

Proyecto de empresa de reparaciones

Contenido

| | |
|---|----|
| ¿De qué trata la aplicación? | 2 |
| ¿Por qué una aplicación de reparaciones? | 2 |
| Diagrama de casos de uso. | 2 |
| ● 1. enviar problema | 3 |
| ● 2. consultar incidencia | 3 |
| ● 3. trabajar incidencia | 4 |
| ● 4. poner en cola la incidencia | 4 |
| ● 5. consultar estado de incidencia..... | 4 |
| ● 6. finalizar incidencia | 5 |
| ● 7. encuesta de satisfacción | 5 |
| Diagrama de clases..... | 6 |
| Diagrama entidad relación..... | 6 |
| Relación de tablas implementadas | 7 |
| Descripción de los campos..... | 7 |
| Implementación del código en la API Rest | 8 |
| Tecnología utilizada..... | 9 |
| Tabla con los tipos de navegación utilizados, las razones para haberlos usado, y en qué componente/clase/método se ha hecho uso de ellos. | 10 |
| Tabla con los tipos de Autenticación implementados, y el componente/clase/método en los que se implementa. | 11 |

¿De qué trata la aplicación?

La aplicación trata sobre una empresa de servicio de reparaciones a distancia o con cita previa para atención en el edificio de la empresa, en el cual el cliente mediante la aplicación móvil manda su caso específico junto con la categoría de su problema (hardware o software) y en caso de que conozca a un trabajador y quiera que él sea el que le atienda puede añadirlo junto con una nota de lo que le pasa. Este mensaje lo obtendrían los trabajadores que tengan asociados la misma categoría del caso (hardware o software).

Los trabajadores pueden o solo dominar hardware, software o ambos, lo que será especificado en "especialidad".

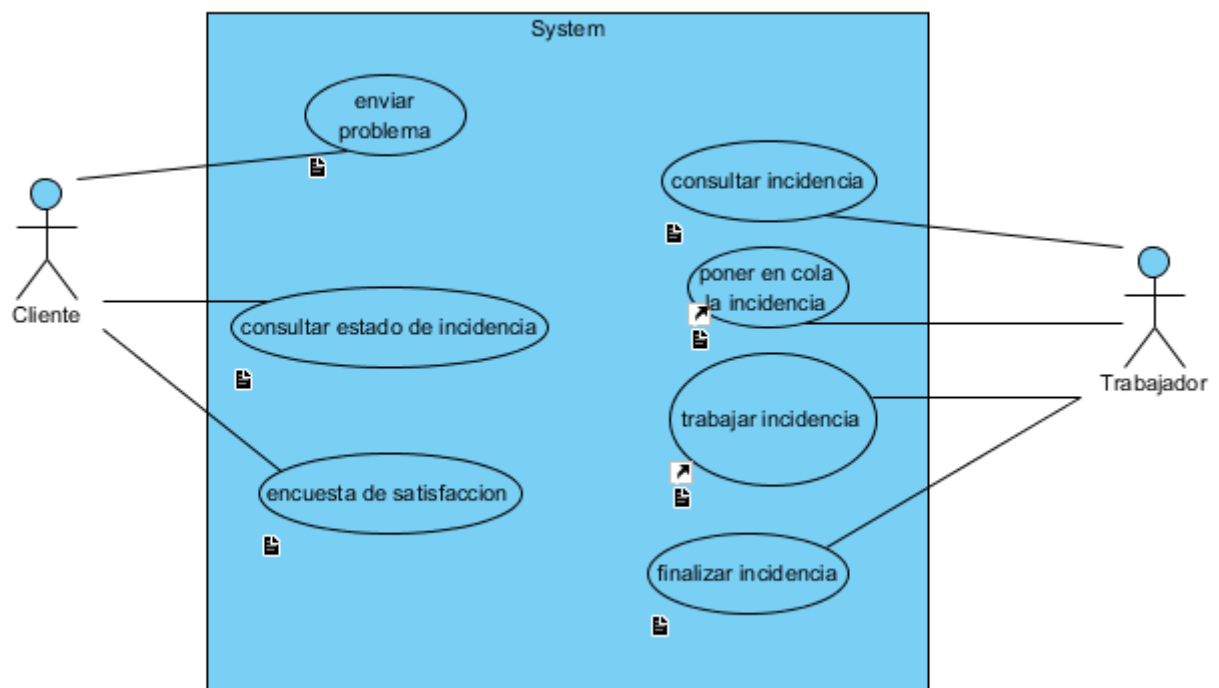
Los trabajadores no están disponibles si ya están efectuando otro trabajo por telefonía, en caso de que ningún trabajador esté disponible, el cliente tendrá un aviso de que ahora mismo no puede atenderle nadie "enCola".

