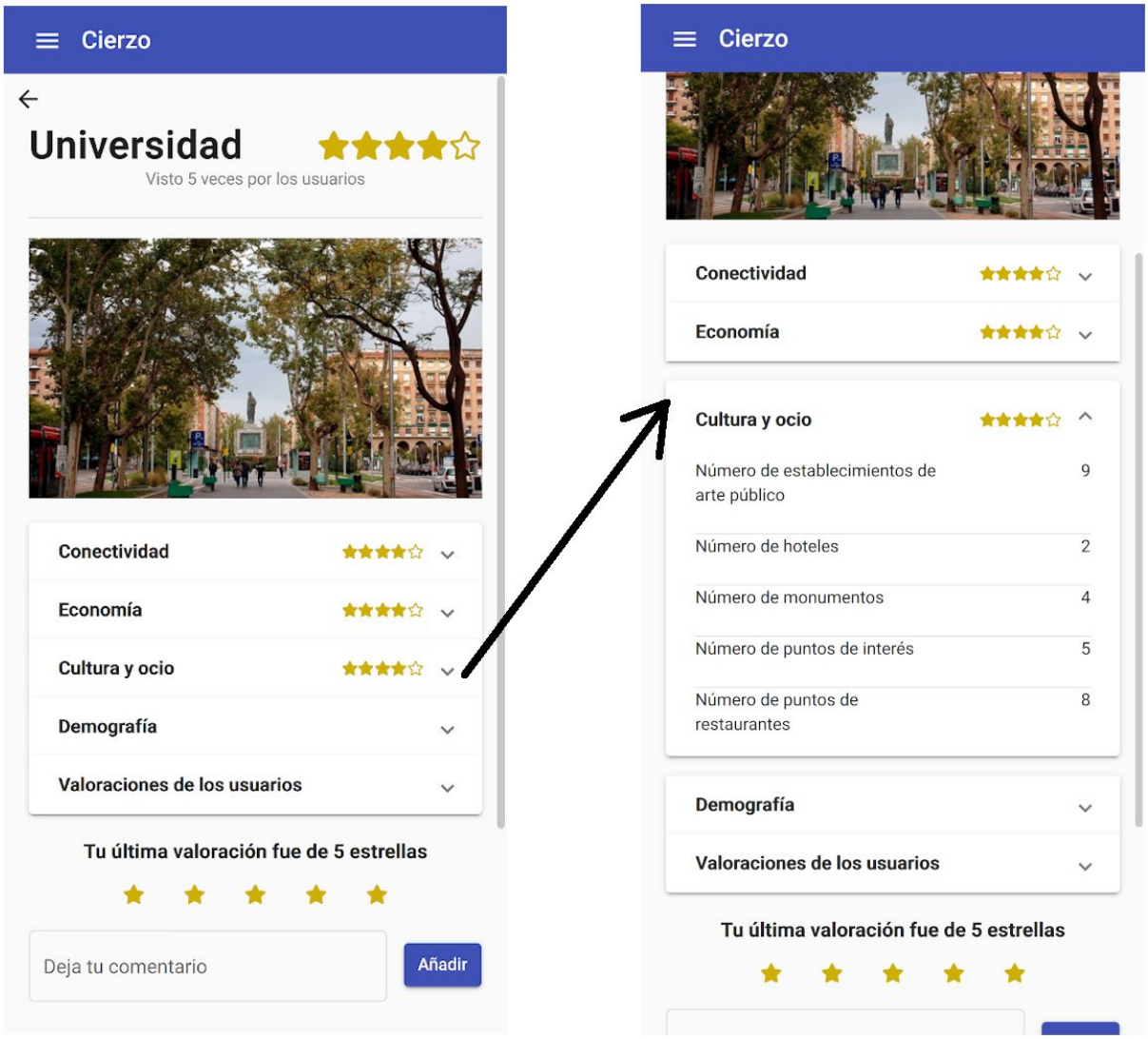
La siguiente pantalla en el menú desplegable se trata del “Ranking”. En ella encontramos un listado de todos los barrios y podemos seleccionar el criterio de ordenación. Además pulsando sobre un barrio accedemos a la pantalla de la información desglosada de ese barrio.



La información de los barrios se obtiene realizando una llamada a la API REST que devuelve únicamente los elementos de cada barrio necesarios para mostrarlos como aparecen en el ranking. La foto, identificador, nombre y número de estrellas de cada categoría.

Al ordenar el ranking seleccionando un parámetro se vuelve a realizar esta llamada pero enviando el parámetro de clasificación. La API devuelve entonces la lista ordenada según el parámetro enviando.

Una vez en la página de información de un barrio hay varias acciones que puede realizar el usuario. La primera sería desglosar la información que hay de cada barrio. Esto se hace pulsando sobre el nombre de la categoría de la cual queremos obtener información. Se desplegará ese apartado mostrando la información que se ha obtenido del ayuntamiento de zaragoza:

