

# Trabajo fin de grado

Desarrollo de Aplicaciones Web

2020/2022

Instituto tecnológico

edix

Ignacio Viseras Riego - Rodrigo Sendino Sanz

# Índice

1	<b>Introducción</b> Introducción [EN] 1-2
2	<b>Objetivos del proyecto</b> Situación actual Sistemas actuales 3-7
3	<b>Módulos Formativos</b> <b>Aplicados en el TFG</b> 8-9
4	<b>Herramientas/Lenguajes</b> <b>Utilizados</b> 10-15
5	<b>Fases del Proyecto</b> 16-36
6	<b>Conclusiones</b> Visuales 37-44
7	<b>Agradecimientos</b> 45
8	<b>Anexos</b> 45
9	<b>Bibliografía</b> 46

# 1 Introducción

En este Trabajo Fin de Grado para el Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Web, exponemos una aplicación web basada en la gestión servicios sanitarios en el ámbito de uso de medico/cliente.

Se ha elegido esta idea debido a la incipiente cantidad de servicios sanitarios online a consecuencia de esta pasada pandemia, en la cual, nos hemos podido percatar de una atención médica de forma telemática es algo necesario y ha llegado para quedarse y puede abrir nuevos campos e integraciones en la rama del desarrollo web que podría mejorar sustancialmente el uso del cualquier sistema sanitario, generando una sensación mucho más positiva y cercana para el usuario.

Esta aplicación web está enfocada para uso médico, ya que cuenta con diferentes campos e información almacenada del usuario que podrá consultar y actualizar los datos siempre que quiera, de estos datos se almacenarán sólo los más vitales, por lo que hace que la aplicación sea más ligera y rápida.

Con esta página web se trata de crear un sistema de gestión médica ágil y sencillo que facilite las gestiones tanto de los pacientes como de los médicos, generando un historial clínico que se puede consultar en cualquier momento, la capacidad de pedir citas y visualizar las mismas en un calendario, ver las medicinas asignadas y los datos de diagnóstico y físicos de forma individual y con tu médico, también se ha incorporado un sistema de mensajes con el cual puedes realizar una consulta a tu médico o poder enviar mensajes a otros pacientes, en definitiva crear una red sanitaria en la que puedas acceder de una forma clara y accesible a la información que necesites en el momento que necesites.

En conclusión, esta aplicación expone la información de manera que los pacientes puedan acceder fácilmente a ella y puedan interactuar para realizar consultas con su médico y realizar las funciones de citación médica y puedan visualizar las mismas de forma clara en su navegador, y puedan acceder hasta una descarga del pdf de la misma citación, esta aplicación pone toda la información a servicio del consultante para que pueda tener toda la información necesaria sobre sus citas médicas, medicinas, diagnósticos, sin que tenga que realizar ningún procedimiento dificultoso para el usuario.

# 1 Introduction

In this Final Degree Project for the Higher Degree Training Cycle in Web Application Development, we expose a web application based on the management of health services in the field of doctor/client use.

This idea has been chosen due to the incipient number of online health services as a result of this past pandemic, in which, we have been able to realize that telematic medical care is something necessary and has come to stay and can open new fields and integrations in the branch of web development that could substantially improve the use of any health system, showing a much more positive and close feeling for the user.

This web application is focused on medical use, since it has different fields and saved information of the user that can consult and update the data whenever he wants, of these data only the most vital will be stored, which makes the application more light and fast.

With this web page, it is about creating an agile and simple medical management system that facilitates the management of both patients and doctors, discovering a clinical history that can be consulted at any time, the ability to request appointments and view them. in a calendar, see the exposed medicines and the diagnostic and physical data individually and with your doctor, a message system has also been incorporated with which you can consult your doctor or be able to send messages to other patients, in definitively create a health network in which you can access in a clear and accessible way the information you need at the time you need.

In conclusion, this application exposes the information in a way that patients can easily access it and can interact to make consultations with their doctor and perform the functions of medical appointment and they can see the same clear forms in their browser, and they can access up to a download the pdf of the same citation, this application puts all the information at the service of the consultant so that he can have all the necessary information about his medical appointments, medications, diagnoses, without having to carry out any difficult procedure for the user.

## 2 Objetivos del proyecto

### Situación actual

Para diversificarnos y destacarnos en el mercado nos hemos fijado 3 factores diferenciadores.

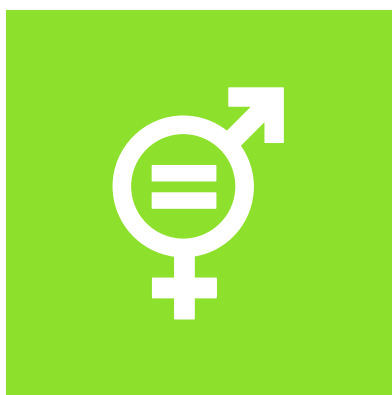
En esta sección mostramos los objetivos prioritarios y como los hemos reflejado nuestra propia estrategia.



#### Cercanía

##### **Conseguir un trato directo y ágil**

Al tener la información accesible siempre que se quiera consultar creamos un trato sincero y directo



#### Accesibilidad

##### **Criterios de diseño y opciones de usuario**

Abordando temas de accesibilidad utilizamos un diseño sencillo y con opciones inclusivas para todos.



#### Interactividad

##### **Chat interno**

Se incorporan un chat donde los usuarios pueden interactuar entre si

# Sistemas actuales

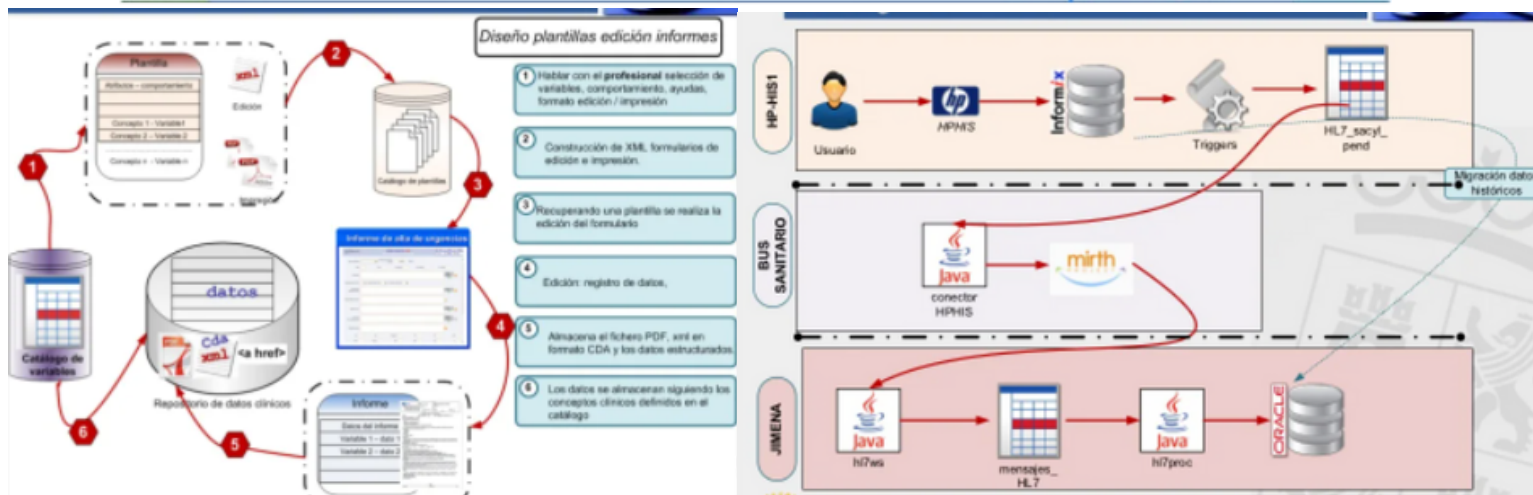
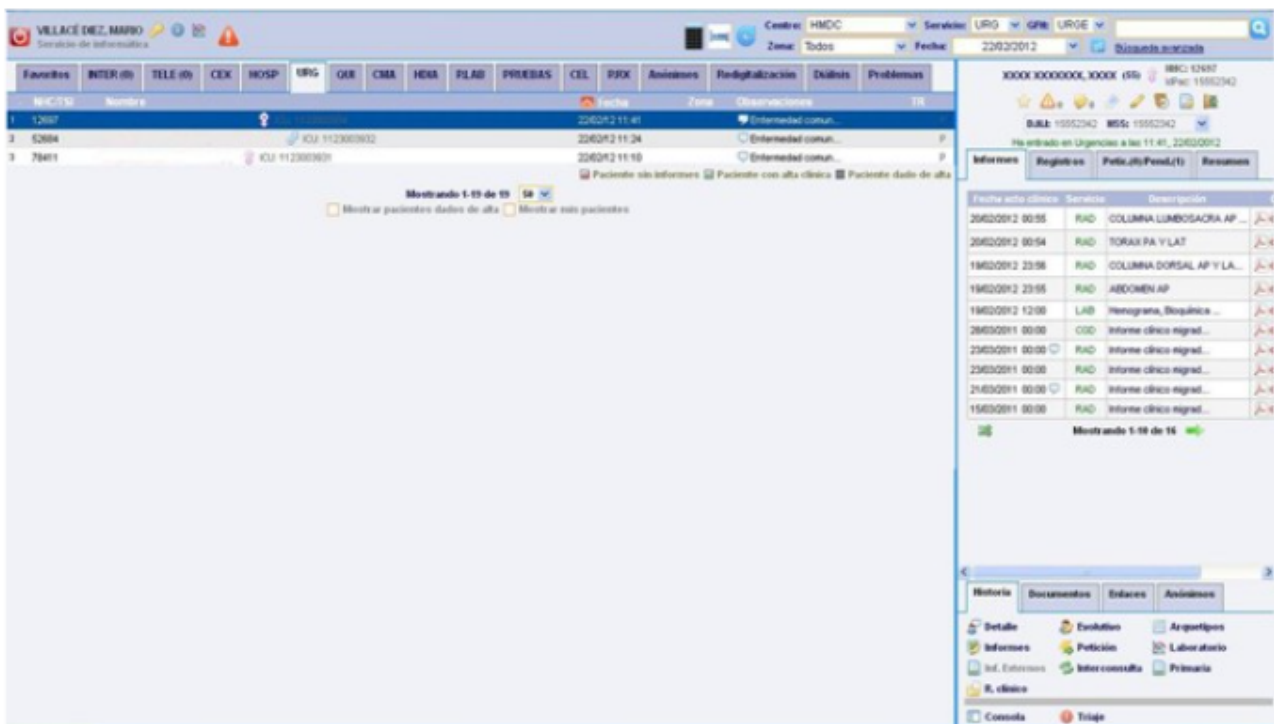
Con el fin de realizar un estudio de la competencia y comprobar los diferentes proyectos en curso de los diferentes sistemas de gestión sanitaria hemos realizado esta tabla:

Nombre	Descripción	Datos interesantes
--------	-------------	--------------------

## Jimena

Servicio de Informático de Hospital en Castilla y Leon

- Buena aceptación de los médicos
- Telemedicina



# Sistemas actuales

Con el fin de realizar un estudio de la competencia y comprobar los diferentes proyectos en curso de los diferentes sistemas de gestión sanitaria hemos realizado esta tabla:

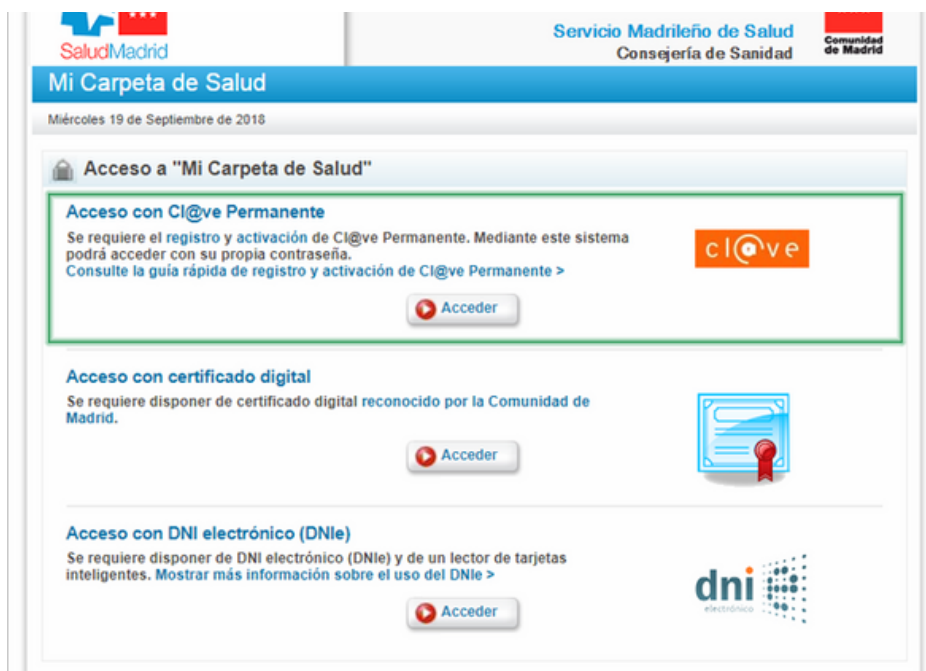
Nombre	Descripción	Datos interesantes
SACYL conecta	Aplicación de gestión de citas en Castilla y Leon	<ul style="list-style-type: none"> <li>• Posibilidad de revisar y descargar documentos</li> <li>• Búsqueda de centros cercanos</li> </ul>



# Sistemas actuales

Con el fin de realizar un estudio de la competencia y comprobar los diferentes proyectos en curso de los diferentes sistemas de gestión sanitaria hemos realizado esta tabla:

Nombre	Descripción	Datos interesantes
<b>Mi Carpeta de Salud</b>	Consulta su información clínica, citas sanitarias y otros servicios en Madrid	<ul style="list-style-type: none"> <li>Acceso mediante cl@ve, dni electrónico...</li> </ul>





# Sistemas actuales

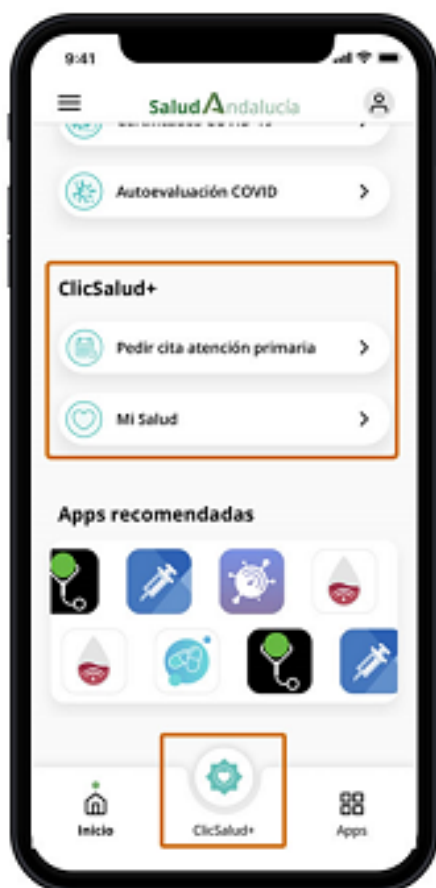
Con el fin de realizar un estudio de la competencia y comprobar los diferentes proyectos en curso de los diferentes sistemas de gestión sanitaria hemos realizado esta tabla:

Nombre	Descripción	Datos interesantes
--------	-------------	--------------------

## Clic Salud+

Consulta su información clínica, citas sanitarias y otros servicios en Andalucía

- Consulta de datos personales



# 3 Módulos Formativos Aplicados en el TFG

Durante el desarrollo del proyecto, hemos utilizado diferentes los módulos del grado los cuales son:

## Programación y Desarrollo Web Entorno Servidor

En el **Back-End** de la aplicación se ha utilizado el lenguaje de programación **Java** utilizando el paradigma de programación orientada a objetos, esta parte se encarga de darle todo el dinamismo de la pagina y controla su lógica.

## Lenguajes de Marcas y Desarrollo Web Entorno Cliente

Hemos utilizado esta asignatura para la parte **Front-End** aplicando el **HTML** (estructura), **CSS** (estilos) y **JavaScript** (Peticiones a apis, alertas). Javascript también se ha utilizado en el formulario de registro para validar la información introducida.

## Bases de Datos

En nuestra aplicación hemos creado un esquema de base de datos que consta de 10 tablas, con diferentes relaciones, las tablas son:

- usuario
- usuario\_perfil
- perfil
- diagnostico
- información
- cita
- historial clinico
- medicina
- tipo
- comentarios

Más adelante exploraremos mas afondo estas tablas.

## Entornos de Desarrollo

Con el fin de realizar un versionado de la aplicación se ha utilizado git el cual es un software de control de versiones diseñado por Linus Torvalds y también hemos realizado la plataforma GitHub creando el repositorio de este mismo proyecto.

## Despliegue de Aplicaciones Web

Hemos utilizado este modulo para realizar un resumen/tutorial de como realizar el despliegue de esta aplicación.

## Diseño de Interfaces Web

En este modulo se basa todo el sistema de estilo y esquema visual de la pagina, creando la guía visual y el wireframe de la aplicación junto a su modelo totalmente responsive.

