MeetGrid

Documentación de proyecto

ÍNDICE

1. Definición del proyecto

1.1. Obtención de la información necesaria para el proyecto

Hablando con conocidos y sus experiencias en redes sociales para conocer pareja ponemos en común los fallos y dificultades de cada una. MeetGrid nace como una respuesta a todo esto.
Se encuesta a distintas personas usuario objetivo, enfocándome en distintos géneros y orientaciones sexuales para conocer sus necesidades e inquietudes, haciendo una aplicación que responda a todas.

Una vez madurada la idea, se recabó documentación del repositorio GitHub de Manuel Conde destinado a ayudar a estructurar el proyecto, así como su consejo para desarrollar la base de datos y la estructura en la que se basaría posteriormente el acceso a datos.

1.2. Descripción detallada del proyecto

El objetivo del proyecto de charlar o acordar una cita con otras personas de la manera más sencilla, privada, menos intrusiva y cómoda posible.

Esta aplicación permitirá registrarse a los usuarios indicando su email (key candidata), nombre de usuario (puede haber repetidos), contraseña, edad, género (opciones en desplegable), localidad (opciones en desplegable), foto de perfil y un campo de texto para que el usuario escriba libremente sobre sí mismo (gustos, hobbies, qué busca en la aplicación...). También podrán modificar su usuario una vez creado.

Al logarse llegaremos a la pantalla central de la aplicación, una rejilla de perfiles. Habrá un filtro de búsqueda para afinar los resultados según edad, zona y género.

Se podrá enviar mensajes a personas que aparezcan en tus vistas, bloquear a otros usuarios libremente y podrán enviarte mensajes, reportar usuarios al superusuario y tener usuarios favoritos con su propia vista exenta de filtros.

1.3. Ámbito del proyecto

Se ha de entregar una aplicación web, que en nuestro caso se puede catalogar tanto como red social como aplicación para “ligar”.

1.4. Adaptación o creación

Se adaptan y aúnan los apartados de diseño y funcionalidad que los usuarios encuestados creen mejores de distintas aplicaciones de citas. Algunas de las aplicaciones de las que se toma inspiración son “Tinder” o “Grindr”.

1.5. Conceptos básicos del proyecto software

¿Qué va a hacer la aplicación?

El eje principal son la visualización de otros usuarios y los mensajes que se envían entre ellos. Previamente podrás ver un perfil que cada persona podrá rellenar a su gusto y modificar libremente.

¿Cuál es su atractivo principal?

Se diferencia de otras aplicaciones en tener una interfaz más amigable, cuyo fin no sea monetizar y cosificar a otros usuarios, sino su comodidad y libertad.

¿Qué problema concreto va a resolver?

Mediante el grid principal da el control total al usuario, evitando vistas en formas de tarjeta y derivados. Las aplicaciones similares hoy en dia monetizan y cosifican a sus usuarios, MeetGrid evitará eso.

¿Qué necesidad va a cubrir?

El ser humano es un animal social, necesitamos contacto con otras personas y nutrirnos de lo que nos aporten, lo cual en estos tiempos por la emergencia sanitaria es muy complicado. Desde la comodidad de tu hogar, puedes conocer y mantener contacto con otras personas con MeetGrid.