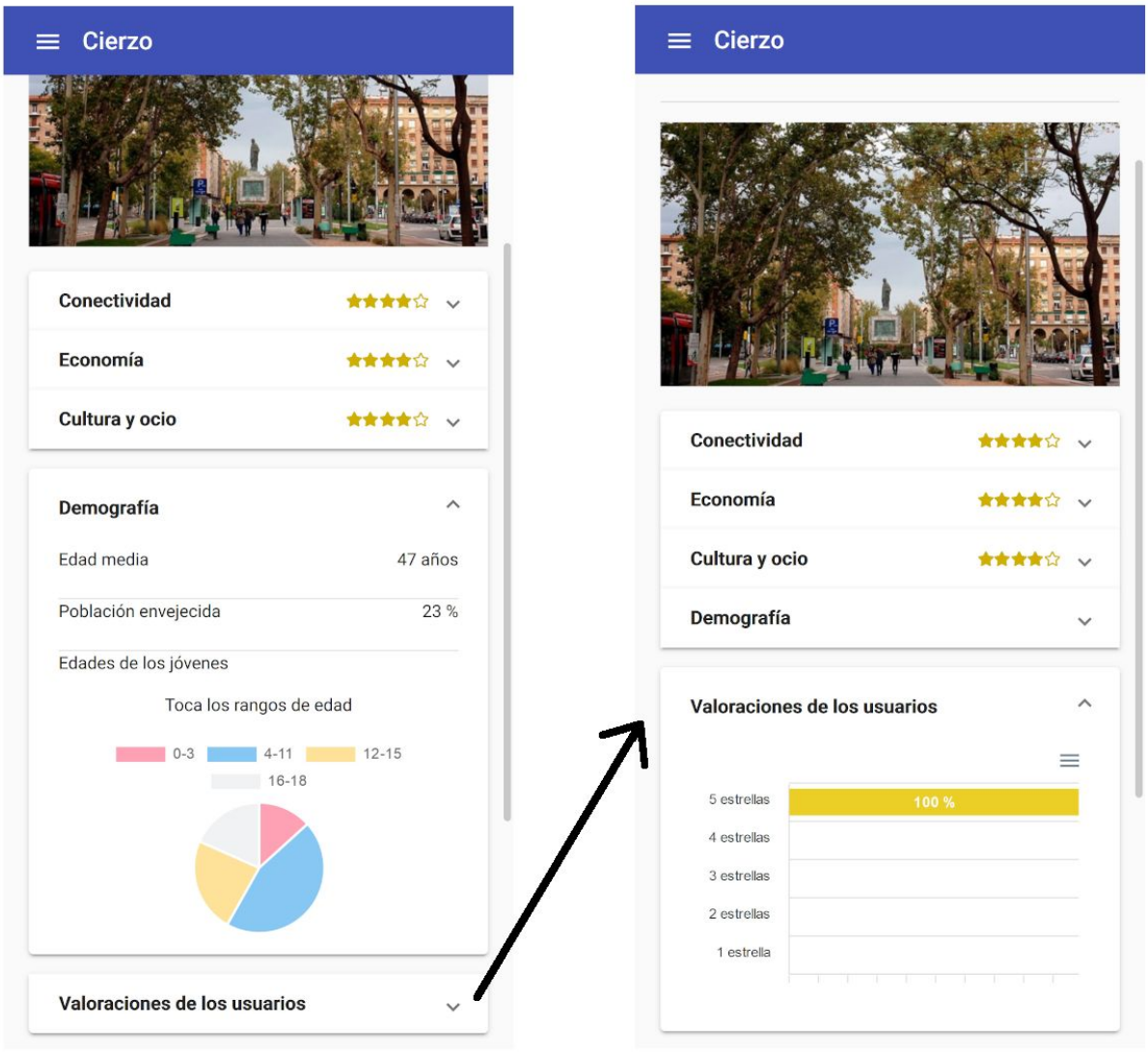


Al acceder a esta pantalla se realizan llamadas a la API REST para obtener toda la información a mostrar de un barrio. Una vez se ha obtenido la información se crean los gráficos de demografía y valoraciones que se explican en la siguiente página.

Cada vez que se realiza un cambio que afecte a los datos que se almacenan en esta pantalla, se realizan dos peticiones a la API, la primera para guardar estos cambios y la segunda para volver a obtener los datos que tiene almacenado el sistema. De esta forma aseguramos la consistencia entre lo que muestra la aplicación y lo que almacena el sistema. Esto ocurre cuando se publica un comentario nuevo, se actualiza la valoración del barrio, se da like a un comentario existente o se elimina algún comentario (o el usuario asociado al mismo).

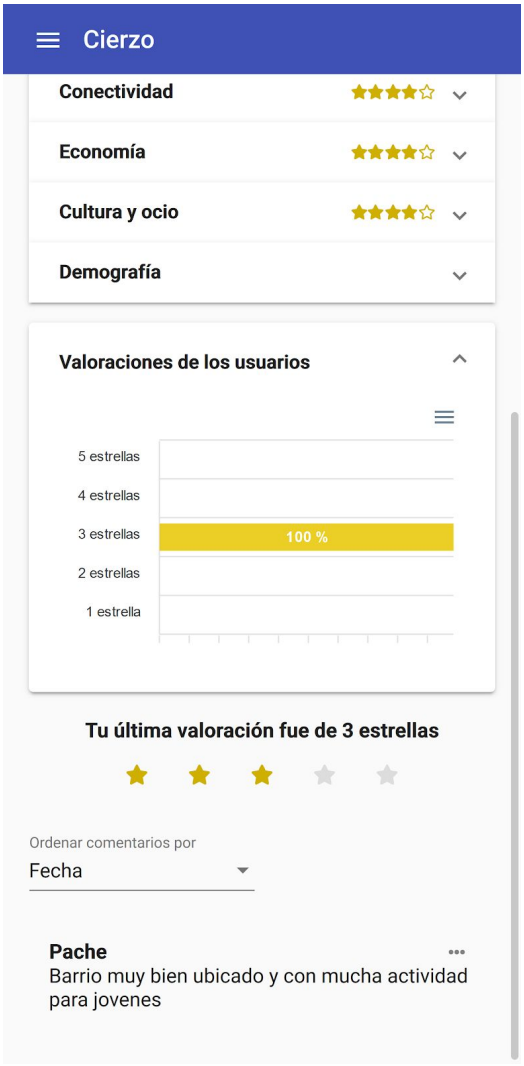
Las dos últimas categorías son algo especiales pues en vez de mostrar un listado de elementos del barrio aparecen gráficas que representan mejor la información. En el caso de la sección de “Demografía” tenemos la edad media, el porcentaje de población envejecida y una gráfica circular donde se ve a simple vista la cantidad de jóvenes que hay en el barrio y su distribución por edades. Podemos interactuar con esta gráfica pulsando en los rangos de edad de cada usuario, ocultando o mostrando los mismos en la gráfica.