También hemos trabajado las asignaturas de **Ingles** traduciendo la introducción de este trabajo e **Empresa e Iniciativa Emprendedora** en los puntos anteriores de Situación actual y Sistemas actuales

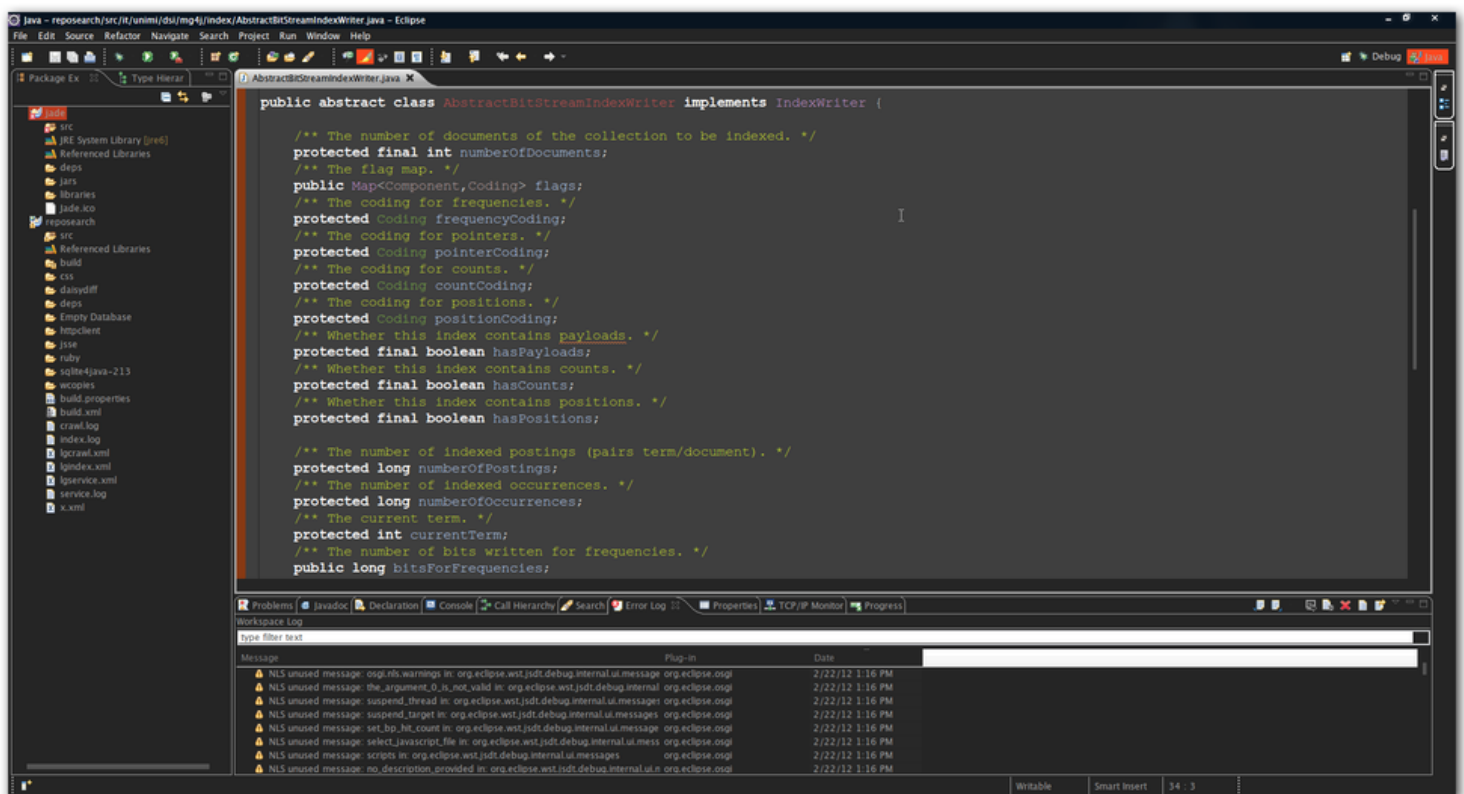
# 4 Herramientas Utilizadas

A continuación vamos a mostrar las herramientas y lenguajes que hemos utilizado en esta aplicación.

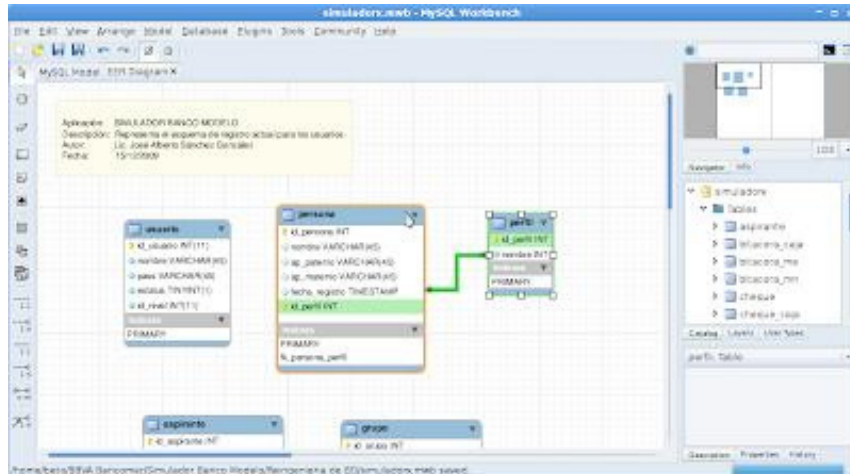
## Eclipse Java EE



Eclipse es un entorno de desarrollo software multi-lenguaje construido alrededor de un workspace al que pueden incluirse un gran número de plug-ins que proporcionan funcionalidades concretas relacionadas con lenguajes específicos o con la interacción con otras herramientas implicadas en el desarrollo de una aplicación.



MySQL Workbench es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, gestión y mantenimiento para el sistema de base de datos MySQL.



MySQL Workbench

Server Intranet MySQL x Mysql@192.168.1.93:3306 x

File Edit View Query Database Server Tools Scripting Help

Navigator: knowledge\_areas x

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

knowledge\_areas

- Columns
- Indexes
- Foreign Keys
- Triggers

Information

SQL: SELECT \* FROM dw2019.knowledge\_areas;

Limit to 1000 rows

Result Grid

id	name	icon	created_at	updated_at
1	Iniciación	wb_incandescent	NULL	NULL
2	Diseño	dashboard	NULL	NULL
3	Backend	dns	NULL	NULL
4	Frontend	web	NULL	NULL
5	Apps	apps	NULL	NULL
6	Sistemas	computer	NULL	NULL

Result Grid

Form Editor

Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	00:55:05	SELECT * FROM dw2019.knowledge_areas LIMIT 0, 10	6 row(s) returned	0.000 sec / 0.000 sec

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Fuente:

<https://icon-icons.com/es/icono/mysql-oficial-logo/169938>

<https://jaehoo.wordpress.com/2009/12/15/mysql-workbench-en-ubuntu-9-10/>

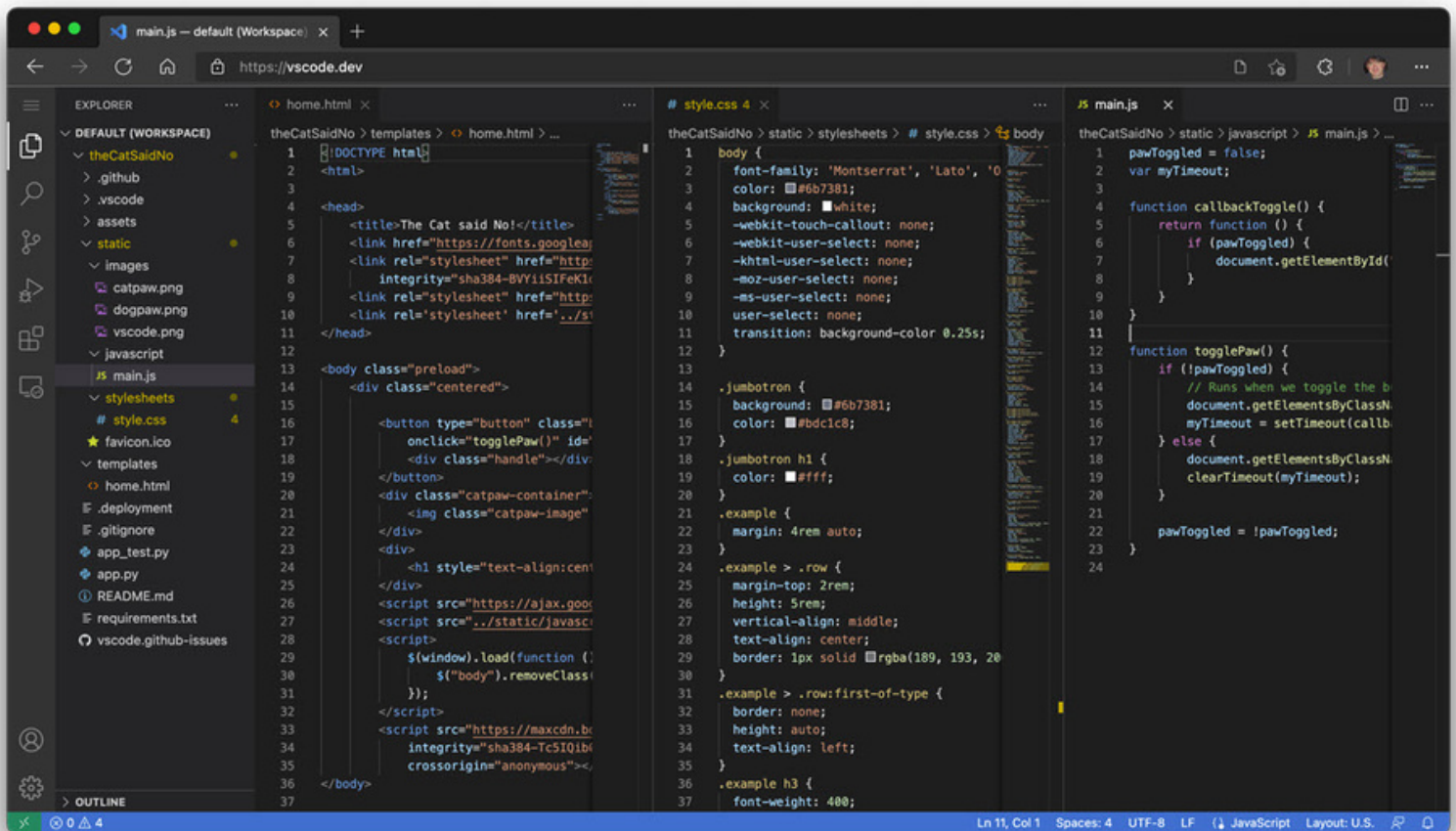
<https://www.arsys.es/blog/interfaces-graficas-mysql>

## Visual Studio Code



Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto,<sup>12</sup> aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft.<sup>3</sup>



Fuente:

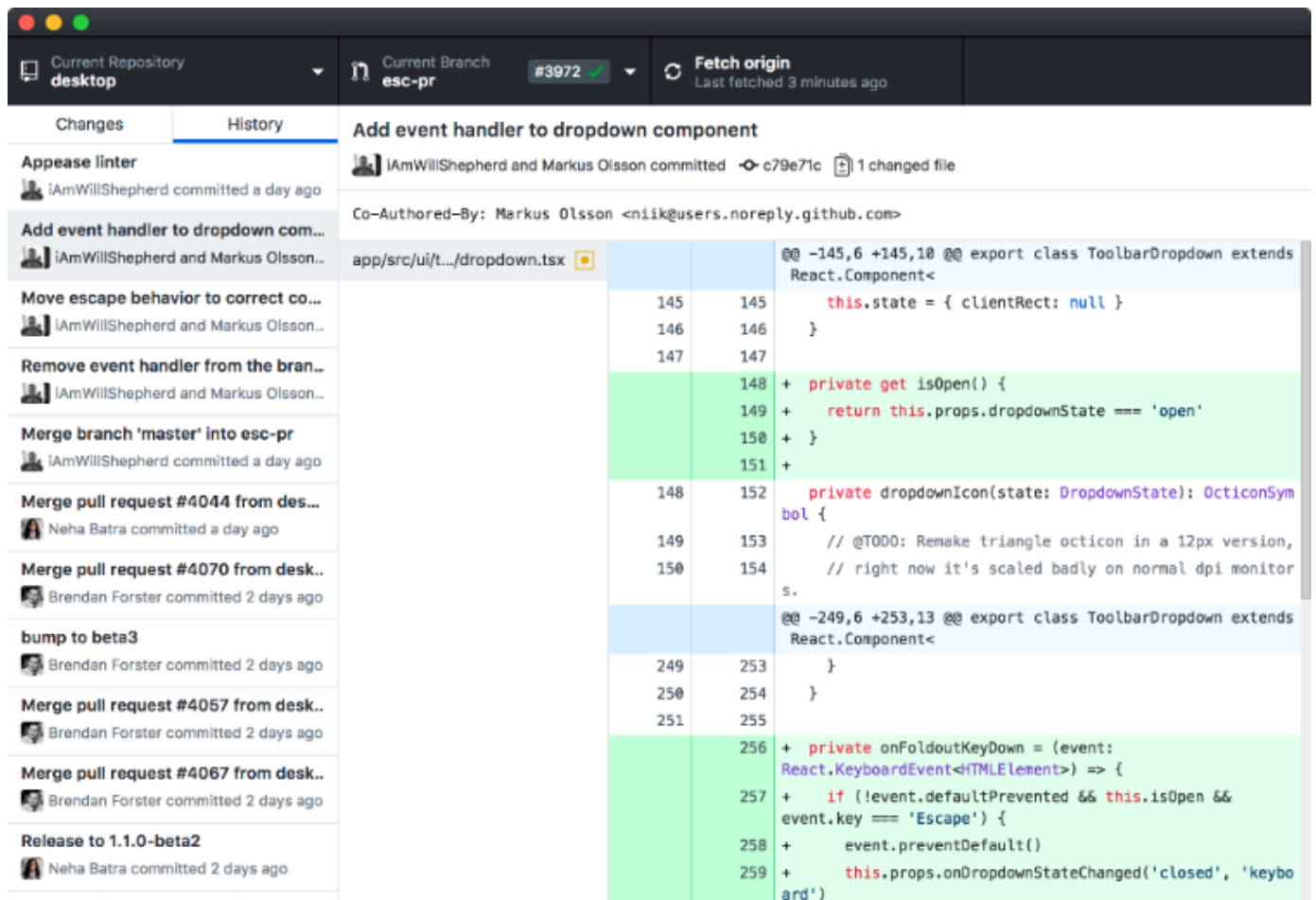
<https://www.whatsnew.com/2021/10/23/visual-studio-code-el-editor-de-codigo-de-microsoft-ya-tiene-una-version-web/>





GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.

El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Anteriormente era conocida como Logical Awesome LLC. El código de los proyectos alojados en GitHub se almacena generalmente de forma pública.



The screenshot shows the GitHub Desktop application interface. At the top, there's a header bar with 'Current Repository: desktop', 'Current Branch: esc-pr #3972', and a 'Fetch origin' button. Below this, the 'History' tab is selected, showing a list of commits. The commit 'Add event handler to dropdown component' by 'iAmWillShepherd and Markus Olsson' is highlighted. To the right, the diff view shows the changes in the file 'app/src/ui/t.../dropdown.tsx'. The diff highlights the addition of a new event handler for the 'onFoldoutKeyDown' event, which calls 'event.preventDefault()' and updates the 'dropdownState' to 'closed' when the 'Escape' key is pressed.

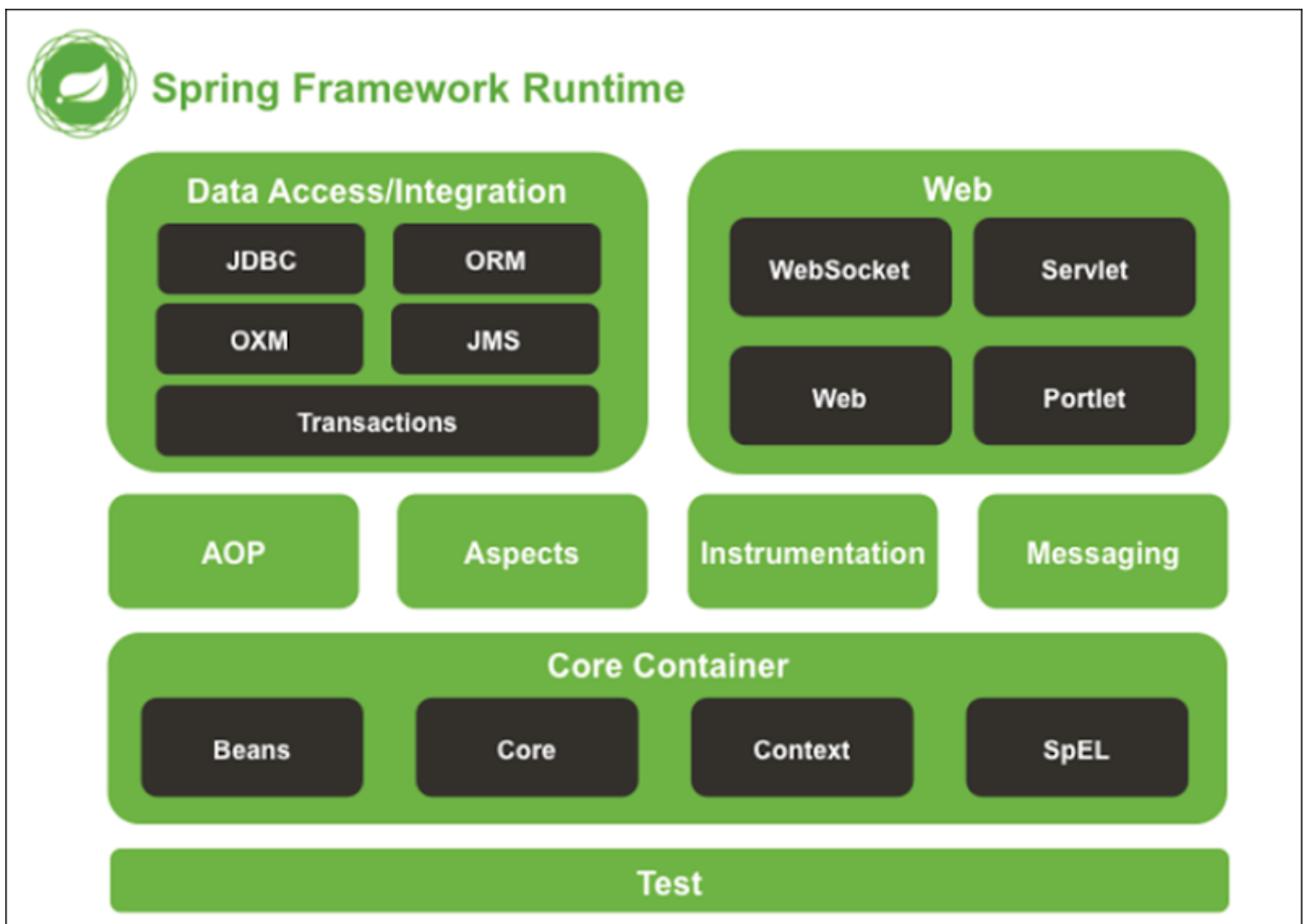
Commit Message	Author	Commit Hash	Files Changed
Add event handler to dropdown component	iAmWillShepherd and Markus Olsson	c79e71c	1 changed file
Co-Authored-By: Markus Olsson <niiik@users.noreply.github.com>			
app/src/ui/t.../dropdown.tsx			

```

@@ -145,6 +145,10 @@ export class ToolbarDropdown extends
  React.Component<
    this.state = { clientRect: null }
  }
+ private get isOpen() {
+   return this.props.dropdownState === 'open'
+ }
+
  private dropdownIcon(state: DropdownState): OcticonSym
  bol {
    // @T000: Remake triangle octicon in a 12px version,
    // right now it's scaled badly on normal dpi monitor
    s.
  }
+ private onFoldoutKeyDown = (event:
  React.KeyboardEvent<HTMLElement>) => {
+   if (!event.defaultPrevented && this.isOpen &&
  event.key === 'Escape') {
+     event.preventDefault()
+     this.props.onDropdownStateChanged('closed', 'keybo
  ard')
  }
  }
  }
  
```

Spring es un framework del lenguaje de programación java, y un framework en programación es el resultado de la evolución de la ingeniería del software, estos son creados por programadores para programadores, con la finalidad de estandarizar el trabajo, resolver, agilizar y manejar los problemas y complejidades que van apareciendo en el mundo de la programación, a medida las exigencias van creciendo.