Tras arreglar o no el problema, el cliente tendría en la aplicación una ventana de feedback del cliente con respecto al servicio, en el cual le pediría que a poder ser de una evaluación del 1 al 5 y si quiere que escriba una pequeña nota de cómo le ha parecido.

¿Por qué una aplicación de reparaciones?

Porque creo que es un buen tema para tratar los campos que hay que implementar en el proyecto al tener clientes, trabajadores y una lista de tareas junto con el feedback del cliente al finalizar el trabajo.

Diagrama de casos de uso.




| | Name | ID | Primary Actors | Task Pool |
|--|--|------|----------------|-----------|
| | enviar problema | UC01 | Cliente | |
| | consultar incidencia | UC02 | Trabajador | |
| | poner en cola la incidencia | UC03 | Trabajador | |
| | consultar estado de incidencia | UC06 | Cliente | |
| | trabajar incidencia | UC04 | Trabajador | |
| | finalizar incidencia | UC05 | Trabajador | |
| | encuesta de satisfaccion | UC07 | Cliente | |

1. enviar problema

ID: UC01


Los clientes mediante la aplicación mandan su caso mientras especifican si el problema es de hardware o software.

| | |
|------------------------------|---|
| Primary Actors |  Cliente |
| Level | N/A |
| Complexity | N/A |
| Use Case Status | N/A |
| Implementation Status | N/A |
| Preconditions | Tener un problema con el ordenador. |
| Post-conditions | No saber arreglar el problema. |
| Author | N/A |
| Assumptions | N/A |

2. consultar incidencia

ID: UC02


El trabajador podrá ver si hay alguna incidencia para ponerse a trabajar en ella.

| | |
|------------------------------|--|
| Primary Actors |  Trabajador |
| Level | N/A |
| Complexity | N/A |
| Use Case Status | N/A |
| Implementation Status | N/A |
| Preconditions | Tener incidencias a consultar. |
| Post-conditions | N/A |
| Author | N/A |
| Assumptions | N/A |

3. trabajar incidencia

ID: UC04


El trabajador se pondrá con un problema e indicará si está en proceso.

| | |
|-----------------------|--|
| Primary Actors |  Trabajador |
| Level | N/A |
| Complexity | N/A |
| Use Case Status | N/A |
| Implementation Status | N/A |
| Preconditions | Tener incidencias para trabajar. |
| Post-conditions | N/A |
| Author | N/A |
| Assumptions | N/A |

4. poner en cola la incidencia

ID: UC03


Dependiendo de si está trabajando en otro problema o incidencia, el trabajador podrá poner en cola otros trabajos para efectuar su mantenimiento en otro momento.

| | |
|-----------------------|--|
| Primary Actors |  Trabajador |
| Level | N/A |
| Complexity | N/A |
| Use Case Status | N/A |
| Implementation Status | N/A |
| Preconditions | Tener incidencias que poner en cola. |
| Post-conditions | N/A |
| Author | N/A |
| Assumptions | N/A |

5. consultar estado de incidencia

ID: UC06

El cliente tendrá permitido en todo momento saber si su incidencia está en espera, en proceso o finalizada.
También podrá cancelar su incidencia en el caso de que todavía esté en espera.


| | |
|-----------------------|---|
| Primary Actors |  Cliente |
| Level | N/A |
| Complexity | N/A |
| Use Case Status | N/A |
| Implementation Status | N/A |

| | |
|------------------------|-------------------------------|
| Preconditions | Haber enviado una incidencia. |
| Post-conditions | N/A |
| Author | N/A |
| Assumptions | N/A |

6. finalizar incidencia

ID: UC05

El trabajador pone la incidencia como completada, ya sea porque el cliente canceló la incidencia a mitad del trabajo o porque el trabajador pudo acabar con ella.

| | |
|------------------------------|--|
| Primary Actors |  Trabajador |
| Level | N/A |
| Complexity | N/A |
| Use Case Status | N/A |
| Implementation Status | N/A |
| Preconditions | Haber trabajado en una incidencia. |
| Post-conditions | N/A |
| Author | N/A |
| Assumptions | N/A |

7. encuesta de satisfacción

ID: UC07

Cuando la incidencia esté resuelta de manera positiva o negativa, los usuarios podrán elegir si puntuar la satisfacción con respecto al trabajo realizado y, si lo desean, también podrán dejar una nota referente a ello.


| | |
|------------------------------|---|
| Primary Actors |  Cliente |
| Level | N/A |
| Complexity | N/A |
| Use Case Status | N/A |
| Implementation Status | N/A |
| Preconditions | Haber sido atendido. |
| Post-conditions | N/A |
| Author | N/A |
| Assumptions | N/A |

Diagrama de clases.