2. Planificación del proyecto web

2.1. Definición de tareas

1. Concepto inicial: Nacimiento de la idea e idear cómo será el proyecto y si es viable y adecuado.
2. Mapa de la web: Teniendo claras las tareas a realizar y cómo se comportará la web, se realiza mapa web que las aúne todas y tenga sentido.
3. Diseño de la base de datos: En base a los requisitos y las tareas que realizan las vistas en el mapa web, se diseña una base de datos que contemple todos los registros, reglas y tipos de datos necesarios.
4. Elección de lenguajes de programación: En base los requisitos de funcionalidad y escalabilidad se eligen unos lenguajes de programación
5. Creación de tablas y registros en la base de datos: Usando el diseño previamente creado de la base de datos y teniendo clara la tecnología a utilizar para hacerlo se crea la base de datos con los registros mínimos necesarios para hacer funcional la página según la lógica planificada.
6. Codificación de los modelos, objetos de acceso a datos y servlets Java: creamos el código acorde a la lógica diseñada previamente.
7. Codificación del interfaz web: Se realiza codificación de parte del cliente que reflejará el resultado de la lógica, así como otros efectos, siendo agradable visualmente e intuitiva.
8. Adaptación responsive mediante Bootstrap 4: Una vez hecha la vista de escritorio, se adapta a tamaño tableta y móvil.
9. Control de formularios mediante JavaScript: se introduce la lógica y regex a nivel de cliente para controlar los datos introducidos por el usuario.
10. Efectos y transiciones JavaScript: se añaden efectos no intrusivos para hacer la aplicación web más llamativa.
11. Investigación de elementos innovadores no vistos en clase: uso de APIs para subida de imágenes a servidor externo e introducción del usuario contenido richtext controlando sus posibilidades.
12. Encriptación de contraseñas: encriptado de contraseñas al registrarse en nuestra base de datos, desencriptación de las mismas para hacer lecturas.
13. Revisión y búsqueda de errores: se prueban todas las funcionalidades buscando algún error que haya pasado por alto.
14. Creación de documentación final y entrega: Documentar proyecto y proponer tanto código como documentación en el repositorio GitHub así como aplicación virtualizada en Proxmox para evaluación.

2.2. Definición de tiempos

Al realizarse este documento en las fases finales del proyecto, se documenta los tiempos en los que se han afrontado las tareas con carácter póstumo para acercarlo a la realidad lo máximo posible.

Tiempos	Tareas
1) Diciembre 2019	Surgir y maduración de la idea.
2) Enero 2020	Estudio de mercado.
3) Febrero 2020	Primeros diseños. Se crea las primeras versiones de la base de datos, mapa de la aplicación y diseños de la interfaz sobre papel.
4) Marzo 2020	Primer diseño de interfaz no funcional en HTML y primeros documentos detallando el diseño.
5) Abril 2020	Creación de diseños y documentación adicional.
6) Mayo-Septiembre 2020	Parón del proyecto por la COVID-19. Ante la incertidumbre y aplazamiento de fechas se paraliza el proyecto.
7) 22 de Septiembre 2020	Se retoma el proyecto, ya con fecha de deadline. Se inicia codificación, subida de la versión 0.1 en GitHub, solo contiene un login y register.
8) Final de Septiembre 2020	Diseño y scripts de base de datos completados. Vista de grid de perfiles y filtros completada. La aplicación girará en torno a ella como estaba previsto. Se intenta hacer una versión Hibernate, pero por especificidad y control de la base de datos se desecha.
9) Octubre 2020	Creación del resto de Servlets y vistas. Diseño responsive, controles de formularios y efectos JavaScript.
10) Noviembre 2020	Se introduce subida de imágenes a Google Firebase, campos richtext y encriptado de contraseñas. Se completa documentación y se pulen detalles. Se virtualiza el proyecto en VM Ubuntu. Versión 0.6.4 candidata a ser 1.0 previa revisión por parte del centro.
11) Diciembre 2020	Finalización y entrega.

3. Análisis del proyecto web

3.1. Requisitos técnicos web

Para su creación:

- Para crear el código de la parte lógica un entorno de desarrollo: Eclipse.
- Para el diseño de la base de datos, motor de base de datos y su interfaz gráfico: MySQL Workbench.
- Para el diseño de la interfaz, previo portarlo a jsp sobre Eclipse, entorno de desarrollo específico para HTML y CSS: Visual Studio Code.

Para su despliegue:

- Como contenedor de aplicaciones y servidor: Tomcat 9 + Catalina.
- Como motor de base de datos: MySQL.