Creando así, en la comunidad de desarrolladores, un abanico de posibilidades para una creación cada vez más evolucionada de aplicaciones.





# 4 Lenguajes Utilizados

Para la realización de esta aplicación web se han utilizado fundamentalmente 2 lenguajes:

## Java

Java es un lenguaje de programación orientado a objetos que se incorporó al ámbito de la informática en los años noventa. La idea de Java es que pueda realizarse programas con la posibilidad de ejecutarse en cualquier contexto, en cualquier ambiente, siendo así su portabilidad uno de sus principales logros.

Fue desarrollado por Sun Microsystems, posteriormente adquirido por Oracle. En la actualidad puede utilizarse de modo gratuito, pudiéndose conseguir sin problemas un paquete para desarrolladores que oriente la actividad de programar en este lenguaje. Puede ser modificado por cualquiera, circunstancia que lo convierte en lo que comúnmente se denomina “código abierto”.



## JavaScript

JavaScript es el lenguaje de programación encargado de dotar de mayor interactividad y dinamismo a las páginas web. Cuando JavaScript se ejecuta en el navegador, no necesita de un compilador. El navegador lee directamente el código, sin necesidad de terceros. Por tanto, se le reconoce como uno de los tres lenguajes nativos de la web junto a HTML (contenido y su estructura) y a CSS (diseño del contenido y su estructura).



# 5 Fases del Proyecto

Esta sección expondremos los diferentes ciclos que ha tenido el proyecto:

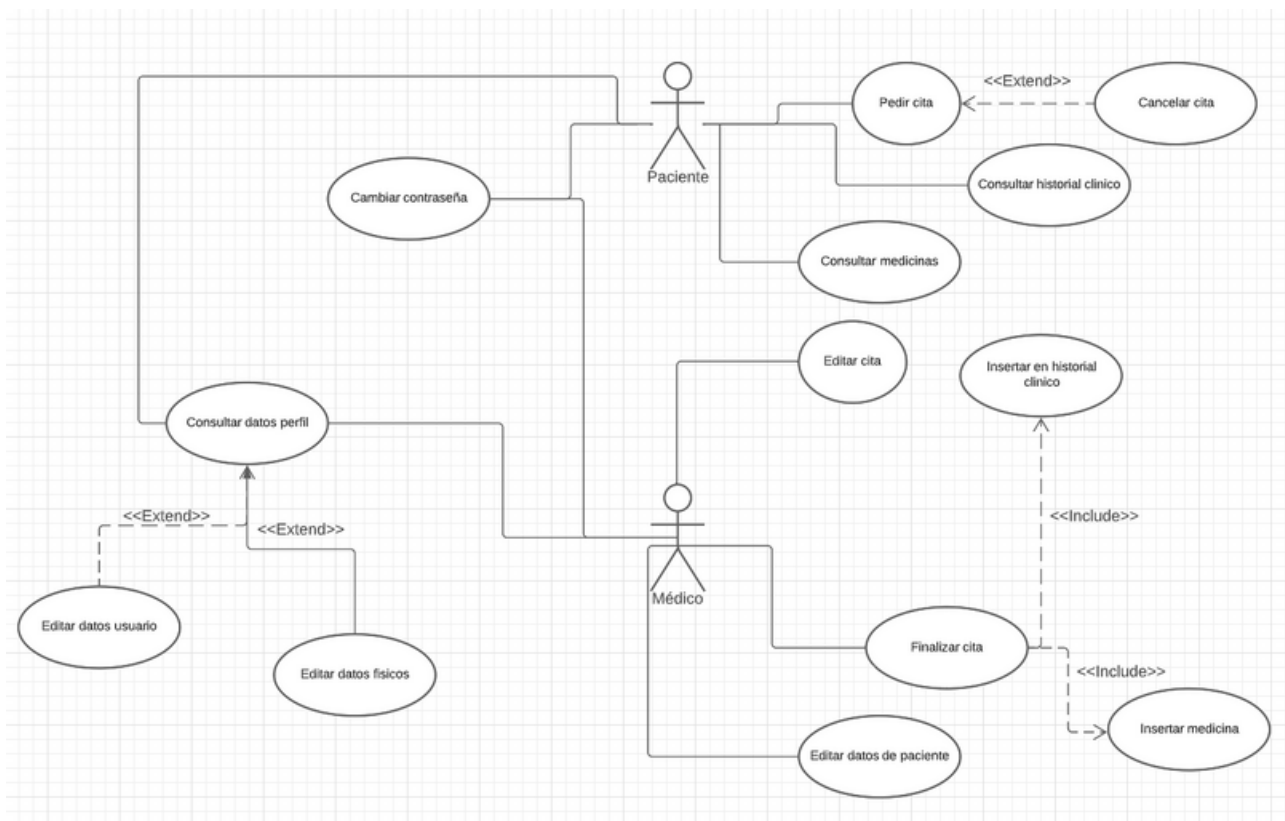
## 1 Idea

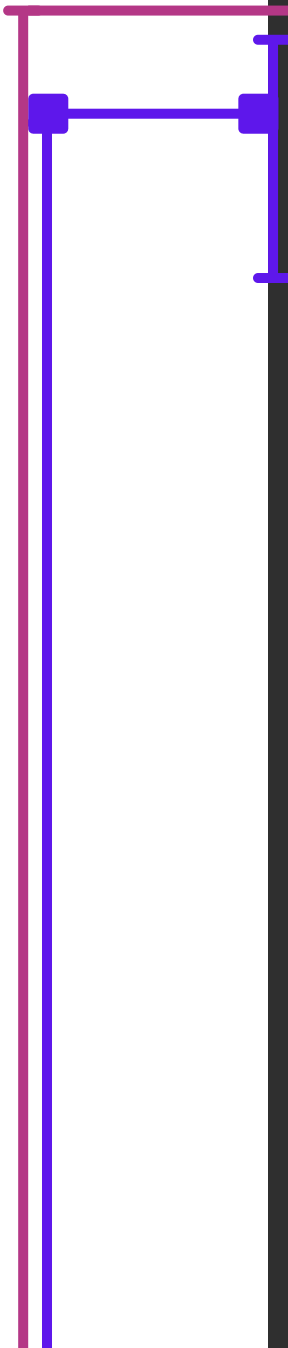
El equipo, después de pensar varias ideas para realizar su trabajo fin de grado (TFG). Nos surgen algunas ideas, las cuales eran montar un blog donde usuario intercambien información sobre diversos temas o una web de un restaurante de comida. Después de meditarlo optamos por realizar un proyecto que auné estos conceptos y sea algo actual, por lo que se optó por realizar una aplicación web de gestión médica.

## 2 Casos de Uso

El diagrama de casos de uso es una forma de diagrama de comportamiento en lenguaje de modelado unificado (UML, del inglés Unified Modelling Language), con la que se representan procesos empresariales, así como sistemas y procesos de programación orientada a objetos.

En nuestro proyecto los actores involucrados en los casos de uso son: Paciente y Médico





```
classDiagram
    class HomeController {
        +PasswordEncoder pwenco
        +IntUsuarioDao udao
        +IntPerfilDao pdao
        +IntDiagnosticoDao ddao
        +IntInformacionDao idao
        +IntCitaDao cdao
        +IntComentarioDao comendao
        +IntMedicinaDao mdao
        +IntHistorialClinicoDao hdao
        +acceso(HttpSession) String
        +mostrarFormInicio(Model) String
        +procesarLogin(Authentication, Model, HttpSession) String
        +buscarCitaPorNombre(Model) String
        +buscarCitaPorNombre(Model, HttpSession, @RequestParam(value="nombre") String) String
        +mostrarUsuario(Model) String
        +verUnUsuario(Model, @PathVariable String) String
        +mostrarCita(Model, @PathVariable int) String
        +mostrarEditar(Model, @PathVariable int) String
        +editarUna(Model, @RequestParam(value="idCita") String, @RequestParam(value="fechaCita") String) String
        +mostrarPedirCita(Model) String
        +pedircita(RedirectAttributes, Model, RedirectAttributes, @RequestParam(value="fechaCita") @DateTime String) String
        +cancelarCita(Model, @PathVariable int) String
        +modificarDatosUsuario(RedirectAttributes, Model, @RequestParam(value="dni") String, @RequestParam(value="password") String) String
        +modificarInformacion(RedirectAttributes, Model, @RequestParam(value="peso") Integer, @RequestParam(value="altura") Integer) String
        +modificarDiagnostico(RedirectAttributes, Model, @RequestParam(value="alergias") String, @RequestParam(value="diagnostico") String) String
        +mostrarRegistro(Model) String
        +procesarRegistro(Model, Usuario) String
        +mostrarAyuda(Model) String
        +mostrarPacientes(Model) String
        +mostrarMedicinas(Model, HttpSession) String
        +mostrarMedicos(Model, HttpSession) String
        +mostrarchat(Model, HttpSession, @PathVariable String) String
        +enviarmensaje(Model, HttpSession, @RequestParam(value="dni1chat") String, @RequestParam(value="mensaje") String) String
        +mostrarTodas(Model, HttpSession) String
        +buscarCitasPorMedico(Model, HttpSession) String
        +mostrarHistorial(Model, HttpSession) String
        +mostrarTodoschat(Model, HttpSession) String
        +mostrarRecuperarContrasena(Model) String
        +mostrarContrasena(Model) String
        +cambioContraseña(Model, HttpSession, @RequestParam(value="contraseñaActual") String, @RequestParam(value="contraseñaNueva") String) String
        +generarEncriptado() String
        +salirprueba(HttpServletRequest, HttpSession) String
        +logout(HttpServletRequest, HttpSession) String
    }
    HomeController --> PasswordEncoder
    HomeController --> IntUsuarioDao
    HomeController --> IntPerfilDao
    HomeController --> IntDiagnosticoDao
    HomeController --> IntInformacionDao
    HomeController --> IntCitaDao
    HomeController --> IntComentarioDao
    HomeController --> IntMedicinaDao
    HomeController --> IntHistorialClinicoDao
```

HomeController

- pwenco : PasswordEncoder
- udao : IntUsuarioDao
- pdao : IntPerfilDao
- ddao : IntDiagnosticoDao
- idao : IntInformacionDao
- cdao : IntCitaDao
- comendao : IntComentarioDao
- mdao : IntMedicinaDao
- hdao : IntHistorialClinicoDao
- acceso(HttpSession) : String
- mostrarFormInicio(Model) : String
- procesarLogin(Authentication, Model, HttpSession) : String
- buscarCitaPorNombre(Model) : String
- buscarCitaPorNombre(Model, HttpSession, @RequestParam(value="nombre") String) : String
- mostrarUsuario(Model) : String
- verUnUsuario(Model, @PathVariable String) : String
- mostrarCita(Model, @PathVariable int) : String
- mostrarEditar(Model, @PathVariable int) : String
- editarUna(Model, @RequestParam(value="idCita") String, @RequestParam(value="fechaCita") String, @RequestParam(value="fechaCita") @DateTime String) : String
- mostrarPedirCita(Model) : String
- pedircita(RedirectAttributes, Model, RedirectAttributes, @RequestParam(value="fechaCita") @DateTime String) : String
- cancelarCita(Model, @PathVariable int) : String
- modificarDatosUsuario(RedirectAttributes, Model, @RequestParam(value="dni") String, @RequestParam(value="password") String) : String
- modificarInformacion(RedirectAttributes, Model, @RequestParam(value="peso") Integer, @RequestParam(value="altura") Integer) : String
- modificarDiagnostico(RedirectAttributes, Model, @RequestParam(value="alergias") String, @RequestParam(value="diagnostico") String) : String
- mostrarRegistro(Model) : String
- procesarRegistro(Model, Usuario) : String
- mostrarAyuda(Model) : String
- mostrarPacientes(Model) : String
- mostrarMedicinas(Model, HttpSession) : String
- mostrarMedicos(Model, HttpSession) : String
- mostrarchat(Model, HttpSession, @PathVariable String) : String
- enviarmensaje(Model, HttpSession, @RequestParam(value="dni1chat") String, @RequestParam(value="mensaje") String) : String
- mostrarTodas(Model, HttpSession) : String
- buscarCitasPorMedico(Model, HttpSession) : String
- mostrarHistorial(Model, HttpSession) : String
- mostrarTodoschat(Model, HttpSession) : String
- mostrarRecuperarContrasena(Model) : String
- mostrarContrasena(Model) : String
- cambioContraseña(Model, HttpSession, @RequestParam(value="contraseñaActual") String, @RequestParam(value="contraseñaNueva") String) : String
- generarEncriptado() : String
- salirprueba(HttpServletRequest, HttpSession) : String
- logout(HttpServletRequest, HttpSession) : String

```

▼ C Usuario
  serialVersionUID : long
  dni : String
  apellido : String
  direccion : String
  email : String
  enabled : int
  imgurl : String
  nombre : String
  onlineusu : int
  password : String
  telefono : String
  citas : List<Cita>
  diagnosticos : List<Diagnostico>
  informacions : List<Informacion>
  perfils : List<Perfil>
  Usuario()
  getDni() : String
  setDni(String) : void
  getApellido() : String
  setApellido(String) : void
  getDireccion() : String
  setDireccion(String) : void
  getEmail() : String
  setEmail(String) : void
  getEnabled() : int
  setEnabled(int) : void
  getImgurl() : String
  setImgurl(String) : void
  getNombre() : String
  setNombre(String) : void
  getOnlineusu() : int
  setOnlineusu(int) : void
  getPassword() : String
  setPassword(String) : void
  getTelefono() : String
  setTelefono(String) : void
  getCitas() : List<Cita>
  setCitas(List<Cita>) : void
  addCita(Cita) : Cita
  removeCita(Cita) : Cita
  getDiagnosticos() : List<Diagnostico>
  setDiagnosticos(List<Diagnostico>) : void
  addDiagnostico(Diagnostico) : Diagnostico
  removeDiagnostico(Diagnostico) : Diagnostico
  getInformacions() : List<Informacion>
  setInformacions(List<Informacion>) : void
  addInformacion(Informacion) : Informacion
  removeInformacion(Informacion) : Informacion
  getPerfils() : List<Perfil>
  setPerfils(List<Perfil>) : void
  toString() : String
  hashCode() : int
  equals(Object) : boolean
  
```

```

▼ C Cita
  serialVersionUID : long
  idCita : int
  direccionCentrosalud : String
  estado : String
  fechaCita : Date
  horaCita : String
  nombreMedico : String
  nombrePaciente : String
  sintomas : String
  usuario : Usuario
  historialClinicos : List<HistorialClinico>
  Cita()
  getIdCita() : int
  setIdCita(int) : void
  getDireccionCentrosalud() : String
  setDireccionCentrosalud(String) : void
  getEstado() : String
  setEstado(String) : void
  getFechaCita() : Date
  setFechaCita(Date) : void
  getHoraCita() : String
  setHoraCita(String) : void
  getNombreMedico() : String
  setNombreMedico(String) : void
  getNombrePaciente() : String
  setNombrePaciente(String) : void
  getSintomas() : String
  setSintomas(String) : void
  getUsuario() : Usuario
  setUsuario(Usuario) : void
  getHistorialClinicos() : List<HistorialClinico>
  setHistorialClinicos(List<HistorialClinico>) : void
  addHistorialClinico(HistorialClinico) : HistorialClinico
  removeHistorialClinico(HistorialClinico) : HistorialClinico
  toString() : String
  
```

```

▼ C Informacion
  serialVersionUID : long
  idInfo : int
  altura : String
  edad : int
  peso : int
  sexo : String
  usuario : Usuario
  Informacion()
  Informacion(Usuario, String, int, int, String)
  getIdInfo() : int
  setIdInfo(int) : void
  getAltura() : String
  setAltura(String) : void
  getEdad() : int
  setEdad(int) : void
  getPeso() : int
  setPeso(int) : void
  getSexo() : String
  setSexo(String) : void
  getUsuario() : Usuario
  setUsuario(Usuario) : void
  
```

```

▼ C Perfil
  serialVersionUID : long
  idPerfil : int
  descPer : String
  usuarios : List<Usuario>
  Perfil()
  getIdPerfil() : int
  setIdPerfil(int) : void
  getDescPer() : String
  setDescPer(String) : void
  getUsuarios() : List<Usuario>
  setUsuarios(List<Usuario>) : void
  toString() : String
  hashCode() : int
  equals(Object) : boolean
  
```

```

▼ C Comentario
  serialVersionUID : long
  idCom : int
  comentario : String
  dni1 : String
  dni2 : String
  Comentario()
  getIdCom() : int
  setIdCom(int) : void
  getComentario() : String
  setComentario(String) : void
  getDni1() : String
  setDni1(String) : void
  getDni2() : String
  setDni2(String) : void
  toString() : String
  