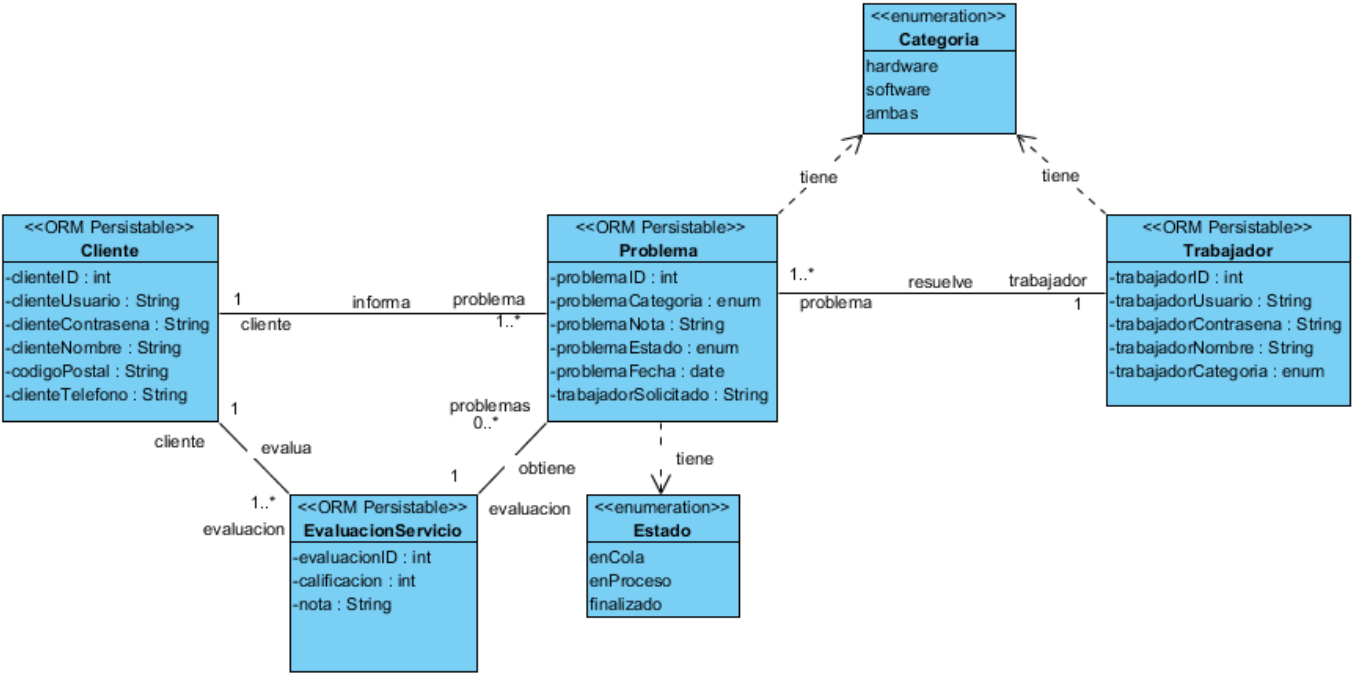
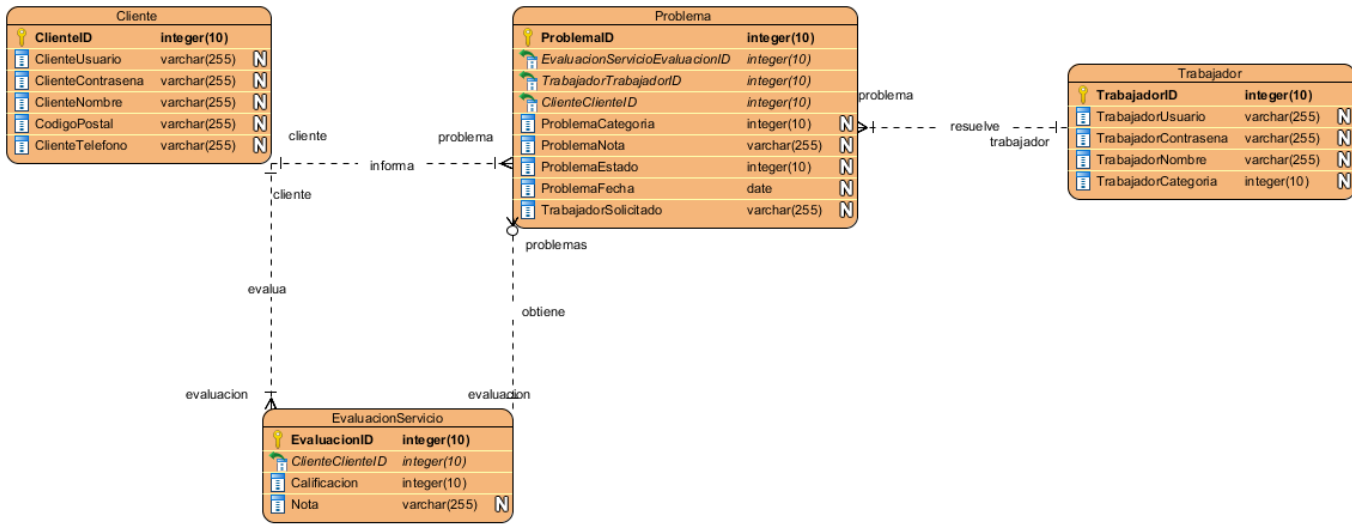