Para su uso:

- Navegador web. Los requisitos de hardware dependen del elegido. Para pruebas se han usado Chrome y Firefox.

3.2. Elementos del contenido

Nuestra aplicación web está formada por 3 partes principales y varias subpartes:

1. Lógica: modelos Java, objetos de acceso a datos, Servlets Java, librerias Maven.

2. Cliente: jsp que almacenan código html con Java embebido para interacturar con los Servlets, Código CSS tanto custom como clases Bootstrap 4, control de formularios JavaScript, efectos Javascript, subida de imágenes a servidor externo en lado cliente con Google Firebase.

3. Persistencia: base de datos diseñada mediante scripts SQL por motivos de escalabilidad y necesidades del diseño.

3.3. Diseño gráfico

Tras un estudio de las tendencias más populares en diseño gráfico de aplicaciones actuales, me decanto por colores planos, iconos minimalistas y diseño material.

El logo se realiza con dos símbolos de mensajes y un corazón, quedando 3 colores que serán la base posterior de la interfaz: negro (los mensajes) , rojo (el corazón) y blanco (el espacio vacío).

Se escoje como color para reslatar el rojo por el significado de los colores en diseño gráfico: se le relaciona con el amor y la pasión.

3.4. Herramientas para web

Al hacerse el proyecto íntegramente en local y en solitario, la única herramienta para web utilizada ha sido “GitHub Desktop” para actualizar el repositorio alojado en GitHub.

Ésta permite realizar control de versiones, así como almacenar en un servidor remoto nuestro proyecto.

4. Ingeniería del proyecto web

4.1. Diseño del software

Todo el software, tanto las clases Java, los Servlets como posteriormente el contenido de los jsp es realizado en base al mapa del sitio para conocer los requisitos de cada vista y así identificar que lógica se ha de crear para que estas funcionen, basandome en la estructura CRUD de los Servlets Java.

Login

•Busca coincidencia de usuario registrado en la BD para dejarlo entrar en la aplicación.

•Si es user lleva a PERFILES, si es admin a GESTIONAR INICIDENCIAS, si no está registrado vuelve a LOGIN.

•READ de USER.

•Guarda en sesión ROLE, ID. (doPost)

Registro

•Crea nuevo usuario en la BD.

•CREATE en USER. (doPost)

ROL USER

PERFILES

•Muestra los perfiles aplicando los filtros que el usuario elija.

•Al pulsar sobre algún perfil permite ir a la pantalla MENSAJE.

•Tambien permite poner como favoritos, bloquear y reportar. Para ésto cada perfil tendrá 3 enlaces debajo con estas opciones. Se tomará el ID en session (para reportes) al pulsar sobre ellos y se hará favorito/bloqueado o bien se llevará a la página de reportes.

•UPDATE de FAVORITE / BLOCK. (doPost)

•READ de USER. (doGet)

PERFIL

•Muestra el perfil del usuario sobre el que hemos pulsado.

•Al pulsar sobre ella lleva a la pantalla PERFIL.

•READ de USER. (doGet)

•Contiene botones para ver las vistas de MENSAJES, BLOQUEAR, REPORTAR Y FAVORITO pasándole la ID del usuario visitado.

MENSAJES

•Permite ver mensajes previos con el id del usuario sobre el que hemos pulsado para acceder a ella.

•Permite crear nuevo mensaje para este usuario.

•READ de MESSAGE. (doGet)

•CREATE en MESSAGE. (doPost)

FAVORITOS

•Visualiza nuestros usuarios favoritos y lleva a la pantalla MENSAJE al pulsar sobre ellos.

•Permite quitar de lista de favoritos.

•READ de FAVORITE. (doGet)

•DELETE de FAVORITE. (doPost)

REPORTAR

•Permite reportar un usuario por conducta indebida con un mensaje a los admin.

•CREATE en REPORT. (doPost)

MODIFICAR PERFIL (USER)

•Visualizamos todos nuestros datos.