```

```

HistorialClinico
  serialVersionUID : long
  idHistorial : int
  desCitahis : String
  instrucciones : String
  cita : Cita
  medicina : Medicina
  HistorialClinico()
  HistorialClinico(String, String, Cita, Medicina)
  getIdHistorial() : int
  setIdHistorial(int) : void
  getDesCitahis() : String
  setDesCitahis(String) : void
  getInstrucciones() : String
  setInstrucciones(String) : void
  getCita() : Cita
  setCita(Cita) : void
  getMedicina() : Medicina
  setMedicina(Medicina) : void
  
```

```

Diagnostico
  serialVersionUID : long
  idDiag : int
  alergias : String
  enfermedades : String
  operaciones : String
  tratamiento : String
  usuario : Usuario
  Diagnostico()
  Diagnostico(String, String, String, String, Usuario)
  getIdDiag() : int
  setIdDiag(int) : void
  getAlergias() : String
  setAlergias(String) : void
  getEnfermedades() : String
  setEnfermedades(String) : void
  getOperaciones() : String
  setOperaciones(String) : void
  getTratamiento() : String
  setTratamiento(String) : void
  getUsuario() : Usuario
  setUsuario(Usuario) : void
  
```

```

Medicina
  serialVersionUID : long
  codMed : int
  activo : String
  cantidad : int
  fechaFin : Date
  fechaInicio : Date
  nombreMed : String
  historialClinicos : List<HistorialClinico>
  tipo : Tipo
  Medicina()
  getCodMed() : int
  setCodMed(int) : void
  getActivo() : String
  setActivo(String) : void
  getCantidad() : int
  setCantidad(int) : void
  getFechaFin() : Date
  setFechaFin(Date) : void
  getFechaInicio() : Date
  setFechaInicio(Date) : void
  getNombreMed() : String
  setNombreMed(String) : void
  getHistorialClinicos() : List<HistorialClinico>
  setHistorialClinicos(List<HistorialClinico>) : void
  addHistorialClinico(HistorialClinico) : HistorialClinico
  removeHistorialClinico(HistorialClinico) : HistorialClinico
  getTipo() : Tipo
  setTipo(Tipo) : void
  
```

```

Tipo
  serialVersionUID : long
  idTipo : int
  descripcion : String
  medicinas : List<Medicina>
  Tipo()
  getIdTipo() : int
  setIdTipo(int) : void
  getDescripcion() : String
  setDescripcion(String) : void
  getMedicinas() : List<Medicina>
  setMedicinas(List<Medicina>) : void
  addMedicina(Medicina) : Medicina
  removeMedicina(Medicina) : Medicina
  
```

```

▼ I IntCitaDao
  ● buscarCitas(String) : List<Cita>
  ● buscarTodos() : List<Cita>
  ● insertUna(Cita) : int
  ● buscarUnaCita(int) : Cita
  ● buscarCitasPorMedico(String) : List<Cita>
  ● buscarCitaPorNombre(String) : List<Cita>
  ● editarCita(Cita) : int
    
```

```

▼ C CitaDaoImplMy8
  ● citarepo : CitaRepo
  ● buscarCitas(String) : List<Cita>
  ● buscarTodos() : List<Cita>
  ● buscarCitaPorNombre(String) : List<Cita>
  ● insertUna(Cita) : int
  ● buscarUnaCita(int) : Cita
  ● buscarCitasPorMedico(String) : List<Cita>
  ● editarCita(Cita) : int
    
```

```

▼ I IntComentarioDao
  ● mostrarChat(String, String) : List<Comentario>
  ● enviarComentario(Comentario) : int
    
```

```

▼ C ComentarioDaoImplMy8
  ● comentariorepo : ComentarioRepo
  ● mostrarChat(String, String) : List<Comentario>
  ● enviarComentario(Comentario) : int
    
```

```

▼ I IntDiagnosticoDao
  ● buscarDiagnostico(String) : Diagnostico
  ● buscarTodos() : List<Diagnostico>
  ● insertUno(Diagnostico) : int
  ● editarDiagnostico(Diagnostico) : int
    
```

```

▼ C DiagnosticoDaoImplMy8
  ● diarepo : DiagnosticoRepo
  ● buscarDiagnostico(String) : Diagnostico
  ● buscarTodos() : List<Diagnostico>
  ● insertUno(Diagnostico) : int
  ● editarDiagnostico(Diagnostico) : int
    
```

```

▼ I IntHistorialClinicoDao
  ● insertUna(HistorialClinico) : int
  ● buscarHistorialClinicoPordni(String) : List<HistorialClinico>
    
```

```

▼ C HistorialClinicoDaoImplMy8
  ● historialrepo : HistorialClinicoRepo
  ● insertUna(HistorialClinico) : int
  ● buscarHistorialClinicoPordni(String) : List<HistorialClinico>
    
```

```

▼ I IntInformacionDao
  ● buscarInformacion(String) : Informacion
  ● buscarTodos() : List<Informacion>
  ● insertUno(Informacion) : int
  ● editarInformacion(Informacion) : int
    
```

```

▼ C InformacionDaoImplMy8
  ● infrepo : InformacionRepo
  ● buscarInformacion(String) : Informacion
  ● buscarTodos() : List<Informacion>
  ● insertUno(Informacion) : int
  ● editarInformacion(Informacion) : int
    
```

```

▼ I IntMedicinaDao
  ● buscarTodasMedicinas() : List<Medicina>
  ● buscarUnaMedicina(String) : Medicina
  ● buscarMedicinasUsuario(String) : List<Medicina>
    
```

```

▼ C MedicinaDaoImplMy8
  ● medirepo : MedicinaRepo
  ● buscarTodasMedicinas() : List<Medicina>
  ● buscarUnaMedicina(String) : Medicina
  ● buscarMedicinasUsuario(String) : List<Medicina>
    
```

```

▼ I IntPerfilDao
  ● buscarPerfil(int) : Perfil
  ● buscarTodos() : List<Perfil>
  ● insertUno(Perfil) : int
    
```

```

▼ C PerfilDaoImplMy8
  ● pfrepo : PerfilRepo
  ● buscarPerfil(int) : Perfil
  ● buscarTodos() : List<Perfil>
  ● insertUno(Perfil) : int
    
```

```

▼ I IntUsuarioDao
  ● buscarUsuario(String) : Usuario
  ● buscarTodos() : List<Usuario>
  ● buscarMedicos(String) : List<Usuario>
  ● buscarConectados(String) : List<Usuario>
  ● buscarPacientes() : List<Usuario>
  ● listaMedicos() : List<Usuario>
  ● contarMedicos() : Integer
  ● insertUno(Usuario) : int
  ● cambioContraseña(String, String) : void
  ● buscarUsuarioPorNombre(String) : Usuario
  ● editarUsuario(Usuario) : int
    
```

```

▼ C UsuarioDaoImplMy8
  ● urepo : UsuarioRepo
  ● buscarUsuario(String) : Usuario
  ● buscarTodos() : List<Usuario>
  ● cambioContraseña(String, String) : void
  ● insertUno(Usuario) : int
  ● buscarUsuarioPorNombre(String) : Usuario
  ● editarUsuario(Usuario) : int
  ● contarMedicos() : Integer
  ● buscarConectados(String) : List<Usuario>
  ● buscarPacientes() : List<Usuario>
  ● buscarMedicos(String) : List<Usuario>
  ● listaMedicos() : List<Usuario>
    
```

# Modelo repository

```
▼ I UsuarioRepo
  ● buscarConectados(String) : List<Usuario>
  ● getByNombre(String) : Usuario
  ● mostrarMedicos(String) : List<Usuario>
  ● listaMedicos() : List<Usuario>
  ● mostrarPacientes() : List<Usuario>
  ● contarMedicos() : Integer
```

I PerfilRepo

```
▼ I CitaRepo
  ● findCitaByUsuario(String) : List<Cita>
  ● findUnaCita(Integer) : Cita
  ● findCitaByMedico(String) : List<Cita>
  ● buscarCitaPorNombre(String) : List<Cita>
```

```
▼ I MedicinaRepo
  ● todasMedicinas() : List<Medicina>
  ● buscarUnaMedicina(String) : Medicina
  ● buscarMedicinasUsuario(String) : List<Medicina>
```

```
▼ I InformacionRepo
  ● findInformacionbyUsuario(String) : Informacion
```

```
▼ I HistorialClinicoRepo
  ● buscarHistorialClinicoPordni(String) : List<HistorialClinico>
```

```
▼ I DiagnosticoRepo
  ● findDiagnosticobyUsuario(String) : Diagnostico
```

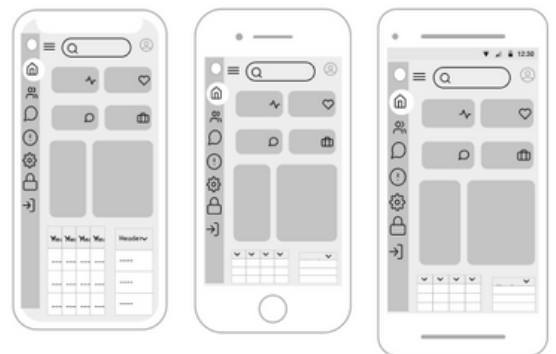
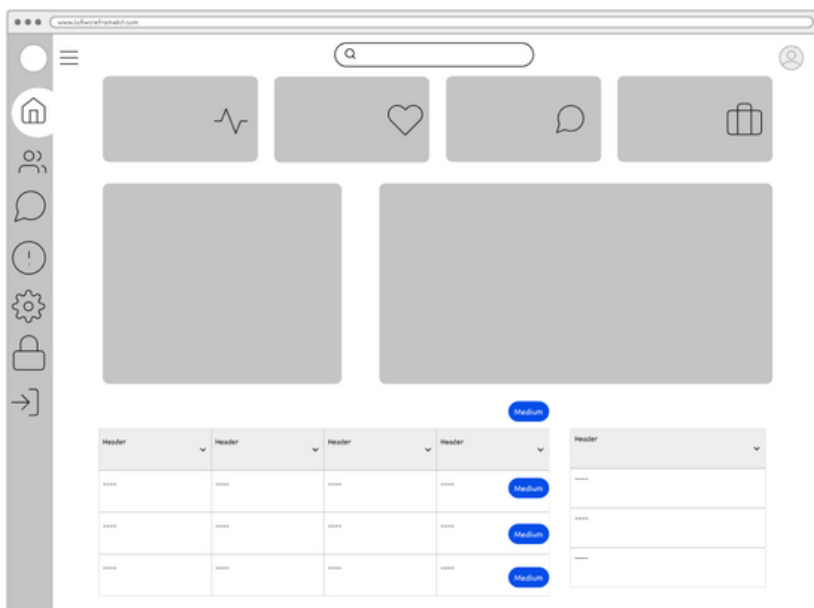
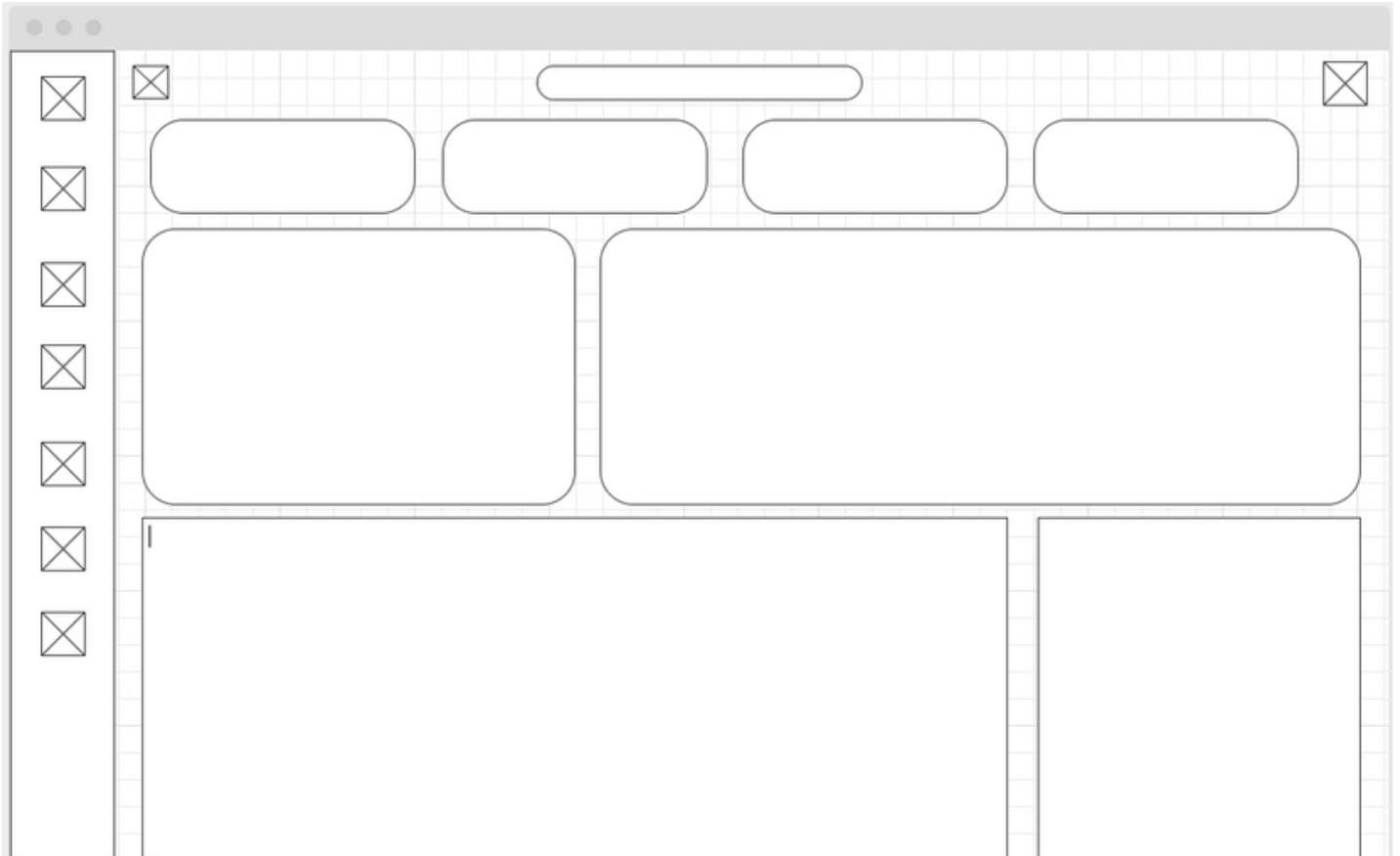
```
▼ I ComentarioRepo
  ● mostrarChat(String, String) : List<Comentario>
```

```
▼ C SecurityConfig
  ● passwordEncoder() : PasswordEncoder
  ● dataSource : DataSource
  ● configure(AuthenticationManagerBuilder) : void
  ● configure(HttpSecurity) : void
```

## Security

### 3 WireFrame

Para hacer el wireframe he utilizado la herramienta de figma para crear como será la interfaz de usuario (UI) y teniendo en consideración la experiencia de usuario (UX) , con la finalidad de crear un diseño simple y que te aporte la información precisa a el alcance de un click, primando los bordes redondeados y diseñando también teniendo muy en cuenta los dispositivos móviles, haciendo uso de diseño enteramente responsivo.





# 4 Guía de Estilos

Las guías de estilo, también denominadas libros de estilo, hacen referencia en el ámbito del marketing online a los manuales para el diseño uniforme de páginas web. Aquí, muestro la guía de estilo de este proyecto.