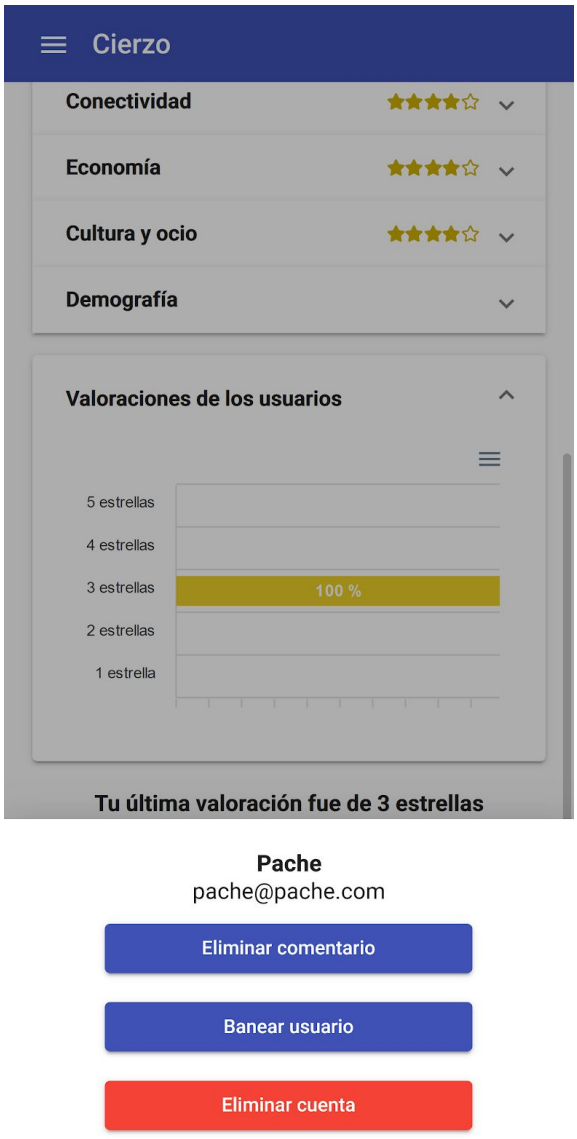
En la sección de “Valoraciones de los usuarios” tenemos una gráfica de barras horizontal en la que se muestra las valoraciones de los usuarios y que porcentaje de usuarios han votado cada estrella. (En el ejemplo solo hay un voto al barrio de la universidad por lo que el 100% de las valoraciones han sido de 5 estrellas).



También se puede valorar el barrio pulsando sobre las estrellas que se encuentran debajo de las secciones de información. Una vez pulsado se guarda y se actualiza la gráfica de valoraciones de los usuarios. No hay límite de veces que se puede valorar, sin embargo el sistema solo almacena la última valoración de cada barrio de cada usuario.

En la imágen de la derecha se aprecia como cambiando la valoración del usuario se cambia también la gráfica de porcentaje de estrellas. En este caso solo hay una valoración (de ahí el 100% en la valoración del usuario).