•Permite modificar nuestros datos de perfil, como la edad o la zona geografica.

•READ de USER. (doGet)

•UPDATE de USER. (doPost)

•Permite borrar el usuario logado de la BD.

•DELETE de USER. (doPost)

MENSAJES RECIENTES

•Visualizamos mensajes de todos los usuarios remitentes a nuestro perfil en las últimas 48h

•READ de MESSAGE (doGet)

ROL ADMIN

REPORTES

•Lee los usuarios reportados por otros, junto con el mensaje del reporte. Permite borrar los usuarios reportados directamente o borrar el reporte en sí.

•READ de REPORT. (doGet)

•DELETE de REPORT (doPost)

•Permite borrar un usuario de nuestra elección de nuestra aplicación (es distinta de DARSE DE BAJA en que no borra nuestro ID, sino uno de nuestra elección)

•READ de USER (doGet)

•DELETE de USER (doPost)

MODIFICAR PERFIL (ADMIN)

•Accede a los perfiles reportados y permite modificarlos acorde al reporte.

•READ de USER. (doGet)

•UPDATE en USER. (doPost)

4.2. Diseño de datos

La persistencia de nuestros datos se basa en la estructura SQL con las siguientes tablas y atributos:

USER (**id**, email (clave candidata), password, name, age, gender, area, pic, description)

ROLE (**id**, type)

MESSAGE (**id**, sender, receiver, content, pic, date)

FAVORITE (**id**, owner, favorited)

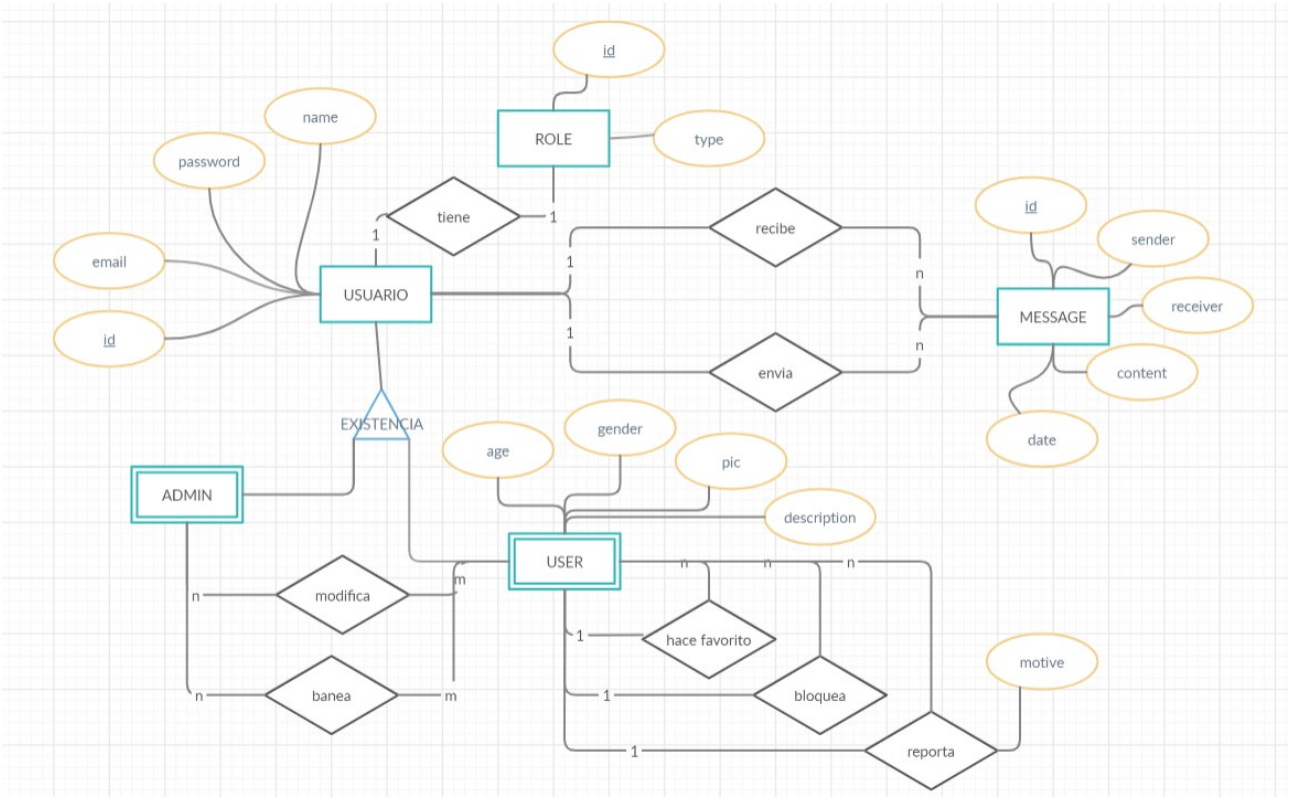
BLOCK (**id**, owner, blocked)