## Logo



Tipografía: 'Poppins', sans-serif;

## Titulos

font-size: 2em

Parrafo

font-weight: 500

## Colores:

Realizado #8de02c

Pendiente #f9ca3f

Cancelado #f00

Enprogreso #1795c9



--blue: #287bff

--white: #fff



--grey: #f5f5f5



--black1: #222



--black2: #999

## Iconos (ion-icon):



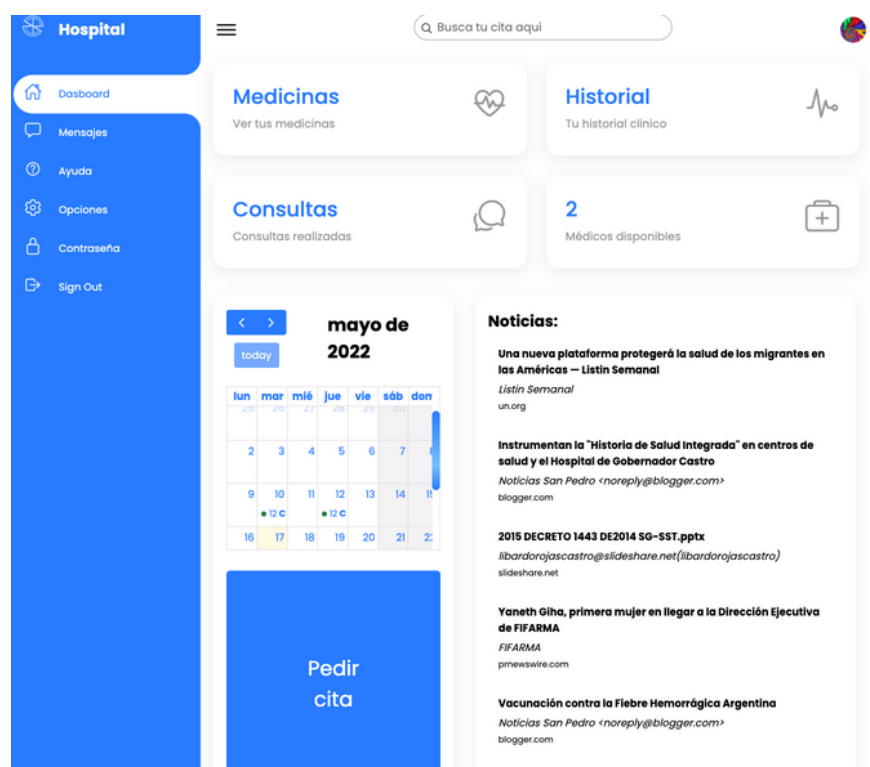
pulse-outline

<ion-icon name="pulse-outline"></ion-icon>

## Estilo de cajas:

border-radius: 20px

box-shadow: 0 7px 25px rgb(0 0 0 / 8%)



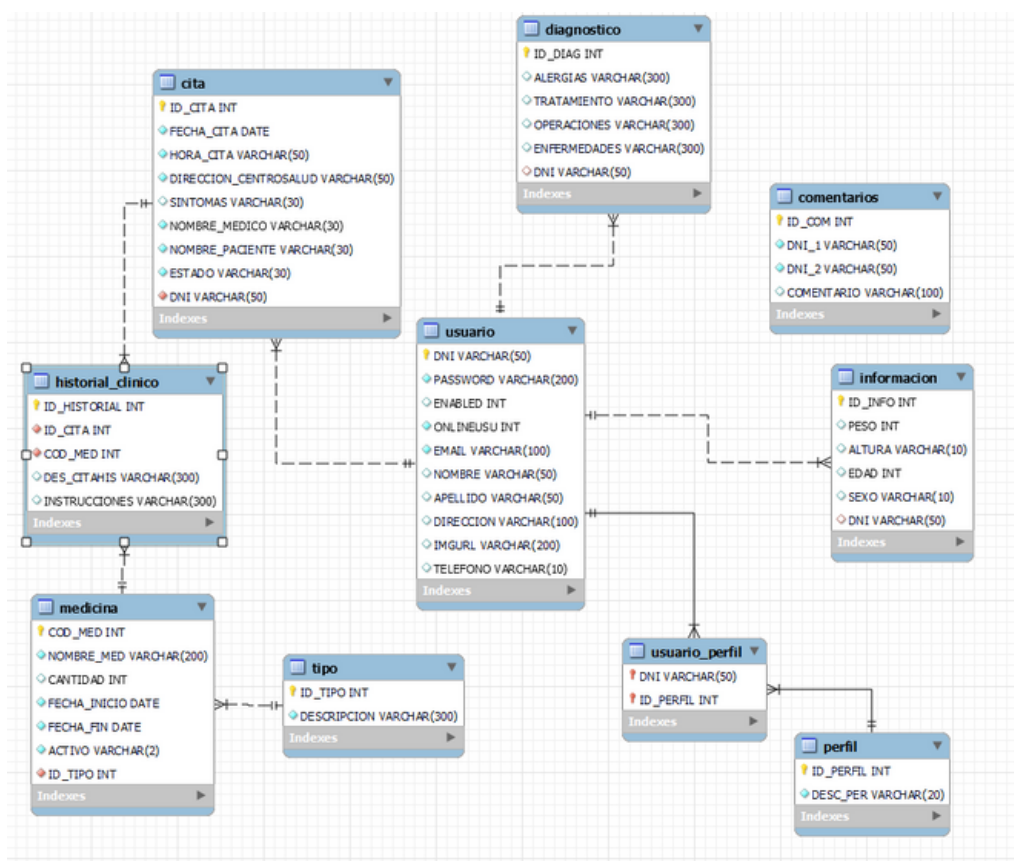
# 5 Bases de Datos

Como nuestra aplicación va a tener: usuarios con sus respectivos datos, medicinas que se pueden recetar, las citas de cuando te podrán atender, el historial medico de cada usuario, necesitaremos un sitio donde poder almacenar esos datos.



Por eso tenemos que crear una Base de datos utilizando SQL como lenguaje y MySQL Workbench como herramienta además de poder almacenar los datos también podemos: acceder, modificarlos, o guardar nuevos registros.

Para este caso se decidió crear un esquema de datos que reúna los datos que vemos necesarios para el mejor funcionamiento de la aplicación.



Hemos utilizado **DDL** (Lenguaje de Definición de Datos) para crear la base de datos y **DML** (Lenguaje de Manipulación de Datos) para insertar y manipular la información.

```
CREATE TABLE IF NOT EXISTS 'hospital_bbdd'. 'usuario' (
  'DNI' VARCHAR(50) NOT NULL,
  'PASSWORD' VARCHAR(200) NOT NULL,
  'ENABLED' INT NULL DEFAULT NULL,
  'ONLINEUSU' INT NOT NULL,
  'EMAIL' VARCHAR(100) NOT NULL,
  'NOMBRE' VARCHAR(50) NULL DEFAULT NULL,
  'APELLIDO' VARCHAR(50) NULL DEFAULT NULL,
  'DIRECCION' VARCHAR(100) NULL DEFAULT NULL,
  'IMGURL' VARCHAR(200) NULL DEFAULT NULL,
  'TELEFONO' VARCHAR(10) NULL DEFAULT NULL,
  PRIMARY KEY ('DNI'),
  UNIQUE INDEX 'EMAIL' ('EMAIL' ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_spanish_ci;
```

**DDL**

```
INSERT INTO 'hospital_bbdd'. 'informacion' ('PESO', 'ALTURA', 'EDAD', 'SEXO', 'DNI') VALUES ('23', '1,23', '23', 'm', '34728920e');
INSERT INTO 'hospital_bbdd'. 'informacion' ('PESO', 'ALTURA', 'EDAD', 'SEXO', 'DNI') VALUES ('34', '1,45', '32', 'f', '483920e');
INSERT INTO 'hospital_bbdd'. 'informacion' ('PESO', 'ALTURA', 'EDAD', 'SEXO', 'DNI') VALUES ('70', '1,80', '45', 'm', '503942f');
INSERT INTO 'hospital_bbdd'. 'informacion' ('PESO', 'ALTURA', 'EDAD', 'SEXO', 'DNI') VALUES ('65', '1,70', '32', 'm', '6574848e');
INSERT INTO 'hospital_bbdd'. 'informacion' ('PESO', 'ALTURA', 'EDAD', 'SEXO', 'DNI') VALUES ('68', '1,78', '25', 'm', '71706550e');
```

**DML**

El código SQL que se utilizó para crear la bbdd se localiza en la misma carpeta donde está este pdf.

# 6 HTML

HTML (HyperText Markup Language) es un lenguaje de marcas utilizado para crear paginas web con este lenguaje podremos mostrar el contenido de la web.



Esto se hace utilizando como herramienta el Visual Studio Code, no es obligatorio usar esta herramienta pero es bastante cómoda.

Algunos ejemplos del código quedarían así,

Codigo del index.html

Asi se veria sin ningun tipo de estilo css

```
<body>
<div class="container">
  <div class="navegation">
    <ul>
      <li>
        <a href="/index">
          <span class="icon"></span>
          <span class="title">
            <h2>Hospital</h2>
          </span>
        </a>
      </li>
      <li class="hovered">
        <a href="/index">
          <span class="icon">
            <ion-icon name="home-outline"></ion-icon>
          </span>
          <span class="title">Dashboard</span>
        </a>
      </li>
      <sec:authorize access="hasAuthority('Medico')">
      <li>
        <a href="/pacientes">
          <!-- mostrar solo a medico -->
          <span class="icon">
            <ion-icon name="people-outline"></ion-icon>
          </span>
          <span class="title">Pacientes</span>
        </a>
      </li>
      </sec:authorize>
      <li>
        <a href="/todoschat">
          <span class="icon">
            <ion-icon name="chatbox-outline"></ion-icon>
          </span>
          <span class="title">Mensajes</span>
        </a>
      </li>
    </ul>
  </div>
</div>
```

## Hospital

- [Dashboard](#)
- [Pacientes](#)
- [Mensajes](#)
- [Ayuda](#)
- [Opciones](#)
- [Contraseña](#)
- [Sign Out](#)

Como se puede apreciar de esta manera no se vería muy agradable para el usuario por eso le tendríamos que añadir el estilo utilizando css

# 7 CSS

Para los estilos de este proyecto se contará con 3 archivos css los cuales son:

- style.css
- stylelogin.css
- styleusuario.css

También se han realizados pequeños ajustes puntuales e individuales utilizando <style> dentro del html, para realizar un diseño lo mas responsivo posible se ha utilizado el inspector del navegador para ir creando las mediaqueries necesarias.

Igualmente también se han utilizado keyframes para la animación del icono y efectos de hover para recalcar botones o campos interactivos.

Las mediaqueries van desde los @media (max-width:1910px) a los @media (max-width: 320px)

```
.user:hover{
  box-shadow: 0 0 0 0 rgba(40, 123, 255, 1);
  transform: scale(1);
  animation: pulse 2s infinite;
}

@keyframes pulse {
  0% {
    transform: scale(0.95);
    box-shadow: 0 0 0 0 rgba(40, 123, 255, 0.7);
  }
}
```

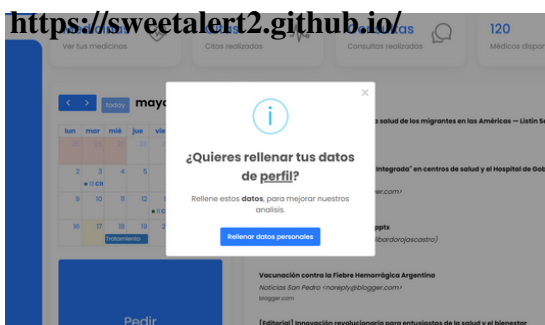
```
@media (max-width:1910px){
  .infoheader .caja{
    display: flex;
    width: 99%;
    flex-direction: column;
    justify-content: space-evenly;
    justify-content: flex-start;
  }
}
```

```
@media (max-width: 320px){
  .user {
    position: relative;
    width: 110px;
    height: 52px;
    left: 38px;
    border-radius: 50%;
    overflow: hidden;
    cursor: pointer;
  }
}
```

# 8 JavaScript

En este proyecto, hemos utilizado el JavaScript con el objetivo de dotar funcionalidad a la pagina, lo hemos utilizado para realizar validaciones de formularios tanto en la navegación del propio menú de la pagina, como para mostrar alertas personalizada utilizando:

<https://sweetalert2.github.io/>



```
//----- Revisar BOTON Mostrar solo cuando no haya datos de informacion d
Swal.fire({
  title: "<strong>¿Quieres rellenar tus datos de <u>perfil</u>?</strong>",
  icon: "info",
  html:
    "Rellene estos <b>datos</b>," + "para mejorar nuestros analisis.",
  showCloseButton: true,
  focusConfirm: false,
  confirmButtonText:
    '<a href="/usuario.html" class="btn">Rellenar datos personales</a> ',
});
```

```
//ejecutar la funcion validar() cuando se pulse el boton de registre ----- NO VAL
function validar() {
  var dni = document.getElementById("dni").value;
  var password = document.getElementById("password").value;
  var email = document.getElementById("email").value;
  var nombre = document.getElementById("nombre").value;
  var apellidos = document.getElementById("apellidos").value;
  var direccion = document.getElementById("direccion").value;
  var telefono = document.getElementById("telefono").value;
  var check = document.getElementById("check").checked;
  if (dni == "" || password == "" || email == "" || nombre == "" || apellidos == ""
  ) {
    Swal.fire({
      icon: 'error',
      title: 'Oops...',
      text: 'Debes rellenar todos los campos',
    });
    return false;
  } else {
    return true;
  }
}
```

```
let toggle = document.querySelector("#toggle");
let navigation = document.querySelector("#navigation");
let main = document.querySelector("#main");

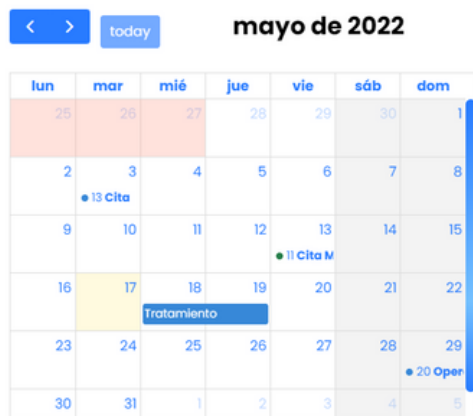
toggle.onclick = function () {
  navigation.classList.toggle("active");
  main.classList.toggle("active");
};

let list = document.querySelectorAll("#navigation li");

function activeLink() {
  list.forEach((item) => item.classList.remove("hovered"));
  this.classList.add("hovered");
}
list.forEach((item) => item.addEventListener("mouseover", activeLink));
```

# 9 API's

En la parte del desarrollo front hemos incorporado una api que muestra las citas que tiene cada usuario en un calendario utilizando la api: <https://fullcalendar.io/demos> y también realizamos otra llamada a otra api, que nos muestra un panel de noticias relacionadas con sanidad, con el fin de enriquecer el contenido de la web utilizando la librería axios para manejar las peticiones AJAX: <https://rapidapi.com/search/news> e insertando esta información utilizando la creación de elementos textos y nodos de js



Pedir cita

## Noticias:

Una nueva plataforma protegerá la salud de los migrantes en las Américas — Listin Semanal

Listin Semanal  
un.org

Instrumentan la "Historia de Salud Integrada" en centros de salud y el Hospital de Gobernador Castro

Noticias San Pedro <noreply@blogger.com>  
blogger.com

2015 DECRETO 1443 DE 2014 SG-SST.pptx

libardorajascastro@slideshare.net (libardorajascastro)  
slideshare.net

Vacunación contra la Fiebre Hemorrágica Argentina

Noticias San Pedro <noreply@blogger.com>  
blogger.com

[Editorial] Innovación revolucionaria para entusiastas de la salud y el bienestar

null  
samsung.com

Yaneth Giha, primera mujer en llegar a la Dirección Ejecutiva de FIFARMA

FIFARMA  
prnewswire.com

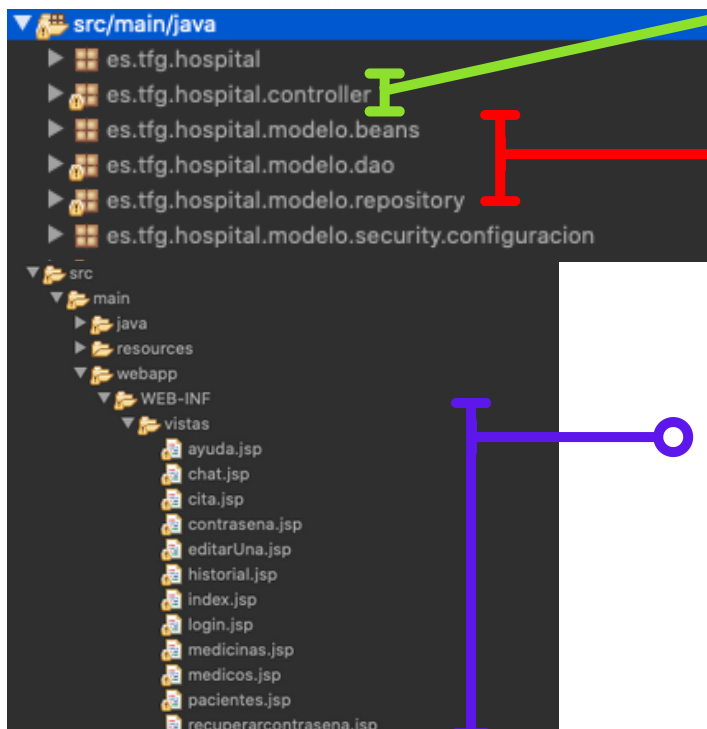
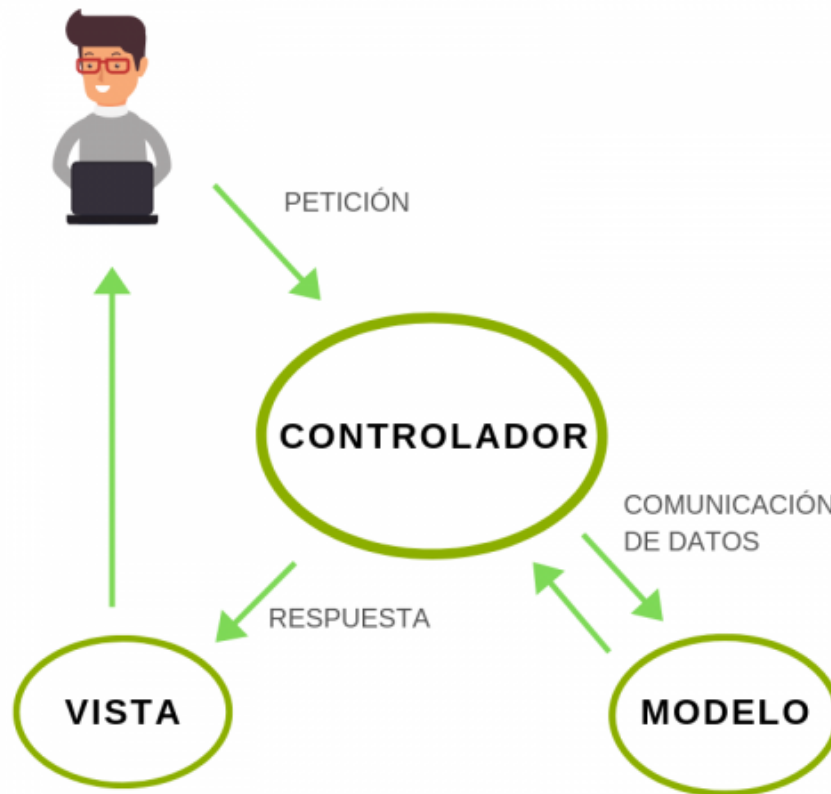
```
C:\> Users > Win10 > Desktop > TFG > TFGFront > index.html
365 today = yyyy + "-" + mm + "-" + dd;
366
367 document.addEventListener("DOMContentLoaded", function () {
368   var calendarEl = document.getElementById("calendar");
369   //----- Resvisar -----
370   var calendar = new FullCalendar.Calendar(calendarEl, {
371     locale: "es",
372     firstDay: 1,
373     headerToolbar: {
374       left: "prev,next today",
375       center: "title",
376       right: ""
377     },
378     //right: 'dayGridMonth,timeGridWeek,timeGridDay,listMonth',
379     //right: "addEventButton",
380   },
381   customButtons: {
382     addEventButton: {
383       text: "Pide tu cita",
384       click: function () {
385         //Idea : añadir la funcion como externa (SE DERIVARA A UN JSP)
386         //Resvisar poner opciones
387         var dateStr = prompt("Enter a date in YYYY-MM-DD format");
388         var titleStr = prompt("Enter a titulo");
389         var horaStr = prompt("Enter a hora");
390         var date = new Date(dateStr + "T" + horaStr + ":00"); // will be in local time
391         var titulo = titleStr + " " + horaStr;
392         location.href = "ir a pagina de cita";
393         if (!isNaN(date.valueOf())) {
394           // valid?
395           calendar.addEvent({
396             title: titulo,
397             start: date,
398             allDay: true,
399           });
400           // Ejecutar solo si viene de poner cita
401           const Toast = Swal.mixin({
402             toast: true,
403             position: "top-end",
404             showConfirmButton: false,
405             timer: 3000,
406             timerProgressBar: true,
407             didOpen: (toast) => {
408               toast.addEventListener("mouseenter", Swal.stopTimer);
409               toast.addEventListener("mouseleave", Swal.resumeTimer);
410             },
411           });
412           // Ejecutar solo si viene de poner cita
413           Toast.fire({
```

```
C:\> Users > Win10 > Desktop > TFG > TFGFront > index.html
541
542 </script>
543 <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
544 <script>
545
546 var noticias = document.querySelector("#noticias");
547 var noticiasTitulo = document.querySelector("#titulonoticas");
548
549 const options = {
550   method: 'GET',
551   url: 'https://free-news.p.rapidapi.com/v1/search',
552   params: {q: 'Salud', lang: 'es', page: '1', page_size: '6'},
553   headers: {
554     'x-rapidapi-key': '98cad11506msd20305fc7490e71p12fb87jsn4u2317af3e7f',
555     'x-rapidapi-host': 'free-news.p.rapidapi.com'
556   }
557 };
558
559 axios.request(options).then(function (response) {
560   console.log(response.data.articles);
561   if(response.status == "error"){
562     noticiasTitulo.innerHTML = " ";
563     const tituloError = document.createElement("h2");
564     const titulo = document.createTextNode("No hay noticias disponibles");
565     tituloError.appendChild(titulo);
566     noticiasTitulo.appendChild(tituloError);
567   }else{
568     var noticiasinfo = response.data.articles;
569     for (let res in noticiasinfo) {
570       console.log(noticiasinfo[res].author);
571       const paratitulo = document.createElement("p");
572       const paratitulo = document.createElement("p");
573       const titulo = document.createTextNode(noticiasinfo[res].title);
574       const subtítulo = document.createTextNode(
575         noticiasinfo[res].author
576       );
577       paratitulo.classList.add("titulonoticas");
578       paratitulo.appendChild(titulo);
579       paratitulo.classList.add("subtitulonoticas");
580       paratitulo.appendChild(subtítulo);
581       const small = document.createElement("small");
582       const autor = document.createTextNode(noticiasinfo[res].clean_url);
583       small.appendChild(autor);
584       const bloqueNoticia = document.createElement("article");
585       bloqueNoticia.appendChild(paratitulo);
586       bloqueNoticia.appendChild(paratitulo);
587       bloqueNoticia.appendChild(small);
588       noticias.appendChild(bloqueNoticia);
```