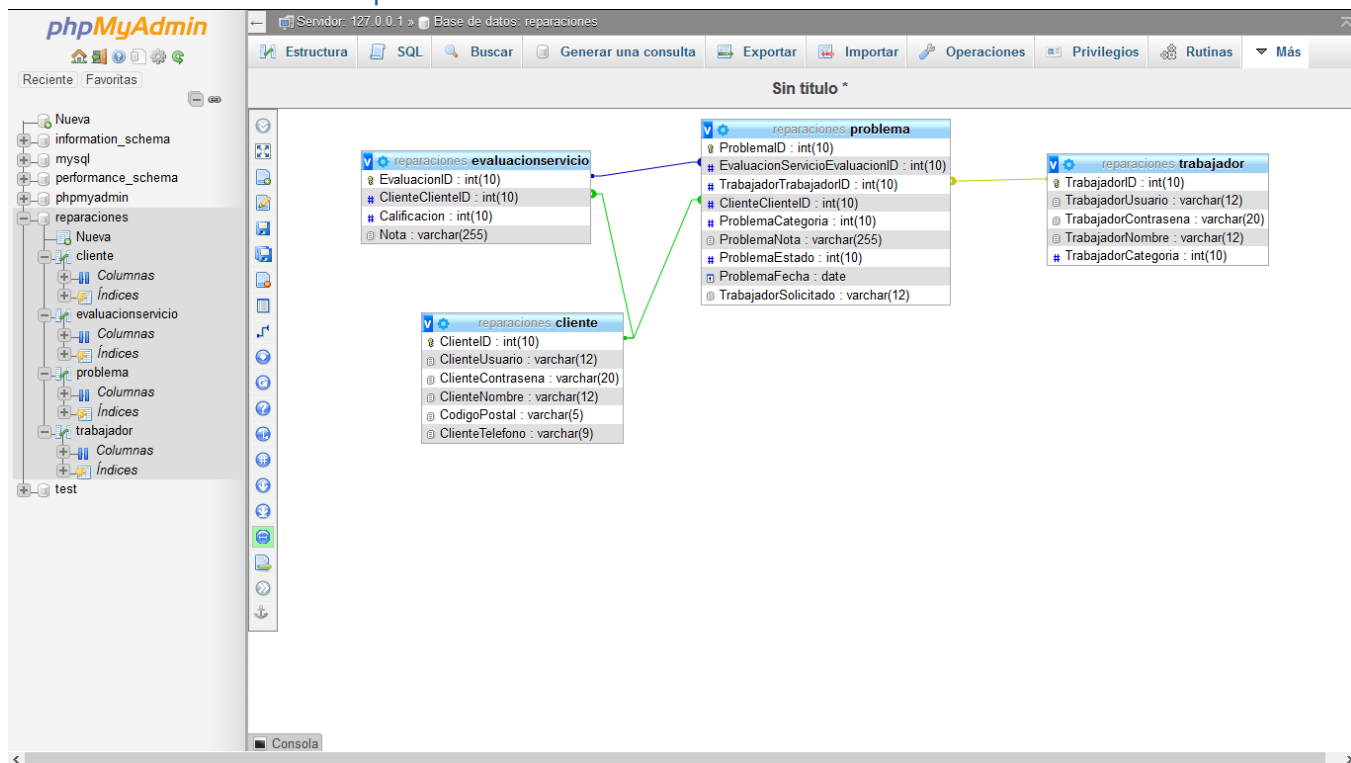