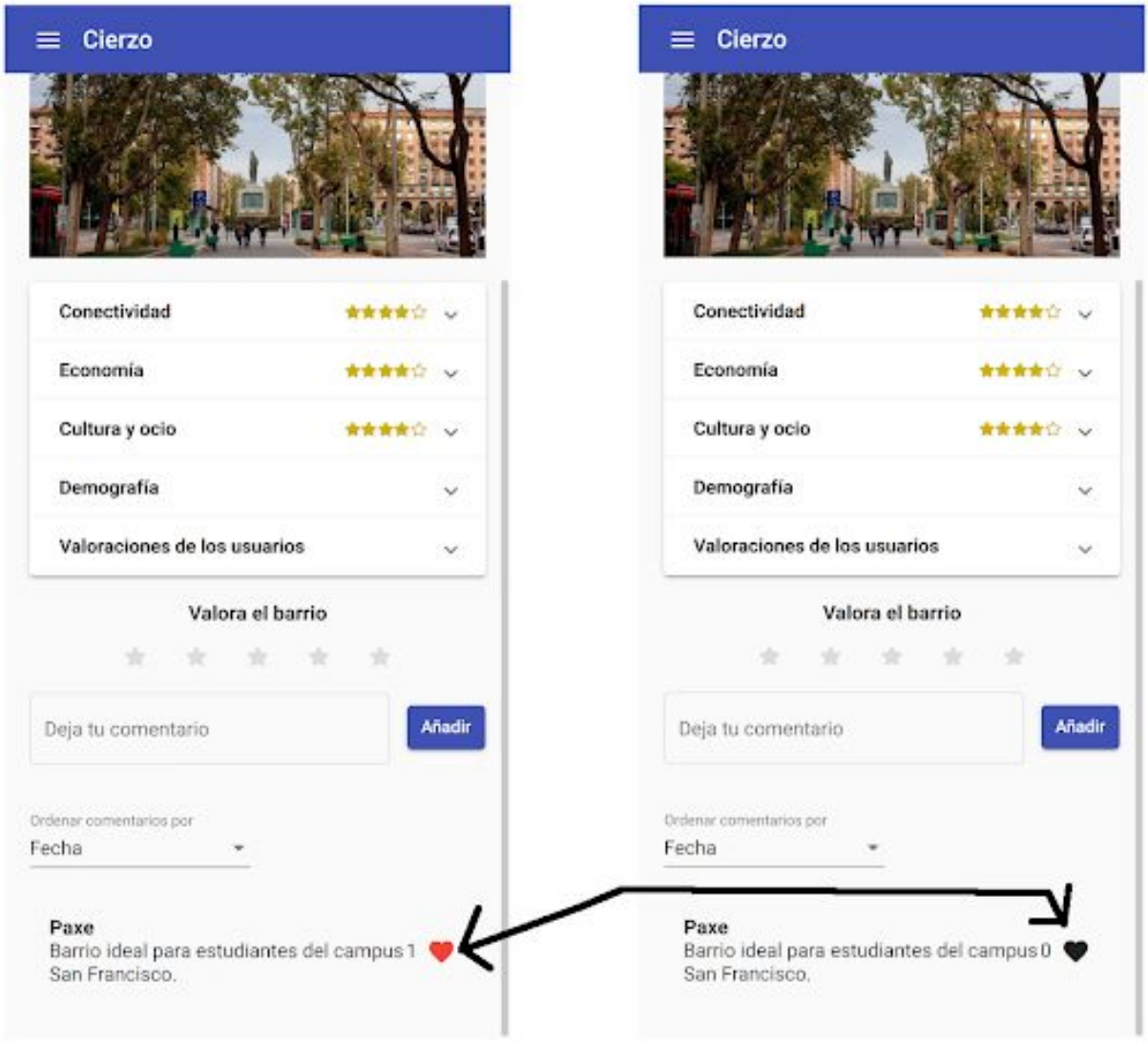


Por último está la sección de los comentarios. En esta sección hay distinción entre lo que puede hacer un usuario normal y un administrador. La función de ordenar los comentarios es común a ambos tipos de cuenta y funciona con un seleccionador desplegable (al igual que la ordenación del ranking de barrios).

La vista de arriba se trata de la de un administrador, en este tipo de cuentas aparecen tres puntos a la derecha del nombre del usuario que ha realizado el comentario. Pulsando sobre estos tres puntos aparecen las opciones de la imágen de la izquierda. Te permite eliminar el comentario, lo cual vuelve a la misma página actualizando los comentarios. Banear a un usuario

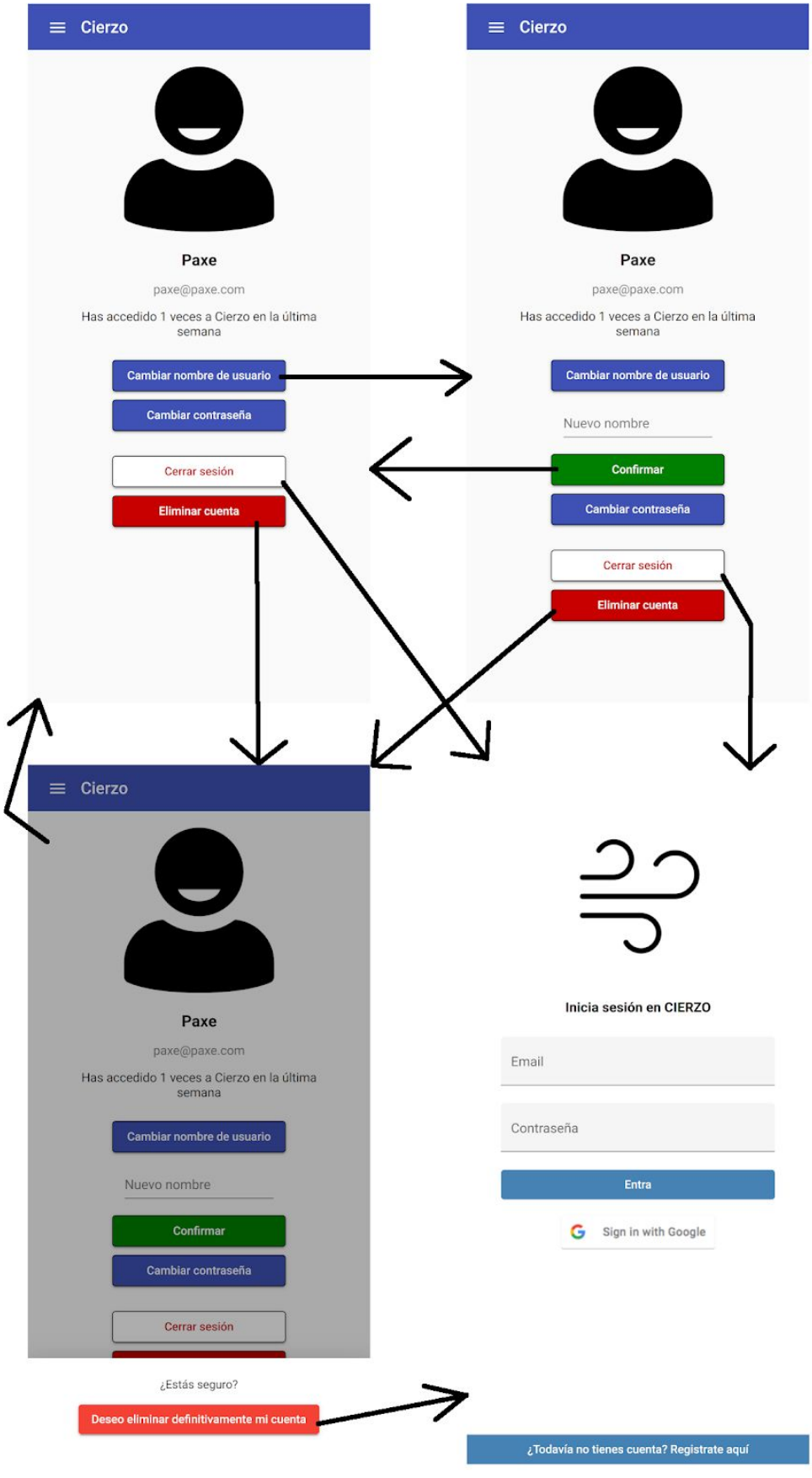
implica impedir que el mismo pueda publicar nuevos comentarios. Por último, la opción de “Eliminar cuenta” elimina la cuenta del usuario que ha realizado el comentario.