# 10 Modelo, Vista, Controlador (MVC)

El Modelo-Vista-Controlador es una arquitectura software.

Se encarga de recibir en el controlador la petición enviada por usuario, una vez recibida la petición el controlador analizara que tiene que hacer y pedirá tanto los datos como las funciones que necesita de los modelo para así poder procesar la información y pasársela a la vista que esta será recibida por el cliente, el cual podrá visualizarlo en su buscador



El controlador se encuentra en la ruta de directorio: **es.tfg.hospital.controller**

El modelo se compone de beans , dao y el repository, estos directorios se encuentran en la ruta: **es.tfg.hospital.modelo.beans, es.tfg.hospital.modelo.dao, es.tfg.hospital.modelo.repository.**

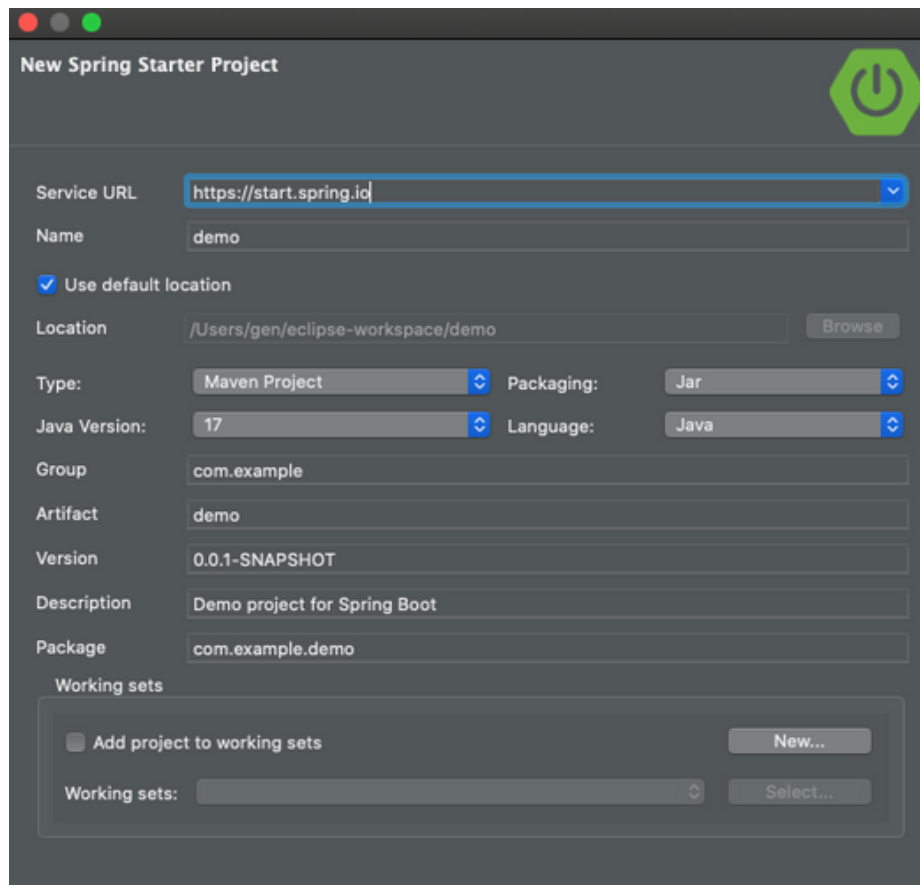
Las vistas se componen de archivos .jsp y se encuentran en el directorio: **src.main.webapp.WEB-INF.vistas**



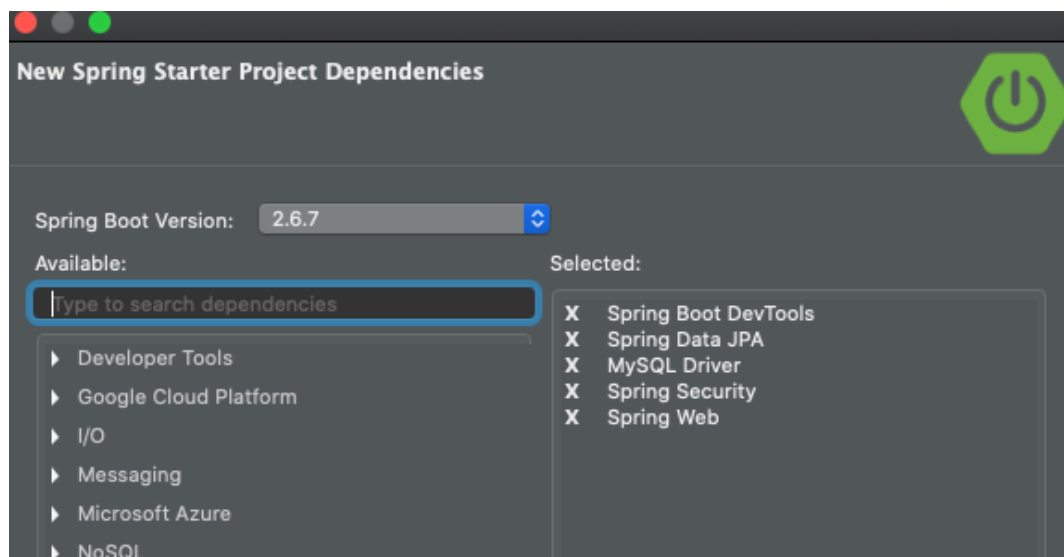
# Configuración del proyecto

Para generar el proyecto seguiremos los siguientes pasos:

- File -> New Spring starter project



- Elegimos las dependencias de Spring que vamos a utilizar



Configuramos el archivo application.properties (**src/main/resources**)

```
1 server.port=8086
2 spring.mvc.view.prefix=/WEB-INF/vistas/
3 spring.mvc.view.suffix=.jsp
4
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6 spring.datasource.url=jdbc:mysql://localhost:3306/hospital_bbdd?serverTimezone=UTC
7 spring.datasource.username=umedico
8 spring.datasource.password=umedico
9
10 spring.jpa.generate-ddl=false
11 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
12 spring.jpa.show-sql=true
13 spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
14
```

Aquí configuramos el puerto , la conexión a la base de datos y la ruta donde se encuentran nuestras vistas.

Botón derecho sobre el propio proyecto:

- Properties -> Project Facets->Convert faceted from ->JPA
- Further configuration available

## JPA

- Plataforma Generic 2.2
- Conexión -> Añadimos conexión, especificamos my sql
- Metemos el driver mysql Conector J 8.0 (jar)
- Propiedades:
  - jdbc:mysql://localhost:3306/base\_datos?Timezone=UTC
  - base\_datos
  - com.mysql.cj.jdbc.Driver
  - Contraseña
  - UserID
  - Test conexión

Botón derecho proyecto->JPA Tools -> new JPA Entities from Tables

- Seleccionar conexión
- Seleccionamos las tablas
- Elegir las asociaciones
- Si es una referencia a una colección de objetos, generar como cascade persist

En el archivo pom.xml también añadiremos otras dependencias necesarias, como por ejemplo el taglib.

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-taglibs</artifactId>
  <version>5.4.5</version>
</dependency>
```

En el directorio dentro de la ruta **src/main/resources static** pondremos nuestros archivos css y js e imágenes y contenido estático.



# 11 Modelo DAO

El Modelo DAO (Data Access Object) es lo que nos permite acceder a los datos utilizando métodos definidos por el propio DAO

En los archivos Dao se tiene que poner @Service arriba de la clase esto se usa para construir una clase de Servicio que habitualmente se conecta a varios repositorios.

Ejemplos casos Modelo DAO en el proyecto:

```
CitaDaoImplMy8.java
1 package es.tfg.hospital.modelo.dao;
2
3 import java.util.List;
4
5
6
7
8
9
10
11 @Service
12 public class CitaDaoImplMy8 implements IntCitaDao{
13
14     @Autowired
15     private CitaRepo citarepo;
16
17     /**
18      * Busca las citas a raiz del dni
19      */
20     @Override
21     public List<Cita> buscarCitas(String dni) {
22         // TODO Auto-generated method stub
23         return citarepo.findCitabyUsuario(dni);
24     }
25
26     /**
27      * Saca una lista de todas las citas
28      */
29     @Override
30     public List<Cita> buscarTodos() {
31         // TODO Auto-generated method stub
32         return citarepo.findAll();
33     }
34
35     @Override
36     public List<Cita> buscarCitaPorNombre(String nombrePaciente) {
37         return citarepo.buscarCitaPorNombre(nombrePaciente);
38     }
39 }
```

```
es.tfg.hospital.modelo.dao
├── CitaDaoImplMy8.java
├── ComentarioDaoImplMy8.java
├── DiagnosticoDaoImplMy8.java
├── HistorialClinicoDaoImplMy8.java
└── InformacionDaoImplMy8.java
```

En esta parte del código lo que hacemos son diferentes métodos para poder obtener los datos que queremos tanto la lista completa como solo uno.

Esto se hace llamando a los métodos de CitaRepo que es un archivo que manda las sentencias SQL a la base de dato con el fin de obtenerlos

```
CitaRepo.java
1 package es.tfg.hospital.modelo.repository;
2
3
4 import java.util.List;
5
6
7
8
9
10 public interface CitaRepo extends JpaRepository<Cita, Integer>{
11
12     @Query("select c from Cita c where c.usuario.dni = ?1 ORDER BY c.fechaCita DESC")
13     List<Cita> findCitabyUsuario(String dni);
14
15     @Query("select c from Cita c where c.idCita = ?1")
16     Cita findUnaCita(Integer idCita);
17
18     @Query("select c from Cita c where c.nombreMedico = ?1")
19     List<Cita> findCitabyMedico(String nombreMedico);
20
21     @Query("select c from Cita c where c.nombrePaciente like %?1%")
22     public List<Cita> buscarCitaPorNombre(String nombrePaciente);
23 }
24
```

○ Sentencia SQL

○ Metodo

# 12 Vista

La Vista es el archivo donde se recibe la información proveniente del Controlador para que este archivo se vea bien se utiliza

- Html
- Css
- JavaScript o para el pdf
- Taglib (son unas etiquetas que permiten realizar algunas acciones como introducir datos de una lista)

En la foto de la izquierda se hace la union del html y del taglib

```
174• <table>
175• <thead>
176• <tr>
177• <td>Nombre</td>
178• <td>Cantidad</td>
179• <td>Fecha Inicio</td>
180• <td>Fecha Final</td>
181• <td>Tipo</td>
182• </tr>
183• </thead>
184• <tbody>
185• <c:forEach var="ele" items="${listaMedicina}">
186• <tr>
187• <td>${ele.nombreMed}</td>
188• <td>${ele.cantidad}</td>
189• <td>${ele.fechaInicio}</td>
190• <td>${ele.fechaFin}</td>
191• <td>${ele.tipo.descripcion}</td>
192• </tr>
193• </c:forEach>
194• </tbody>
195• </table>
196• </div>
```

Aqui podemos ver como queda el resultado recalcar que el estilo css se localiza en otro archivo

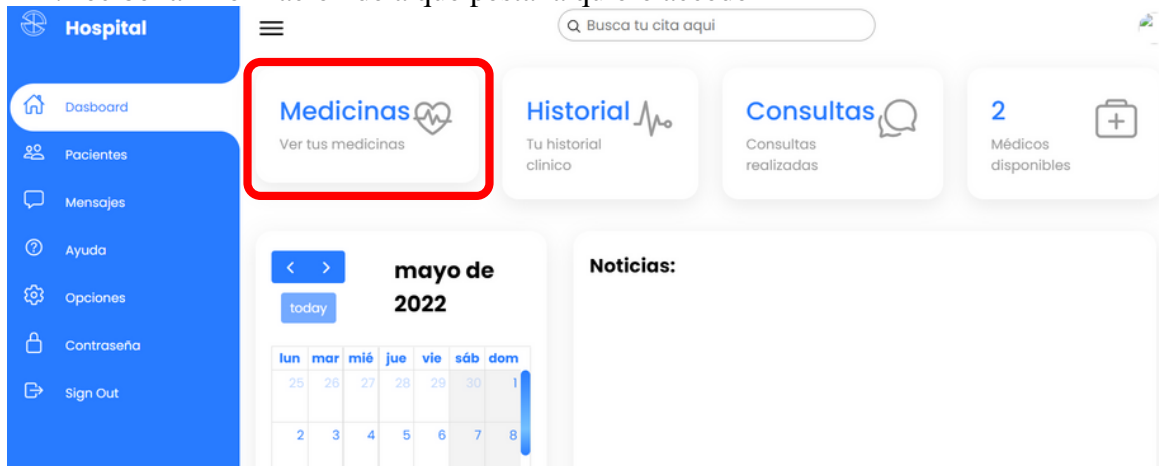
Tus medicinas				
Nombre	Cantidad	Fecha Inicio	Fecha Final	Tipo
DALSY	12	2022-06-12	2023-04-12	ANALGESICO
DALSY	12	2022-06-12	2023-04-12	ANALGESICO
DALSY	12	2022-06-12	2023-04-12	ANALGESICO

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>
```

# 13 Controllador

El Controlador es el cerebro de todo el programa, este se encarga de unir las peticiones del usuario para así poder tramitar la información de tal manera, que si el usuario quiere acceder a una ventana el Controllador

1. Recibe la información de a que pestaña quiere acceder



2. Realiza la petición de los datos necesarios, los pasa en una etiqueta para que el jsp los inserte y devuelve el archivo .jsp

```
526  
527 @GetMapping("/medicinas")  
528 public String mostrarMedicinas(Model model, HttpSession misesion) {  
529     List<Medicina> listaMedicina= mdao.buscarMedicinasUsuario((String) misesion.getAttribute("dni"));  
530     model.addAttribute("listaMedicina", listaMedicina);  
531     return "medicinas";  
532 }
```

Petición de datos

Nombre del archivo .jsp

Etiqueta de los datos

3. Así es como lo visualizaríamos



Nombre	Cantidad	Fecha Inicio	Fecha Final	Tipo
DALSY	12	2022-06-12	2023-04-12	ANALGESICO
DALSY	12	2022-06-12	2023-04-12	ANALGESICO
DALSY	12	2022-06-12	2023-04-12	ANALGESICO

# 14 Apache Tomcat

Apache Tomcat es un contenedor de servlets que se puede usar para compilar y ejecutar aplicaciones web realizadas en Java. Implementa y da soporte tanto a servlets como a páginas JSP (Java Server Pages) o Java Sockets. Además, Tomcat es compatible con otras tecnologías como Java Expression Language y Java WebSocket, del ecosistema Java.

Estructura de directorios:

- **bin** - arranque, cierre, y otros scripts y ejecutables.
- **common** - clases comunes.
- **conf** - ficheros XML y los correspondientes DTD
- **logs** - logs de Catalina y de las aplicaciones
- **server** - clases utilizadas solamente por Catalina
- **shared** - clases compartidas por las aplicaciones web.
- **webapps** - directorio que contiene las aplicaciones web.
- **work** - almacenamiento temporal de ficheros y directorio

## Despliegue de la aplicación

En caso de tener que desplegar la aplicación solo tendremos que seguir los siguientes pasos

Para realizar el despliegue de la aplicación se podía utilizar Heroku que es una plataforma como servicio de computación en la Nube que soporta distintos lenguajes de programación.