REPORT (**id**, owner, reported, motive)

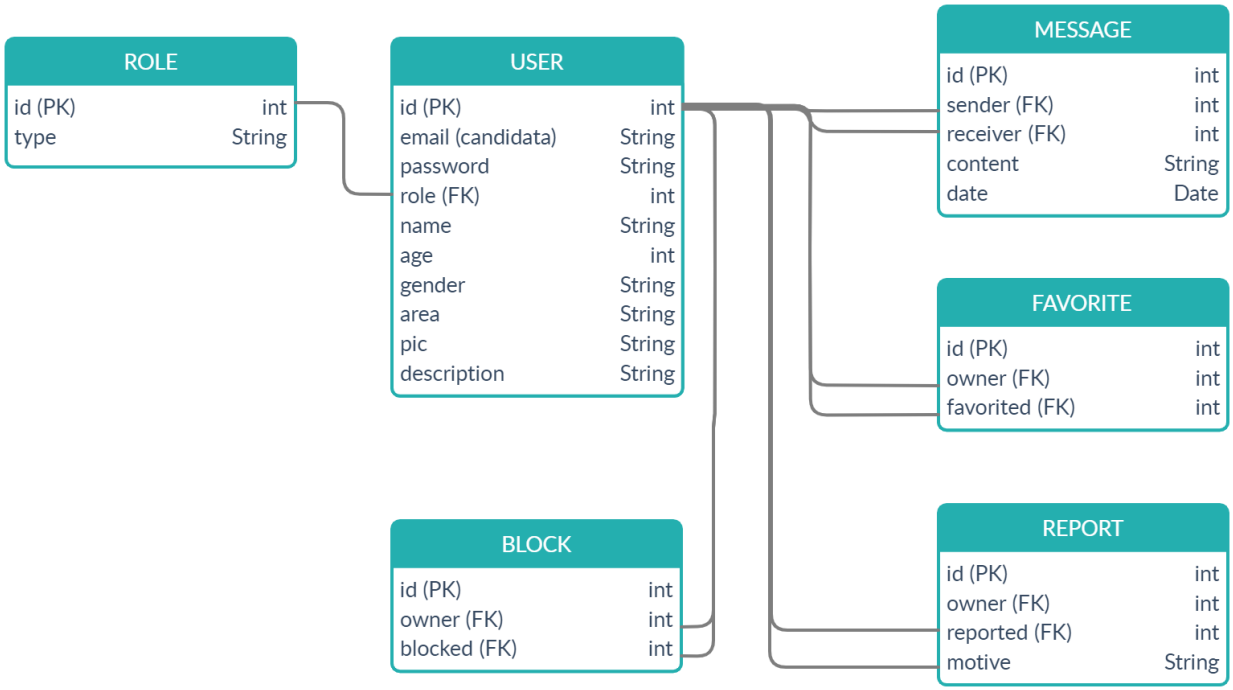
Las claves están marcadas en negrita.