En el caso de tratarse de una cuenta normal de usuario aparecerá adicionalmente un cuadro de texto gracias al cual se puede añadir comentarios acerca del barrio. Además, en vez de los tres puntos aparecerá un corazón y un número entero que indica el número de “likes” que tiene este comentario. El corazón sirve para añadir un like al comentario. Sólo se puede dar un like por comentario. Al dar like se vuelve rojo el corazón y se suma uno al contador de likes. También se puede quitar este like volviendo a pulsar sobre el corazón.



Siguiendo con la lista de secciones del menú desplegable se llega a la pantalla de “Perfil”. En ella se muestra un icono de un usuario y se muestra información acerca de su cuenta, además de el número de inicios de sesión que ha realizado el usuario en la última semana. Aparecen también una serie de botones que permiten cambiar el nombre de usuario y la contraseña del usuario. Además hay dos botones adicionales que permiten al usuario cerrar la sesión actual

(eliminando el token de sesión) y eliminar su cuenta definitivamente del sistema. La navegación sería la siguiente:



Al entrar en esta página se realiza una llamada GET a la API REST para obtener la información del usuario asociado al token.

Las funciones de cambiar tanto el nombre de usuario como la contraseña realizan una llamada PUT a la API modificando la información asociada al usuario.

La función de eliminar cuenta realiza una llamada DELETE a la API que elimina al usuario de la BBDD.

La función de cerrar sesión elimina el token asociado a la sesión y redirige a la pantalla de inicio de sesión.

Ya se ha observado la navegación en todas las pantallas comunes a usuarios y administradores. Ahora se muestran las dos pantallas exclusivas para administradores. La primera se trata de “Estadísticas”, en ella aparecen dos pestañas: “Barrios” y “Cuentas”. En cada una aparecen gráficas que muestran información interesante acerca de los diferentes barrios que existen en el sistema y de todos los usuarios que lo conforman. En el apartado de “Barrios” hay tres gráficas de barras. Miden el número de visualizaciones de cada barrio, el número de valoraciones de cada barrio y el número de comentarios en cada barrio. Todo esto de manera sencilla y visual. Además, las gráficas en ambas secciones son interactuables, lo cual implica que pulsando sobre los elementos sale información adicional.

Al acceder a esta pantalla se realizan llamadas a la API para obtener los datos relativos a los barrios y usuarios.

Una vez obtenidos estos datos se crean los gráficos para poder mostrarlos.

Si se producen cambios durante la consulta de esta página (como añadir un comentario nuevo) no se reflejará a no ser que se actualice la página y se vuelva a realizar la llamada a la API con los datos actualizados.

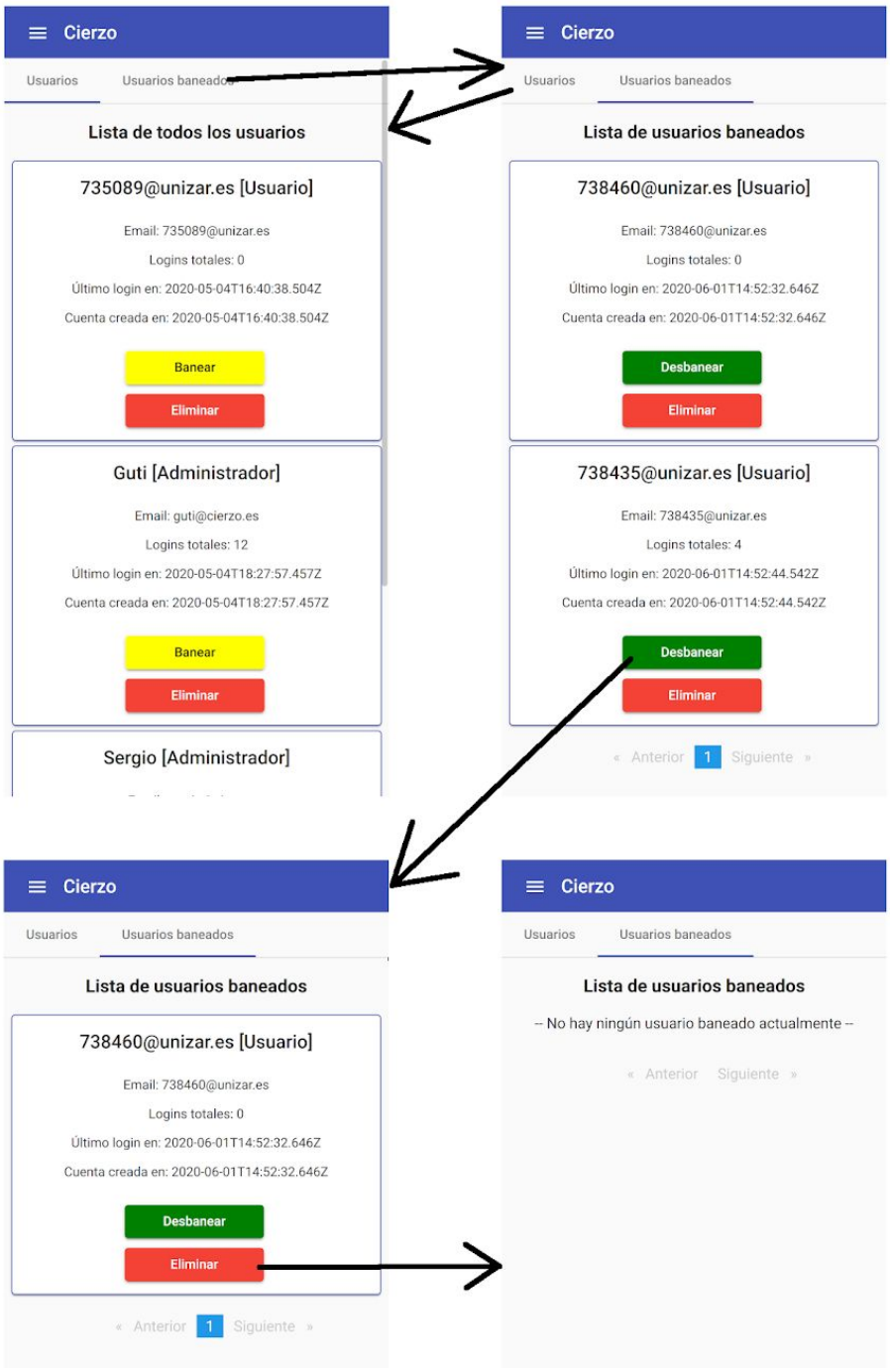
Por último encontramos la sección de gestión de usuarios. En ella el administrador podrá eliminar usuarios, banearlos y desbanearlos, además de consultar la información asociada a la cuenta. Al banear a un usuario se le incluye en la lista de usuarios baneados y al desbanear al mismo se elimina y actualiza la lista completa de usuarios. En esta pantalla solo se puede navegar entre estas dos secciones (además de todas las demás desde el menú desplegable lateral vertical).