Para ello necesitaremos:

- Un proyecto Spring Boot existente con base de datos MySQL
- Una cuenta verificada de Heroku
- Heroku CLI instalado
- Git instalado
- Servidor comunitario MySQL con cliente de línea de comandos MySQL

### 1- Implementar un proyecto Spring Boot en Heroku

- Abra un nuevo símbolo del sistema en Windows o terminal en Mac o Linux.
- Cambie el directorio actual al directorio raíz de su proyecto
- Cree un nuevo repositorio local de Git: **git init**

```
@)git init
Initialized empty Git repository in G:/Heroku/SpringBootInMemoryFormAuth/.git/
```

- Agregue los archivos del proyecto al índice del repositorio local: **git add**.
- Confirme los archivos en el repositorio local de Git: **git commit -m "first commit"**

```
@)git commit -m "commit to deploy"
[master (root-commit) 906adc1] commit to deploy
20 files changed, 939 insertions(+)
```

- Iniciar sesión en Heroku: **heroku login**

```
@)heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/4cce92c4-
2gDbQAAAA4xMTYuMTEwLjE3LjIwMG4GAGjfdC98AWIAAVGA.BU38ZbOpCZxgSgFp_cyCI_N0c
Logging in... done
Logged in as your email address
```

- Crea una nueva aplicación de Heroku: **heroku create nombre-app**

```
@)heroku create my-spring-app-1
Creating my-spring-app-1... done
https://my-spring-app-1.herokuapp.com/ | https://git.heroku.com/my-spring-app-1.git
```

- Implemente el proyecto a través de Git: **git push heroku master**

```
remote:      Default types for buildpack -> web
remote:
remote: -----> Compressing...
remote:      Done: 69.7M
remote: -----> Launching...
remote:      Released v3
remote:      https://my-spring-app-1.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/my-spring-app-1.git
 * [new branch]      master -> master
```

## 2- Instalar el complemento ClearDB MySQL

Utilizaremos estos comandos para trasladar nuestra base del mySQLWorkbench

```
heroku addons:create cleardb:ignite -a nombre-app
```

Una vez instalado, este complemento crea una nueva variable de configuración para su aplicación, llamada `CLEARDB_DATABASE_URL`, que almacena información de conexión a la base de datos. Es algo como esto:

```
CLEARDB_DATABASE_URL: mysql://username:password@hostname/schema_name
```

## 3- Exporte la base de datos MySQL local a un archivo SQL

```
mysqldump -u nombre_de_usuario -p nombre_esquema > mysql_dump_file.sql
```

Este comando exporta el esquema de base de datos dado a un archivo `.sql`. Reemplace nombre de usuario, `schema_name` y `mysql_dump_file` por valores reales. Tenga en cuenta que debe ingresar la contraseña de MySQL para ejecutar este comando.

## 4- Importar archivo SQL al servidor MySQL remoto

A continuación, le mostraré cómo conectarse al servidor MySQL remoto para importar el archivo `.sql`. Escriba este comando para ver las variables de configuración de su aplicación:

```
heroku config -a nombre_de_tu_aplicación
```

Extraiga el nombre de usuario, la contraseña, el nombre de host y el nombre del esquema de la variable de configuración `CLEARDB_DATABASE_URL` . Y escriba el siguiente comando para conectarse al servidor MySQL remoto con el programa cliente MySQL Command Line:

```
mysql -u nombre_de_usuario -p -h nombre_de_host
```

Es necesario introducir la contraseña para iniciar sesión. Luego, en el indicador del cliente de línea de comandos de MySQL, escriba este comando para conectarse al esquema de la base de datos creado por el complemento ClearDB:

```
connect schema_name ;
```

A continuación, escriba el siguiente comando para obtener el juego de caracteres y la intercalación de la base de datos actual:

```
select @@character_set_database, @@collation_database
```

Luego reemplace todos los valores del conjunto de caracteres por utf8; cotejar/cotejar por utf8\_general\_ci.

Guarde el archivo .sql. Luego escriba exit para salir del programa cliente MySQL Command Line.

Ahora escriba este comando para importar el archivo .sql al servidor MySQL remoto:

```
mysql -u username -p -h hostname < mysql_dump_file.sql
```

Debe ingresar la contraseña cuando se le solicite. Y espere un momento hasta que el comando termine en silencio. Luego, puede conectarse nuevamente al servidor MySQL remoto y escribir los siguientes comandos para verificar que los datos se importaron correctamente:

```
connect schema_name;  
show tables;  
desc table table_name;  
select * from table_name;
```

5- Acceda a la base de datos MySQL en la aplicación Spring Boot

Tenga en cuenta que si su aplicación implementada es una aplicación Spring Boot, el complemento ClearDB MySQL también agregará las siguientes variables de entorno:

```
SPRING_DATASOURCE_URL  
SPRING_DATASOURCE_NOMBRE DE USUARIO  
SPRING_DATASOURCE_PASSWORD
```

# 6 Conclusiones

Como conclusión final en el proyecto se observa que el uso de la aplicación web es más que intuitiva y accesible que las existentes citadas anterior mente mejorando su funcionalidad y la experiencia de usuario.

En cuanto a los resultados se han creado los métodos necesarios para cumplir las funciones que ha de tener la aplicación, tambien añadimos nuevas mejoras sobre la idea original como puede ser la exportación a pdf, el sistema de chat entre usuario o la capacidad de ver si los usuarios están conectados o no.

En cuanto a las posibles mejoras del proyecto podrían ser tales como una optimización del chat para que recargue la conversación en el chat o incluir y mejorar querys de búsqueda de citas y listas de usuario o médico.

## Visuales

A login form mockup centered on a blue background. The form is white with a blue border. It has a title 'Login' in blue. Below the title are two input fields: the first contains '123' and the second contains three dots. Below the input fields is a blue button with the text 'Entrar'. Below the button are three lines of text: '¿Se te ha olvidado la contraseña?' followed by a blue link 'Recuperar contraseña', '¿No tienes una cuenta?' followed by a blue link 'Regístrate ahora'.

Hospital

Dashboard

Mensajes

Ayuda

Opciones

Contraseña

Sign Out

Medicinas

Ver tus medicinas

Historial

Tu historial clínico

Consultas

Consultas realizadas

2

Médicos disponibles

<

>

today

mayo de 2022

lun	mar	mié	jue	vie	sáb	dom
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22

12 Cita

12 Cita

8:30 C

8:30 C

Pedir cita

Noticias:

Salud urbana: el barrio determina tu bienestar o tus enfermedades

Manuel Franco

elpais.com

Instrumentan la "Historia de Salud Integrada" en centros de salud y el Hospital de Gobernador Castro

Noticias San Pedro <noreply@blogger.com>

blogger.com

Una nueva plataforma protegerá la salud de los migrantes en las Américas — Listin Semanal

Listin Semanal

un.org

Terrel Guerrero Karina.pdf

KarinaFiorelaTERREL@slideshare.net(KarinaFiorelaTERREL)

slideshare.net

Anatomía.pptx

yrianamartinez@slideshare.net(yrianamartinez)

slideshare.net

Nace EL PAÍS Salud y Bienestar

El País

elpais.com

Ultimas citas

Fecha	Dirección	Estado	Ver todas
2022-06-11	Dire 1	Cancelado	Ver mas
2022-05-28	Dirección1	Pendiente	Ver mas

Conectados

jaume

juame

maria

Hospital

Dashboard

Pacientes

Mensajes

Ayuda

Opciones

Contraseña


Sign Out

Todas las citas

Nombre Paciente	Fecha	Dirección	Estado	Información
rodrigo	2022-04-28	Direcion rontonda 2	Realizado	Ver mas Editar
rodrigo	2022-06-12	Direccion1	Cancelado	Ver mas Editar
jaume	2022-04-23	Direcion rontonda 1	Pendiente	Ver mas Editar
123	2022-05-12	Dire 1	Realizado	Ver mas Editar
123	2022-06-11	Dire 1	Cancelado	Ver mas Editar
123	2022-04-12	Dire 2	Realizado	Ver mas Editar
123	2022-05-10	calle 1º 3d	Cancelado	Ver mas Editar
nuevo	2022-05-20	Direccion2	Realizado	Ver mas Editar
nuevo2	2022-05-28	Direccion1	Realizado	Ver mas Editar
nuevo2	2022-05-28	Direccion1	Cancelado	Ver mas Editar
123	2022-05-27	Direccion1	Pendiente	Ver mas Editar
123	2022-05-21	Direccion1	Pendiente	Ver mas Editar
123	2022-05-21	Direccion1	Pendiente	Ver mas Editar
123	2022-05-28	Direccion1	Pendiente	Ver mas Editar

38





**Pide tu cita**


Fecha

Hora


Centro

Motivo


Médico




**Chats**




jaume  
juame




maria  
maria



pepe  
pepe

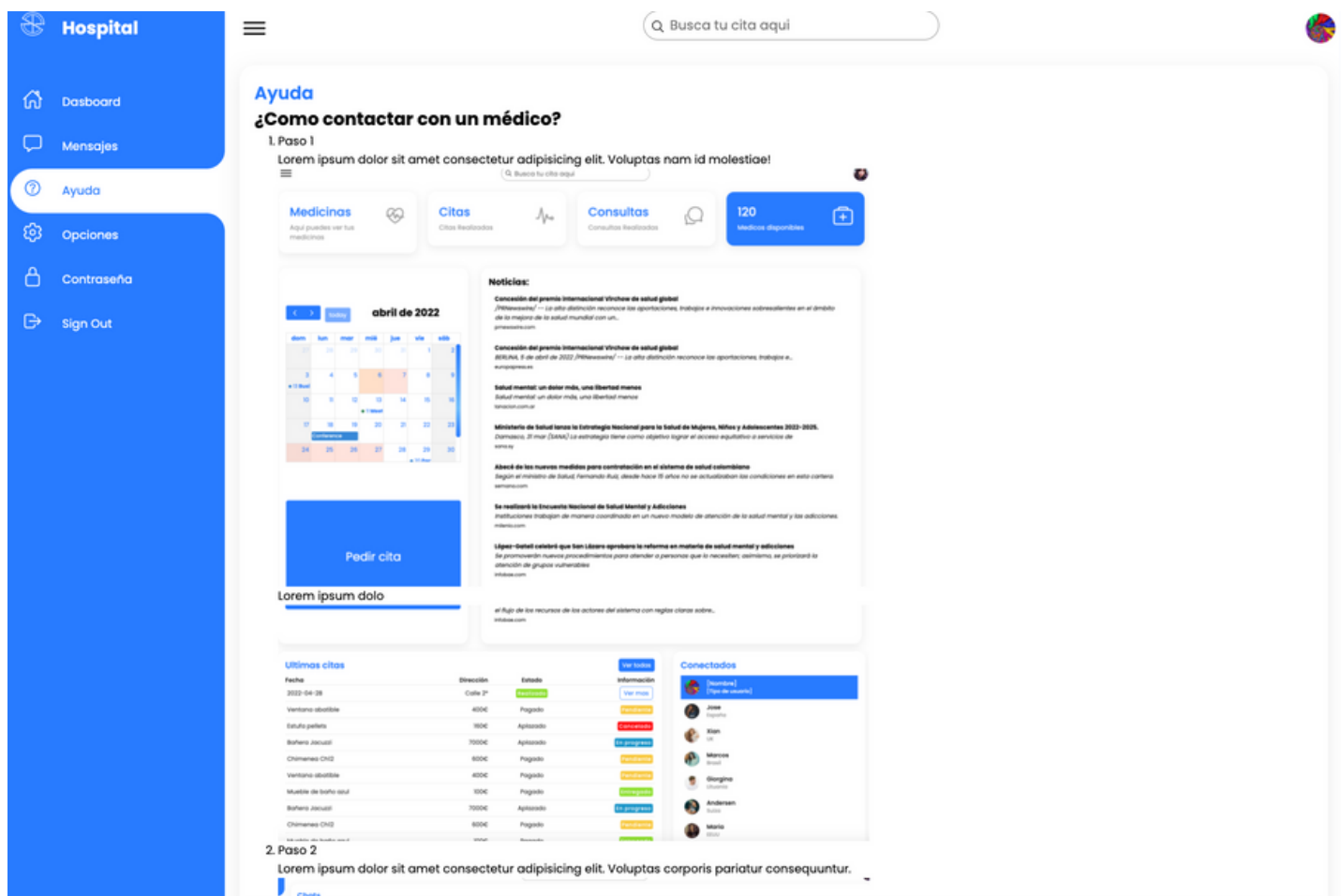
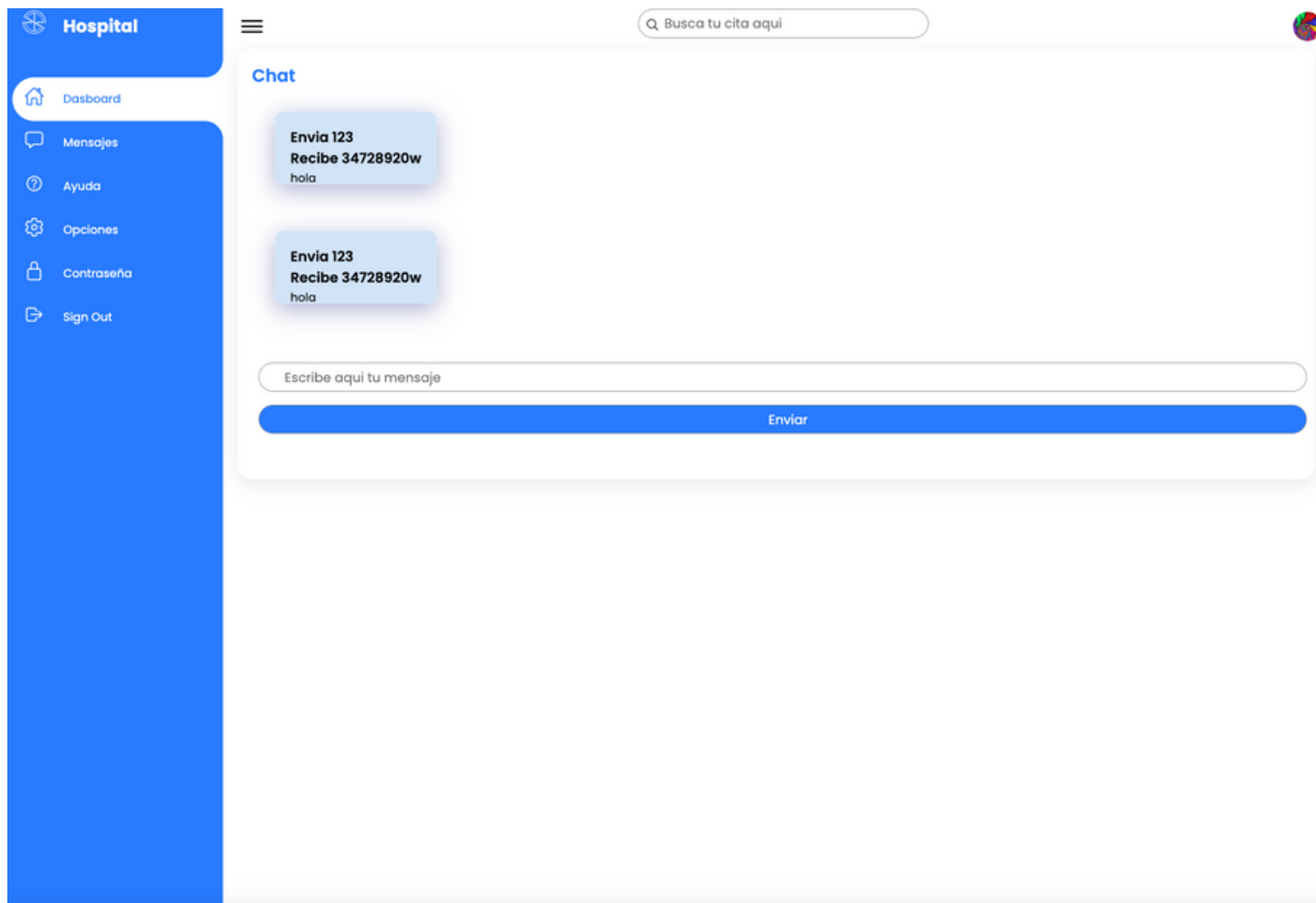



rodrigo  
sendino



nuevo  
nuevo

39



Hospital

Dashboard

Mensajes


Ayuda

Opciones

Contraseña

Sign Out

Busca tu cita aqui



Datos usuario

DNI123

Email123@123.123

Nombre123

Apellido123

Domicilio123

Telefono123

Imagen URLhttps://empresas.blogthinkbig.com/wp-content/uploads/2019/11/imagen3-245003649.jpg