Diagrama entidad relación.



Relación de tablas implementadas



Descripción de los campos.

Cliente: Esta tabla se relacionará con evaluación de servicio para otorgarle su id al servicio y a problemas para efectuar lo mismo.

ClienteID: Un int de hasta 10 cifras que nos proporcionará la id en todo momento de los nuevos clientes.

ClienteUsuario: Varchar de hasta 12 caracteres que el usuario usará para hacer log-in en la aplicación.

ClienteContraseña: Varchar de hasta 20 caracteres para que ponga una clave de seguridad en su cuenta.

ClienteNombre: Varchar de 12 caracteres para que el usuario ingrese su nombre.

CódigoPostal: Varchar de 5 caracteres para que se ingrese el código postal del usuario.

ClienteTeléfono: Varchar de 9 caracteres para que se ingrese el número de teléfono del usuario.

Trabajador: Esta tabla se relacionará únicamente con problema para otorgar su id al problema al cual el trabajador este atendiendo.

TrabajadorID: Un int de hasta 10 cifras que nos proporcionará la id en todo momento de los nuevos

TrabajadorUsuario: Varchar de hasta 12 caracteres que el usuario usará para hacer log-in en la aplicación.

TrabajadorContraseña: Varchar de hasta 20 caracteres para que ponga una clave de seguridad en su cuenta.

TrabajadorNombre: Varchar de 12 caracteres para que el usuario ingrese su nombre.

TrabajadorCategoría: int enumerado que permite especificar la categoría del trabajador a hardware, software o ambas.

Problema: Esta tabla se relaciona con todas las demás para recibir las claves primarias de todas las tablas y así tener en cuenta al cliente atendido, al trabajador que lo atendió y a la respuesta del cliente al trabajo realizado.

ProblemaID: Un int de hasta 10 cifras que nos proporcionará la id en todo momento de los nuevos

ProblemaCategoria: int enumerado que permite al usuario especificar si el problema se trata de hardware o software.

ProblemaNota: Varchar de 255 caracteres que permite al usuario explicar el problema.

ProblemaEstado: int enumerado que permite asignar el estado de la incidencia en enCola, enProceso o finalizado.

ProblemaFecha: date que dejará el día en el cual el usuario subió el problema a la plataforma para que fuera atendido.

TrabajadorSolicitado: varchar de 12 caracteres que da la oportunidad al usuario de poder pedir un trabajador exacto para que le atienda.

EvaluacionServicio: Esta tabla se conecta con cliente para obtener el cliente atendido y con problema para darle al problema la evaluación que ha recibido el mismo.

EvaluaciónID: Un int de hasta 10 cifras que nos proporcionará la id en todo momento de las nuevas evaluaciones de los trabajos.

Calificacion: int que permite al usuario ponerle una nota a la reparación.

Nota: Varchar de 255 caracteres que permitirá al usuario dejar su opinión escrita de cómo fue la reparación.

Implementación del código en la API Rest

| | |
|-----------------------------|---|
| Título | Muestra todos los clientes |
| URL | /Reparaciones-1.0-SNAPSHOT/webresources/org.miproyecto.reparaciones.cliente |
| Método | GET |
| Parámetros de la URL | No tiene |
| Parámetros de datos | No tiene |
| Respuesta con éxito | Código: 200 keep alive Contenido de ejemplo: [{"clienteID": 1, "clienteUsuario": "FranCoruña", "clienteContrasena": "contrasena", "clienteNombre": "Francisco", "codigoPostal": "35010", "clienteTelefono": "662949985"}, {"clienteID": 2, "clienteUsuario": "Cliente2", "clienteContrasena": "cliente2", "clienteNombre": "Alfredo", "codigoPostal": "36012", "clienteTelefono": "778447512"}] |