En ambas listas se ha añadido paginación para evitar tener que hacer demasiado scroll y no sobrecargar las pantallas. La navegación es simple dado que solo se añaden y eliminan elementos de la lista y se cambian los botones (al desbanear un usuario, se actualizará en la lista de usuarios completa y se cambiará por el botón de banear).

Al acceder a esta pantalla se muestra información acerca de todas las cuentas almacenadas en el sistema gracias a una llamada GET a la API. Una vez obtenidos los usuarios se muestran en forma de listado de tarjetas.

Al banear o desbanear un usuario se lanza una petición PUT a la API para cambiar el booleano que indica si un usuario está baneado o no. Después se refresca la lista volviendo a realizar el GET y creando de nuevo ambos listados.

Al eliminar una cuenta el procedimiento es el mismo, la única diferencia es que en vez de un PUT se realiza un DELETE, eliminando la cuenta del usuario.



Analíticas

Hay diferentes analíticas implementadas en el sistema, todas ellas pensadas de forma que aporten valor adicional o faciliten la visualización de información. Además, se han diseñado gráficas exclusivas para los administradores de la aplicación que se encuentran en la sección de “Estadísticas”.

La analítica más simple sería la que indica el número de inicios de sesión de cada usuario en la pantalla del perfil del mismo. Debajo del correo electrónico hay una línea indicando el número de inicios de sesión de esa cuenta en la aplicación. Esto permite a los usuarios ver el uso que le han dado hasta ahora a la app.

Otra analítica simple sería la de veces que se ha consultado un barrio (o número de visualizaciones de un barrio). Es una métrica que va creciendo para cada barrio con cada vez que un usuario accede a ver la información que contiene. Esto permite a los usuarios ver qué barrios son los más populares y los que más se consultan. Además esta métrica sirve para un gráfico en la pantalla del administrador de estadísticas donde se muestra toda esta información recopilada en una gráfica de barras. Con esta gráfica se puede observar de manera sencilla qué barrios se visitan con más frecuencia.

La pantalla donde se concentran la mayor parte de las analíticas se trata de “Estadísticas”, disponible únicamente para administradores. En ella encontramos la gráfica mencionada anteriormente de visualizaciones por barrio, también tenemos una gráfica de barras que resume el número de valoraciones de cada barrio; esta gráfica permite observar a simple vista qué barrios son más valorados (con cuales interactúan más los usuarios) y por tanto podremos saber en qué barrios la valoración de los usuarios es más objetiva. Después hay una gráfica de barras con el número de comentarios de cada barrio. Con esta gráfica podemos ver también qué barrios son más populares o en cuales se han publicado muchos comentarios. Esto sirve para poder moderar los comentarios de barrios que tienen muchos.

Estos tres gráficos en conjunto permiten a los administradores obtener una visión global acerca de los barrios, su popularidad y la interacción de los usuarios con los mismos.

Adicionalmente en la pantalla de estadísticas tenemos una segunda pestaña que analiza las cuentas del sistema. Existen dos gráficos, el primero muestra la relación entre cuentas de administradores y cuentas de usuario normales. Esta gráfica permite tener una visión rápida de la proporción entre usuarios y administradores. Por otro lado tenemos una gráfica que muestra la relación entre el número de cuentas baneadas y el número de cuentas completamente activas. De ella se extrae información directa sobre la moderación de los administradores y del número de cuentas con comentarios no aptos para esta plataforma. Por último, debajo de ambas gráficas hay una línea de texto que indica el número de inicios de sesión globales y el total de cuentas de

usuario existentes. Todo esto aporta una visión global acerca del estado de las cuentas de usuario que pueden acceder al sistema.

Adicionalmente existen dos gráficas más en la sección de información de un barrio. En el apartado de demografía se encuentra una gráfica que muestra la distribución de población de menores de 18 años por rangos de edad. Es una manera visual de simplificar la presentación de la información dado que son porcentajes.

La otra gráfica es la de las valoraciones de los usuarios. También es una forma de simplificar la información que se quiere mostrar de una forma visualmente agradable. Se ha seguido el ejemplo que implementa amazon: Una gráfica de barras horizontal con los porcentajes de estrellas que ha recibido ese barrio. Esta gráfica permite ver de manera sencilla cuántas estrellas dan la mayoría de usuarios al barrio.

Despliegue del sistema

La aplicación web será desplegada en Heroku, es por ello que se necesitará una cuenta en la plataforma.

Desde el dashboard de Heroku se seleccionará *New* y *Create new app*, se introducirá el nombre *cierzo* y se seleccionará región *Europa*.

Para empezar el proceso de despliegue será necesario descargar todo el código fuente del siguiente repositorio de github: https://github.com/sergiocostamoreno/VirusFive.git. Además, se hará uso de Heroku CLI, su enlace de descarga es: https://devcenter.heroku.com/articles/heroku-cli.