Actualizar


Datos fisicos

Peso23

Altura45

Edad36

Sexo♂

Hospital

Dashboard

Mensajes


Ayuda

Opciones

Contraseña

Sign Out

Busca tu cita aqui



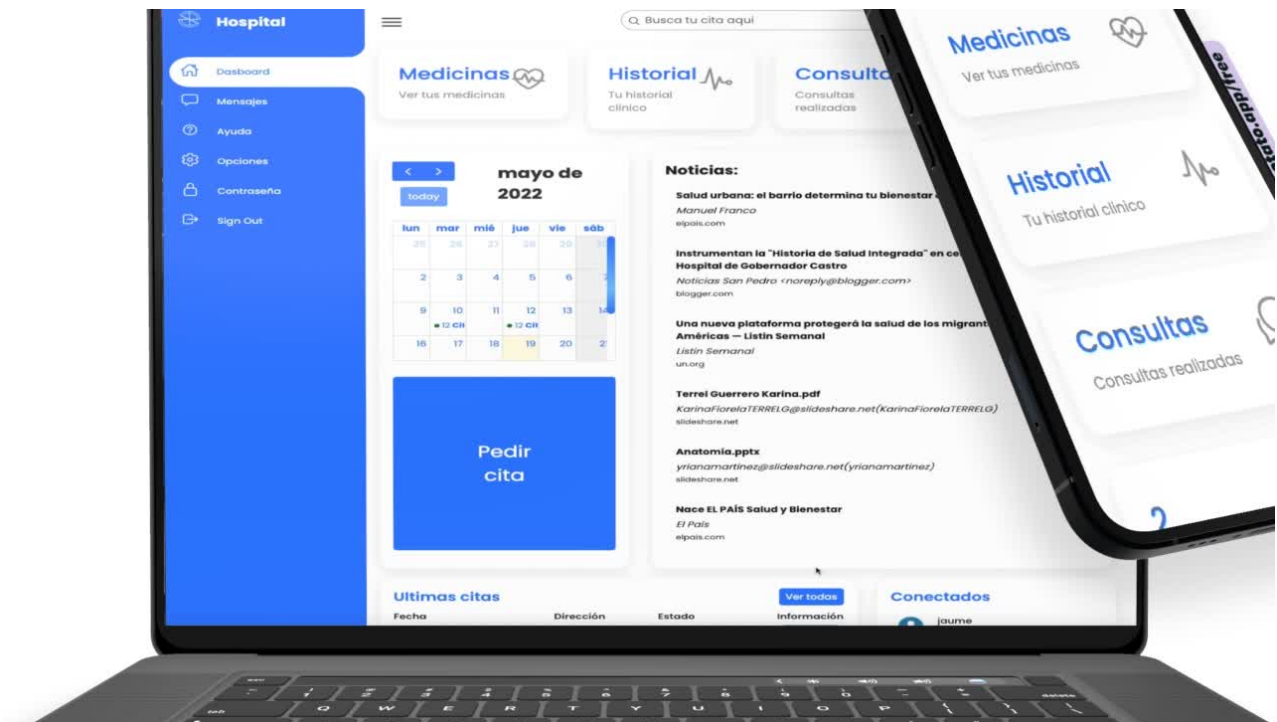
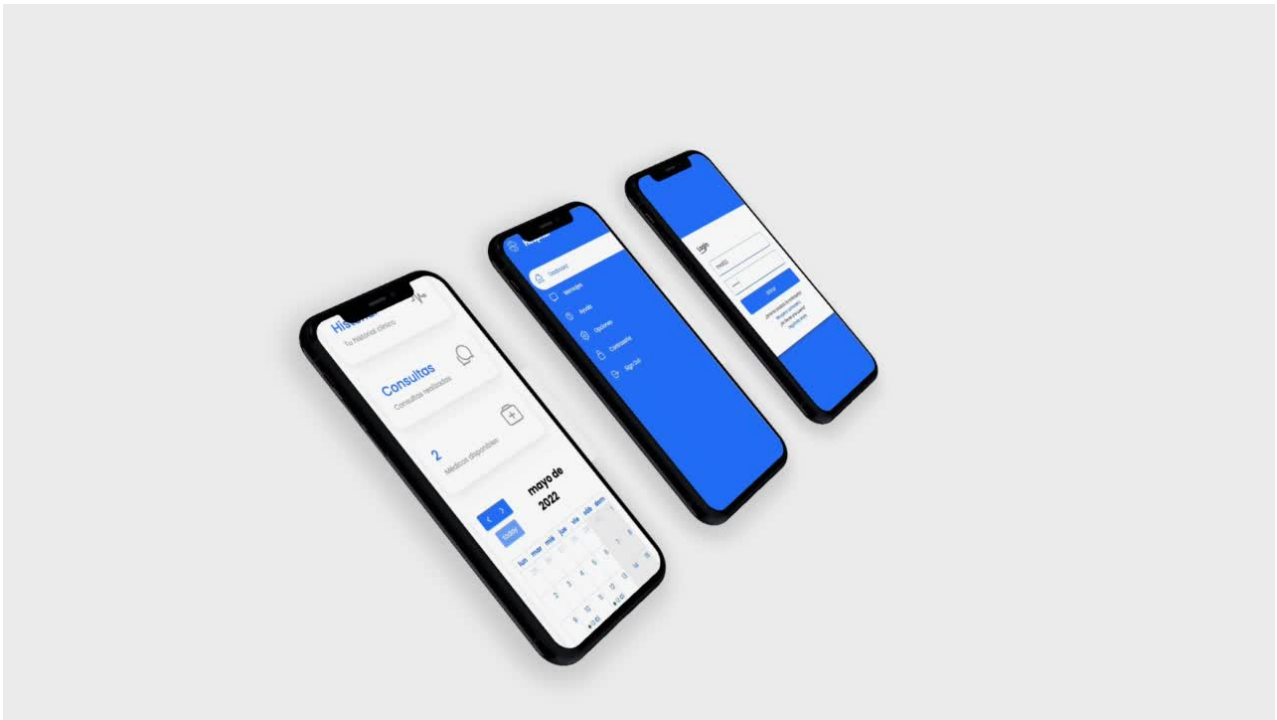
Cambiar contraseña

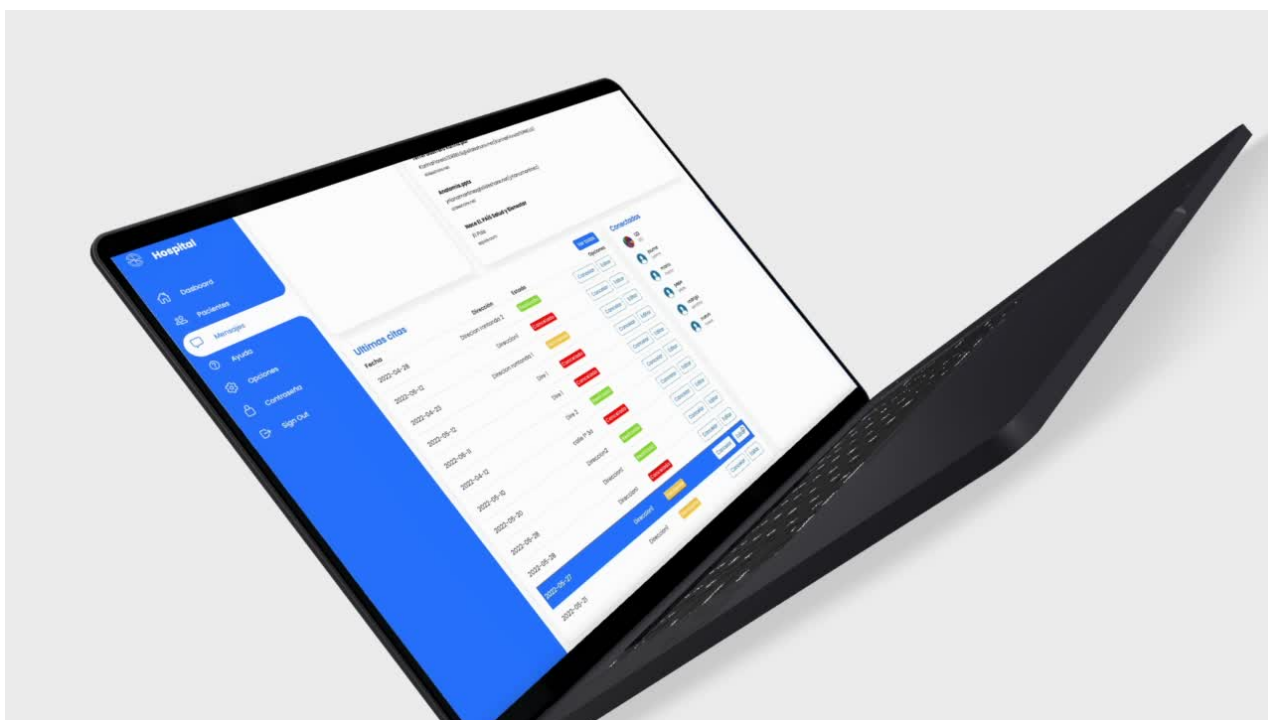
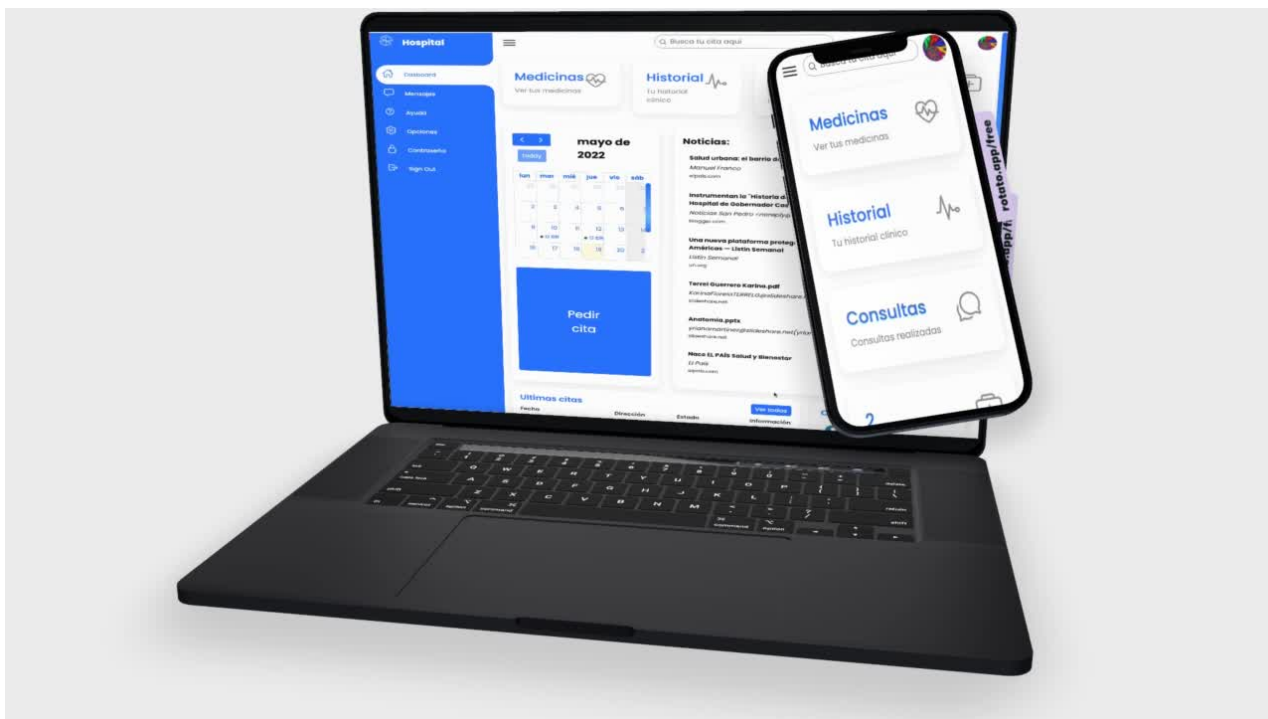
Contraseña actual

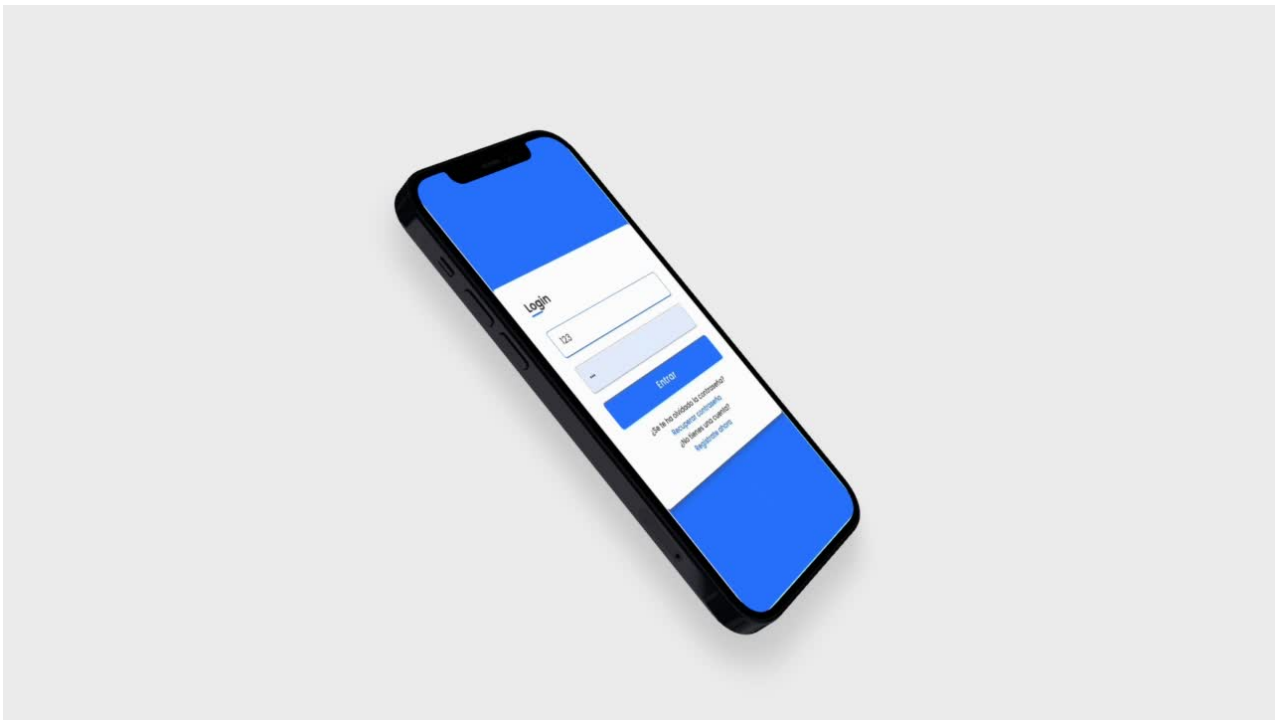
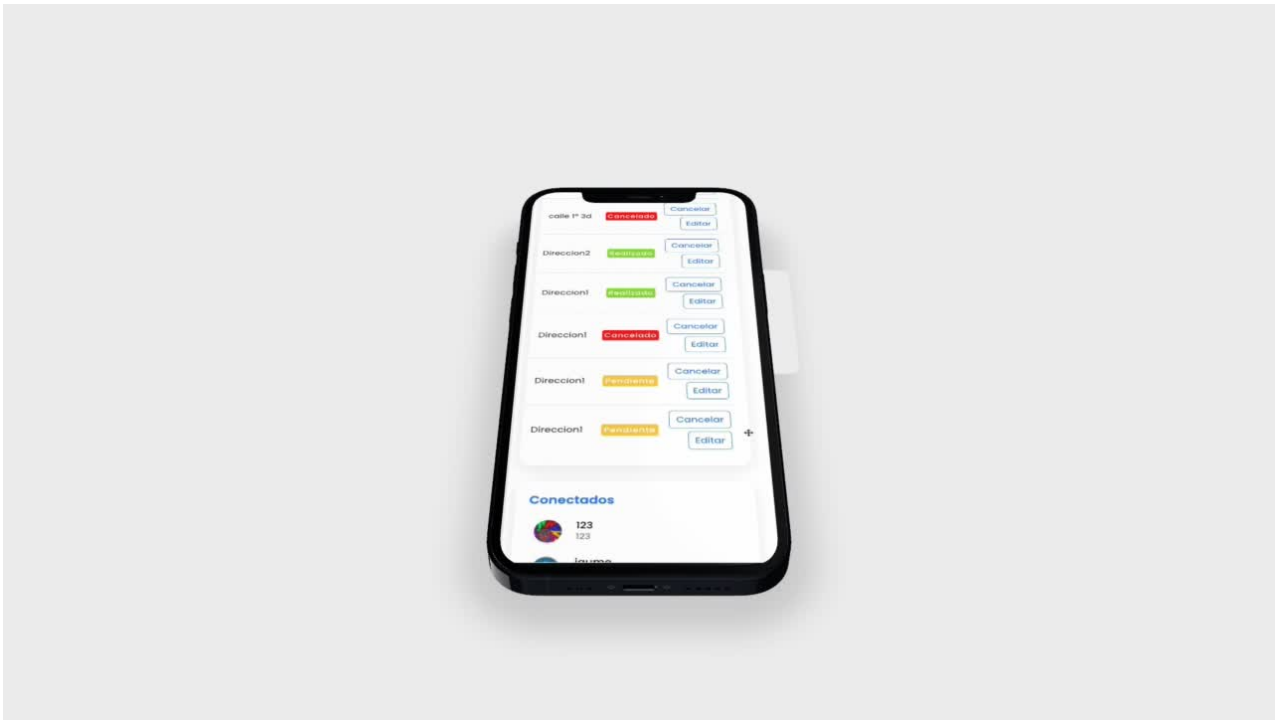
Nueva contraseña

Respite la nueva contraseña

Cambiar Contraseña







# 7 Agradecimientos

En estos 2 años nos hemos esforzado todos para conseguir una formación de calidad pese a los cambios y la incertidumbre.

Queríamos agradecer por el apoyo a las siguientes personas :

A Ascensión Blázquez y Daniel Díaz.

A los profesores:

- Raúl Salgado Vilas
- Félix de Pablo
- Maria de Gracia Carrión
- Alfredo Cordero
- Tomás Escudero Delgado

y a nuestra familia y amigos.

# 8 Anexos

Para testear el proyecto navegamos a el repositorio:

[https://github.com/RodrigoSendinoSanz/01\\_Hospital\\_TFG](https://github.com/RodrigoSendinoSanz/01_Hospital_TFG)

Para crear la base de datos utilizaremos el archivo que se encuentra en la carpeta [https://github.com/RodrigoSendinoSanz/01\\_Hospital\\_TFG/blob/main/MemoriaTFG/Parte-Back/CodigoBBDD.sql](https://github.com/RodrigoSendinoSanz/01_Hospital_TFG/blob/main/MemoriaTFG/Parte-Back/CodigoBBDD.sql)

- Creamos la base de datos con el usuario y contraseña **umedico**
- Importamos la carpeta descargada del github en el programa eclipse
- Una vez totalmente importado hacemos click en restart
- y en cualquier navegador escribir en su barra de búsqueda: **http://localhost:8086/**





# 9 Bibliografía

## [Textos]

Los textos se han basado en su mayoría en la plataforma de educación de unir y relación propia en base a la formación adquirida.

Aun que también se ha obtenido información en Wikipedia, google y codejava.net

## [Imágenes]

La mayoría de las imágenes presentes en el proyecto son de google imágenes, recortes del mismo proyecto o imágenes de canva.

## [Diseño web]

Inspiración YouTube OnlineTutorials4Designers