| | |
|-----------------------------|--|
| Título | Añadir Cliente |
| URL | /Reparaciones-1.0-SNAPSHOT/webresources/org.miproyecto.reparaciones.cliente |
| Método | POST |
| Parámetros de la URL | No tiene |
| Parámetros de datos | {"clienteUsuario":"Post","clienteContraseña":"post","clienteNombre":"ElPost","codigoPostal":"14510","clienteTelefono":"123654123"} |
| Respuesta con éxito | Código: 204 { "id":19 } |

| | |
|-----------------------------|---|
| Título | Modificar Cliente |
| URL | /Reparaciones-1.0-SNAPSHOT/webresources/org.miproyecto.reparaciones.cliente/:id |
| Método | PUT |
| Parámetros de la URL | id=19 |
| Parámetros de datos | {"clienteUsuario":"PostPut","clienteContraseña":"post","clienteNombre":"ElPost","codigoPostal":"14510","clienteTelefono":"123654123"} |
| Respuesta con éxito | Código: 204 { "id":20 } |

| | |
|-----------------------------|---|
| Título | Borrar Cliente |
| URL | /Reparaciones-1.0-SNAPSHOT/webresources/org.miproyecto.reparaciones.cliente/:id |
| Método | DELETE |
| Parámetros de la URL | id=20 |
| Parámetros de datos | No tiene |
| Respuesta con éxito | Código: 204 { "id":20 } |

Tecnología utilizada.

- a) Breve descripción de las razones para la elección de la tecnología para implementar el FrontEnd.

La tecnología utilizada en el front-end fue en un inicio react, ya que vi que tenía una curva de dificultad que se elevaba gradualmente, pero luego vi que tenía ciertas incompatibilidades con hibernate (el ORM escogido), por lo que estuve buscando alternativas hasta llegar a ionic, ¿por qué

ionic? Pues porque tiene muchísima más documentación y en comparación a react es muy fácil de aprender a llevarlo.

b) Breve descripción de las razones para la elección de la tecnología para implementar el BackEnd.

Como Back-end escogí hibernate, ya que a diferencia de por ejemplo Node.js me pareció muy sencillo de implementar, sobretodo gracias a la ayuda posterior de la implementación que otorga el visual paradigm para el proyecto con hibernate.

c) Tabla comparativa con las razones para usar una u otra tecnología en este Proyecto.

| Ionic | React |
|---|--|
| Fácil de aprender con muchos componentes prediseñados. | Curva de dificultad bien trabajada con algunos componentes nativos ya creados. |
| Gran documentación y muy detallada. | En comparación a ionic tiene una documentación muy básica lo que te deja vendido a la hora de ciertas inquietudes con el código. |
| El código puede ser usado para los, Web, Windows Phone, Escritorio y Android. | El código puede ser usado (la mayoría del tiempo no en su totalidad) para los, Web, Windows Phone, Escritorio y Android. |
| Rendimiento medio. | Gran rendimiento. |
| Se usa Apache Cordoba para acceder a las funciones de hardware del móvil. | React compila en código nativo y puede acceder a las funciones nativas del dispositivo. |
| Se puede testear el desarrollo muy rápido en el navegador, sin necesidad de emuladores pesados. | Puede ser probado en emuladores o en móviles reales. |
| Ionic tiene muchas preguntas y respuestas en StackOverflow con una gran comunidad. | React tiene más comunidad en github y es muy fuerte. |
| Se basa en Angular 2 respaldado por Google. | Se basa en React respaldado por Facebook. |

Tabla con los tipos de navegación utilizados, las razones para haberlos usado, y en qué componente/clase/método se ha hecho uso de ellos.

| Tipo de navegación | Lugar dónde se produce Formulario/ Componente/Clase/Método | Razones para usar este tipo de navegación (especialmente de usabilidad) |
|-------------------------|---|---|
| Push/pop(NavController) | métodos | Preferí usar la navegación de push/pop para no sobrecargar la aplicación de pestañas como con el método de navegación tab |
| Push/pop(NavController) | anadir-cliente ver-trabajadores ver-problemas ver-evaluaciones | |

Tabla con los tipos de Autenticación implementados, y el componente/clase/método en los que se implementa.

| Tipo de autenticación | Lugar dónde se produce Formulario/ Componente/Clase/Método | Observaciones |
|-----------------------|--|---|
| [(ngModel)] | Formulario | Si no se llena el formulario no deja seguir con el envío del mismo. |