Todas las tablas tienen una id long autogenerada.

Esquema entidad-relación:



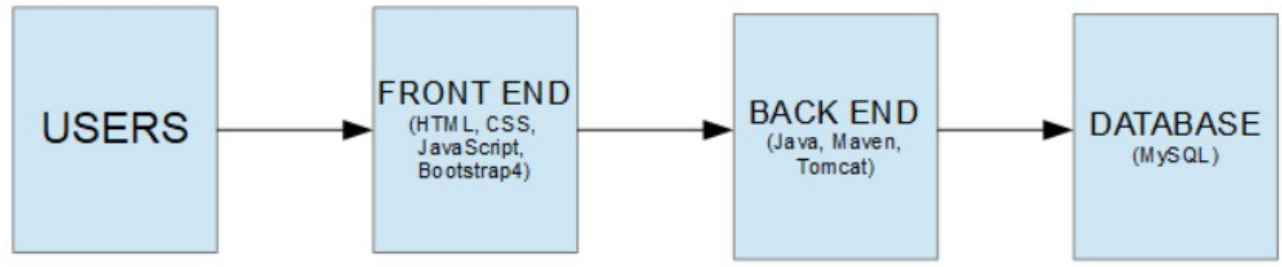
Esquema relacional:



4.3. Diseño del sistema

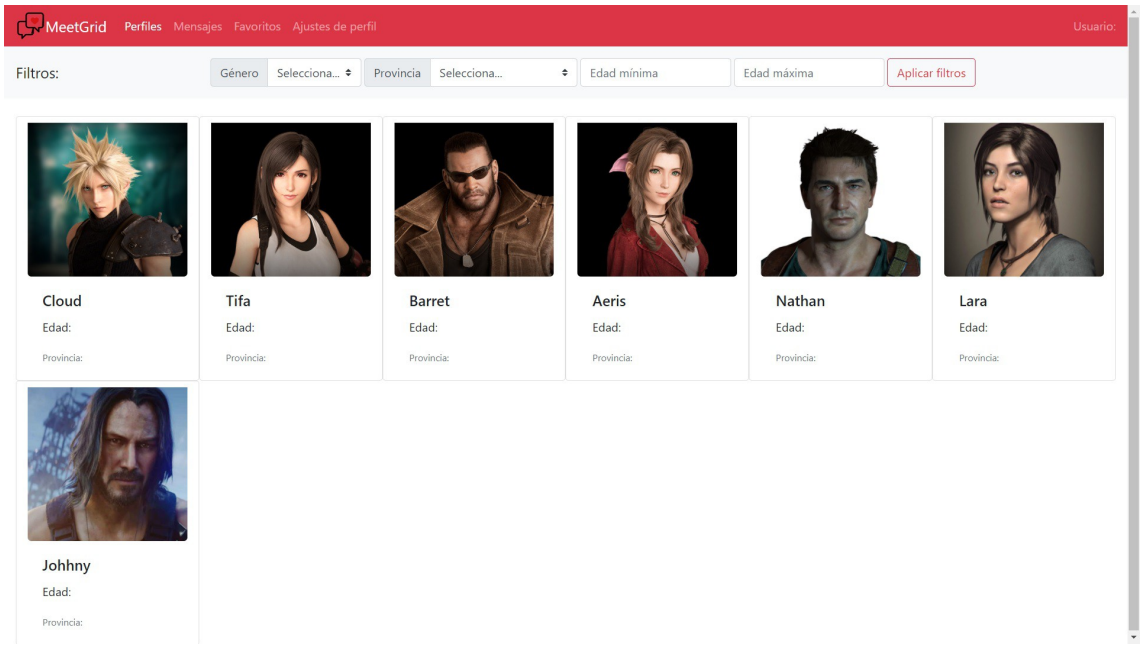
Mediante llamadas a ellos desde la interfaz, los servlets llaman a métodos de objetos de acceso a datos que mediante modelos Java atacarán la base de datos y obtendrán la información deseada para plasmarla en la vista y/o redirigir a otras vistas.