Dentro del repositorio se encuentra el directorio *FrontCierzo*, al que se accederá por medio de una terminal. Una vez en el directorio, se ejecutará el comando *ng build --prod*. Como resultado del build, se generará el directorio *FrontCierzo/dist/FrontCierzo* con una serie de compilados. Todo el contenido de *FrontCierzo/dist/FrontCierzo* se copiará en *Backend/public*.

Llegado este punto, la aplicación está lista para subirse a la plataforma Heroku.

Desde el directorio *Backend* se iniciará sesión en Heroku con el comando *heroku login*.

A continuación, con los siguientes comandos se inicializará el repositorio con el código existente:

```
-  git init
-  heroku git:remote -a cierzo
```

Para finalizar, con los siguientes comandos se guardarán los cambios y se iniciará el proceso de subida de ficheros a la plataforma:

```
- git add .
- git commit -am "título del commit"
- git push heroku master
```

La aplicación será accesible desde https://cierzo.herokuapp.com.

Validación

Para validar el sistema se ha utilizado la tecnología de testing del frontend "*protractor*". Se han realizado 10 tests que comprueban funcionalidades básicas del sistema.

De esta manera, ejecutando el fichero de pruebas se realiza una revisión completa de las características principales de la aplicación lo que permite realizar pruebas de manera muy rápida después de algún posible cambio en la aplicación.

Las 10 funcionalidades que se comprueban son las siguientes:

1. Login exitoso (con credenciales válidas)
2. Login erróneo (con credenciales no válidas)
3. Acceder a la información de un barrio a través del mapa.
4. Añadir un comentario a un barrio y darle like al comentario
5. Quitarle like al comentario añadido en el punto anterior
6. Añadir una valoración a un barrio
7. Ordenar comentarios por likes (se añaden dos comentarios, se da like al que has introducido primero y se ordenan por likes para comprobar si va antes que el otro)
8. Cerrar sesión
9. Cargar el apartado de rankings
10. Mirar una página estática (en concreto la de las FAQ)

Estas son las pruebas que se llevan a cabo de forma automática pero durante toda la etapa del desarrollo se han realizado múltiples pruebas de otros tipos. Por ejemplo el uso de postman para testear tanto el api de la base de datos como el api del backend.

Problemas encontrados durante el desarrollo

Durante el desarrollo de la aplicación ha habido diversos problemas y cuellos de botella que han ralentizado el mismo. Aquí se explicarán varios de ellos divididos en backend y frontend.

Backend

En cuanto al modelo de datos, se ha tenido que actualizar varias veces debido a detalles pasados por alto durante el diseño. Por ejemplo se tuvo que añadir a los comentarios un array de los usuarios que le han dado like para que estos no pudieran darle más de una vez.

Otra cuestión es que la información del ayuntamiento no está actualizada, pues los últimos datos datan de 2016 aproximadamente y se emplean datos de 2013. Se estuvieron buscando otras fuentes actualizadas más recientemente, sin éxito. Además hay ciertas informaciones que no proporciona, como por ejemplo las frecuencias de las líneas de bus. Esto hace que el cálculo de puntuaciones no sea del todo justo, pues se podrían hacer algunos cálculos con las frecuencias de bus y que fueran más representativos. Además, la idea de esta aplicación era un poco más ambiciosa, pero ha sido limitada debido a la falta de datos del ayuntamiento.

Otro problema surgió cuando al hacer alguna petición al backend no se ejecutaba el método *subscribe* para alguna petición http. Esto hacía que el *observable* que se creaba, al no tener ningún suscriptor, no se ejecutaba.

También hubo un pequeño malentendido con las funciones de Mongoose y se pensaba que al hacer *save*, los datos se guardarían en la base si no estuviesen creados. Pero esto no ocurría así, por lo que la primera vez que se accedió al ayuntamiento a por los datos se tuvo que hacer un *create* y ya se hace *save* cada ejecución con *cron* de la tarea de obtención de datos. Los datos se actualizan correctamente.

Frontend

Se ha comprobado que la curva de aprendizaje de Angular es certera. Diversos problemas han ido surgiendo a lo largo del desarrollo del frontend, pero se ha conseguido un resultado satisfactorio.

Por ejemplo, al llamar funciones de otro elemento angular. Al pulsar cualquier botón de confirmación hay que llamar a la función de otro componente. Para ello no solo hay que importarlo sino que además hay que instanciarlo.

Otra cuestión del uso de los *mat-bottom-sheet*, al principio se recargaba la página entera en vez de actualizar solo un componente porque no se conocía cómo indicarle al componente padre que debía cambiar. Una vez se descubrió la propiedad para pasar la información del hijo al padre se pudo implementar mejor.

La librería Passport generó problemas también. Hubo problemas en la configuración para conseguir realizar todas las estrategias, por lo que se ha dejado únicamente aquella para el acceso con Google.

Otros detalles del front end han sido librerías que en un principio no han funcionado y se han tenido que buscar otras, como por ejemplo para usar diagramas de barras horizontales. Tampoco funcionó una librería que permite la paginación y se tuvo que buscar otra. También se encontró con la estaticidad de *apex chart*, ya que los gráficos se crean y no se pueden modificar. Por lo tanto, cada vez que un dato cambia se ha de crear otro gráfico completo. Esto no han sido problemas muy serios pero han hecho un poco más lento el desarrollo del código.

Al final durante las pruebas surgió otro problema con un *loader* que se quedaba infinitamente y se paraba la aplicación, por lo que se decidió quitar. Esto se decidió debido a que los tiempos de respuesta de Heroku no eran demasiado grandes.

Como conclusión, se puede observar que la mayoría de estos problemas han surgido por un diseño inicial que no comprendía ciertos detalles que han ido surgiendo después, pero que habrían sido evitables con un diseño un poco más elaborado, o por la falta de familiaridad con las tecnologías empleadas. Al ser la primera vez que se han usado de manera independiente muchas de las herramientas se le suma al tiempo de implementación el tiempo de aprendizaje de las mismas.