La tecnología que usa el sistema para ello es la siguiente, plasmada en el stack tecnológico:



4.4. Diseño de la interfaz del usuario

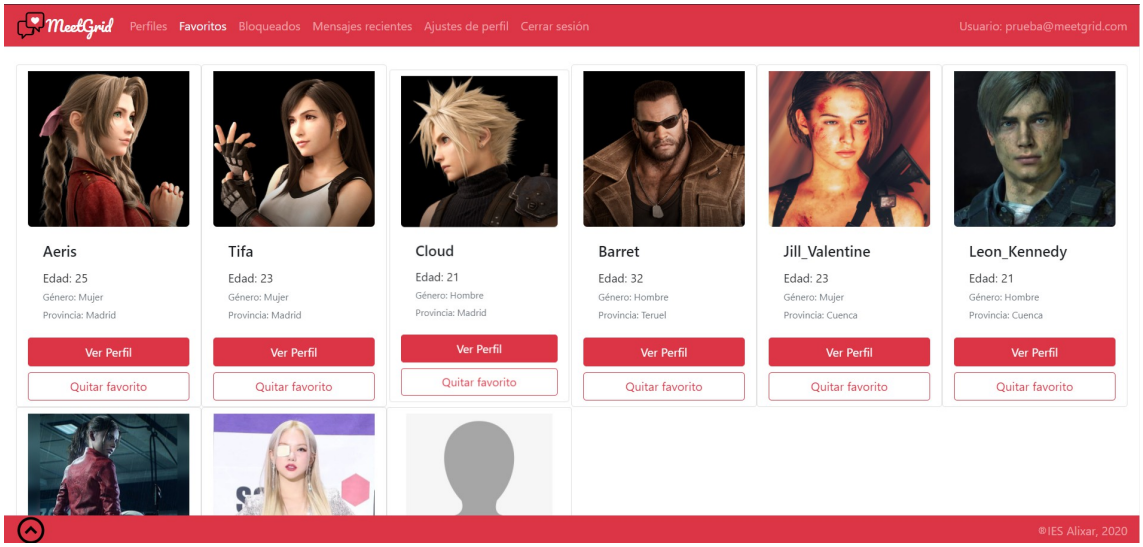
Basado en el diseño gráfico indicado en el punto 3.3 de este manual, hice primeramente un primer diseño en un HTML, haciendo uso solo de clases Bootstrap 4 y css custom para acomodar las imágenes.
Este diseño fué de la rejilla de perfiles, el centro de la aplicación y lo que le da nombre:



Usé este primer diseño como base para construir el resto de la página, usando la misma barra header+nav, colores y las cards de Bootstrap 4 para cargas de listas de perfiles.

También a través de él decidí que cada recuadro tendría solo una fina linea gris para delimitarla, buscando una interfaz clara y limpia.

Este es el resultado final basado en este prototipo:



Teniendo clara la estructura, colores y demás, pasé a sopesar viendo tanto webs como aplicaciones móviles que partes añadir, quitar o modificar para adherirme a las convenciones actuales y llegué a las siguientes conclusiones:

-Las aplicaciones móviles no suelen tener footer, por lo que mi footer sólo se muestra en la versión escritorio. El usuario móvil ve mas intuitivo un solo deslizar de dedo para volver al principio que localizar y pulsar un botón como haría en un escritorio con un ratón.

-El header+nav será desplegable en la versión movil y tablet, también para adecuarme a convenciones y por motivos de espacio.

-Header+nav y footer serán sticky para comodidad del usuario y rápida navegación por la aplicación. Esto hace que haya que calcular los márgenes top y bottom con Bootstrap 4 y diseñarlo de tal manera que no pise ninguna información.