Análisis de problemas potenciales

Uno de los problemas más fáciles de aparecer puede ser que un usuario no recupere sus credenciales de inicio de sesión, Cierzo no incluye ningún sistema de recuperación o modificación de credenciales sin iniciar sesión. Además, tampoco se verifica que el correo electrónico facilitado en el registro exista; esto se podría solucionar haciendo obligatoria una confirmación por email.

Cierzo extrae datos de la api pública del ayuntamiento de Zaragoza, por lo que si esta cambia sus entradas o el formato de sus salidas la información sobre los barrios dejará de actualizarse. Debido al esquema interno de la aplicación, no se dejará de mostrar información al usuario, la información se almacena en nuestra base de datos y se actualiza semanalmente luego en el peor de los casos esta no se actualizará en X tiempo.

El despliegue contratado en Heroku es de una máquina gratuita con 512 MB RAM que pasa a estado de reposo tras 30 minutos de inactividad. No se espera un uso masivo en cuanto a número de usuarios pero, en caso de que esto sucediera, se debería aumentar el plan.

Distribución de tiempo (diagramas de Gantt, distribución de horas y tareas a cada miembro del equipo, horas totales invertidas, horas por miembro)

https://docs.google.com/spreadsheets/d/122VvABpITusiTUz5HI5JzPeCsvQ9O1j5ImB56tlsT64/edit?usp=sharing

Conclusiones

Esta aplicación nos ha permitido conocer en profundidad el stack MEAN, como coordinar unas capas con otras y los problemas más comunes que pueden surgir.

Consideramos que se trata de una tecnología cuyo aprendizaje es fundamental hoy en día en el ámbito de las páginas web. En diversas aproximaciones de empresas con ofertas de trabajo, siempre mencionan la tecnología MEAN o algunos de los lenguajes de sus capas como angular o nodejs.

Creemos que desarrollar una aplicación real es la manera más efectiva de conocer en detalle la tecnología por lo que estamos profundamente satisfecho con el aprendizaje que hemos obtenido.

Por otro lado cabe mencionar la gestión del proyecto que va más allá del aspecto informático. Esto incluye la gestión de un equipo de trabajo, el reparto de tareas según los gustos y habilidades y la gestión de posibles problemas que puedan surgir debido a disparidades de opiniones mal llevadas.

Por lo general también estamos contentos con este ámbito debido a que el equipo ha trabajado comprometido en la misma línea. Esto ha sido más fácil debido a que el grupo se conoce desde el comienzo de la carrera.

Por acabar con estas reflexiones de conclusión, cabe destacar que estamos muy satisfechos con el producto final conseguido ya que creemos que puede resultar bastante útil a un sector de la población local de Zaragoza.

Valoración personal de cada miembro

Jorge Terreu: estoy muy satisfecho con el conocimiento adquirido de angular, nodejs y mongo. Creo que realizar una aplicación real completa tiene un grado de aprendizaje difícilmente alcanzable de otro modo. He disfrutado realizando este trabajo, lo cual creo que es fundamental para un buen trabajo. Por otro lado, me gustaría felicitar a los miembros de equipo que han hecho un gran esfuerzo en este proyecto aún teniendo otras asignaturas y trabajos paralelos. En especial me gustaría destacar la labor de Alejandro Gutiérrez y Sergio Costa, que han capitaneado el backend y frontend respectivamente. ¡Buen trabajo boys!

Sergio Costa: me atrevería a decir que es la asignatura que más he aprovechado y disfrutado este cuatrimestre. Han sido horas y horas aprendiendo conceptos y técnicas que, de verdad, creo que son útiles para lo que quiero y aspiro a hacer nada más tener el título bajo el brazo. Me gustaría destacar el compromiso que hemos tenido como grupo semana a semana reuniéndonos todos los lunes a las 16:00, y la brillante idea que puso en marcha la maquinaria, gracias J contigo empezó todo.

Juan Vallejo: estoy muy satisfecho con esta asignatura por varios motivos. En primer lugar se nos ha enseñado con las prácticas el uso de una serie de tecnologías muy útiles para el desarrollo de aplicaciones web. Gracias a estos conocimientos hemos podido aventurarnos en crear una aplicación web utilizando el stack MEAN que hemos aprendido e integrando otras tecnologías para hacer la aplicación más completa y diversa. Por todo ello y el gran trabajo del equipo VirusFive estoy contento con los conocimientos adquiridos y la organización y buen rollo que ha llevado el equipo.

Daniel Revillo: La verdad es que sin esta asignatura no habría salido de la carrera sabiendo cómo programar aplicaciones. Es en esta asignatura en la que de verdad se aprende a programar, la estructura y la implementación real de una aplicación desde cero. Gracias a ella se amplía nuestra visión y nos permite probar aquellos aspectos que no teníamos tan claros, además de emplear tecnologías muy usadas actualmente. Por último el hecho de que se haga en grupos permite fomentar ese trabajo en equipo que puede costar a muchos informáticos. ¡Ahora a celebrarlo!

Alejandro Gutiérrez: me ha gustado bastante y me ha parecido una asignatura muy interesante ya que es de las pocas que enseñan aspectos desde el punto de vista práctico y gracias a esto ha sido mucho más llevadera. Muy bien estructurada cronológicamente lo que ha hecho posible aprender cómo desarrollar una aplicación web desde cero. También, resaltar el gran trabajo de mis compañeros que después de tantas horas de programación se merecen una jarrica bien fría de cerveza.

Bibliografía:

[1] https://editor.swagger.io/

[2] https://www.npmjs.com/package/winston

[3] https://www.npmjs.com/package/jsonwebtoken

[4] https://www.npmjs.com/package/cron