-El usuario quiere esperas 0. Se sopesó una animación de carga pero se retiró para ganar en inmediatez. Solo el login tiene un efecto fade-in muy corto. En su lugar y para denotar el cambio de vista y finalización de carga, el icono de la aplicación vibra cada vez que carga una vista. Así consigo la alerta visual esperada, inmediatez y nula intrusividad.

-La altura de los formularios es a propósito. Está diseñado para que la animación que lleva a los errores sea evidente e intuitivo a que altura queda el error.

-Los botones serán estándar y rojos siempre que tengan funciones de navegación por el sitio. En el momento que haya alguno que, por ejemplo, borre registros, se diferenciará con un estilo outline.

4.5. Diseño del contenido

Al contribuir los mismos usuario todo el contenido (textos e imágenes) mediante sus perfiles y mensajes, y el CSS e iconos adaptarse a los ya creados en Bootstrap 4, sólo se ha necesitado elegir imágenes para el carrusel publicitario del login y la creación del logo.

5. Identificación y cuantificación de contingencias

A continuación se detallan las posibles contingencias:

1. Caída del servicio Google Firebase: MeetGrid no tendría capacidad de guardado de imágenes al depender de una API y servidor externos.

2. Incidencia con servidor Bootstrap: al depender de código externo, si no es posible acceder a él los estilos de la aplicación no cargarían correctamente.

3. Nueva versión de Bootstrap: nuestra aplicación podría quedar obsoleta y tener un menor soporte por parte de navegadores.

4. Actualizaciones de navegadores: Se ha realizado pruebas sobre versiones Chrome y Firefox actuales, en próximas versiones podría no funcionar correctamente y necesitar una actualización.

5. Soporte Javascript: si se usa en un navegador sin soporte no funcionarían ni los controles por regex de introducción de datos, ni la subida de imágenes, ni el editor de texto ni las animaciones.

6. Fallo en la máquina host al correr Tomcat o MySQL: Es necesario asegurarnos que la máquina corre ambos servicios y si no lo hace, iniciarlos manualmente.

6. Aseguramiento de calidad

Se han realizado casos de prueba, con diferentes entradas y salidas, localizando una gran cantidad de errores durante la elaboración de este proyecto.

Estas pruebas también han permitido la creación tanto de vistas nuevas como comprobaciones y restricciones nuevos.

Por ejemplo, el campo imagen al crear un usuario no es obligatorio, y al crear un usuario sin foto me di cuenta que ese campo a veces mostraba un icono genérico de error ante un valor null.
De ahí nació la imagen placeholder que ahora se muestra mediante una sencilla evaluación en los JSP que la llama al tener un valor vacío o null la imágen del usuario a mostrar.

También se hicieron pruebas similares con valores distintos seleccionados e intorducidos en los filtros, en el registro, envío de mensajes, campos richtext con el editor enriquecido de texto y otros.

Otras pruebas que se hicieron son de navegación. Mediante pulsado de botones y el nav, hacia todos los sentidos posibles, verificando que todas las vistas carguen con los datos correctos y se pisen los datos correctos en sesión al hacerlo.

Se han probado también el correcto visualizado y navegación en todas las vistas para todos los viewports usando los ppi universalmente aceptados y coincidiendo con los de bootstrap.

Otra prueba es la de añadido a la misma vista. Se han hecho pruebas en las que al añadir un dato nuevo o modificarlo en la misma vista que se muestra, sin recargar se muestre automáticamente siempre. Esto es así tanto por ejemplo en la modificación de perfiles como envío de mensajes.

Se ha hecho prueba remota de uso con una persona nivel informático “usuario” para ver si era intuitiva la navegación y provocaba algún paso un posible error y también fue satisfactoria.

Mediante todo esto, el estado actual de la aplicación es que no muestra errores ni visuales, ni en funcionamiento.