UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Seguimiento de espacios colectivos - parte 2

Alumno: Eynar Arnez Andrade
Director: Xavier Burgués Illa
Especialidad: Ingeniería del Software
Fecha: 23 de Octubre 2020

Índice

Resumen

Actualmente en la ciudad de Maputo, los responsables y colaboradores que se encuentran allí, consideran que existe la necesidad de disponer un sistema que sea capaz de gestionar las deficiencias en el servicio público (contenedores llenos de basura, alambrados sin luz, papeleras incendiadas) que van surgiendo en la ciudad. Por lo que se pensó en crear un sistema, el cual permita gestionarlas.

Se desarrolló una aplicación web multiplataforma y una interfaz para móviles antiguos, con el objetivo de que la comunidad de Maputo la pueda utilizar, ya sea la persona la que notifique la incidencia, como el personal que solvente el desperfecto.

Resum

Actualment a la ciutat de Maputo, els responsables i col·laboradors que es troben allí, consideren que existeix la necessitat de disposar un sistema que sigui capaç de gestionar les deficiències en el servei públic (contenidors plens d'escombraries, filats sense llum, papereres incendiades) que van sorgint a la ciutat. Pel que es va pensar a crear un sistema, el qual permeti gestionar-les.

Es va desenvolupar una aplicació web multiplataforma i una interfície per a mòbils antics, a l'objectiu que la comunitat de Maputo la pugui utilitzar, ja sigui la persona la que notifiqui la incidència, com el personal que solvent el desperfecte.

Summary

Currently in the city of Maputo, those responsible and collaborators who are there, consider that there is a need to have a system that is able to manage the deficiencies in the public service (bins full of garbage, wiring without light, burning bins) that are emerging in the city. So it was thought to create a system, which allows to manage them.

A cross-platform web application and an interface for older mobile phones were developed, with the aim of enabling the Maputo community to use it, whether the person reporting the incident or the staff that is responsible for the damage.A cross-platform web application and an interface for older mobile phones were developed, with the aim of enabling the Maputo community to use it, whether the person reporting the incident or the staff that is responsible for the damage.

1. Contexto

Este proyecto es un Trabajo Final de Grado(TFG) de la titulación Grado en Ingeniería Informática (GEI) de la especialidad Ingeniería del Software de la Facultad de Informática de Barcelona (FIB). Este trabajo que se está llevando a cabo, es un prototipo de sistema que se quiere instaurar en los barrios de George Dimitrov MAPUTO[1].

1.1 Formulación del problema

El objetivo del proyecto es crear un sistema que permita a los ciudadanos informar, a través de un teléfono móvil, de las situaciones problemáticas en el espacio público (fugas de agua, alumbrado estropeado, etc) y que las autoridades lo puedan gestionar y solucionar.

Como en el país hay gente que todavía utilizan móviles antiguos, que pueden o no tener acceso a internet, el sistema también tendrá conexión vía códigos USSD[2].

1.2 Agentes implicados

En este proyecto habrá agentes implicados que formarán parte del proyecto y agentes que interactúan con el sistema. Se han detectado los siguientes:

- **Director**: Es el encargado de la supervisión de la correcta realización del proyecto y se encarga de la guía y supervisión del trabajo realizado. El director es Xavier Burgués Illa.

- **Desarrolladores:** Los programadores son las personas encargadas de la planificación, desarrollo. Tendrán la función de documentar el sistema para que los ciudadanos de Maputo lo puedan utilizar. Los programadores son Eynar Arnez y Oriol Fort

- **Ayuntamiento MAPUTO**: Encargados de dar soporte al Director, facilitar los requisitos necesarios para realizar el sistema y finalmente se encargará de llevar el sistema.

- **Ciudadanos de Maputo:** Agentes que utilizarán el sistema.

2. Justificación

La tecnología móvil está llegando a todos los sectores, eso permite a un ciudadano contribuir con su ciudad para construir un smart city.

Como se puede ver en la web ‘Las 10 mejores ‘apps’ que impulsan la participación ciudadana y la transparencia’.
https://www.compromisoempresarial.com/transparencia/2018/02/las-10-mejores-apps-que-impulsan-la-participacion-ciudadana-y-la-transparencia/[3]

“La sociedad actual se encuentra en la “Cuarta Revolución Industrial””

Listado de algunas aplicaciones o webApps.

- *Cityzn: “Esta app permite que los habitantes de las ciudades aporten ideas para fomentar el bienestar común y facilitar el consenso entre vecinos y comunidades”*
- *YoVeoVeo (Ecuador): “Con el objetivo de reducir el exceso de burocracia e incentivar al ciudadano a denunciar incidencias a la administración nació esta app en Ecuador, que sirve de canal directo de comunicación entre el gobierno y los ciudadanos comprometidos con la gestión pública.”*
- *Arrels Localizador (España): “Desde la Fundación Arrels han desarrollado esta aplicación cuyo objetivo es facilitar el trabajo a los equipos de asistencia a personas sin hogar, gracias a la participación ciudadana. Desde la app, cualquier usuario puede informar de la localización de personas que pasan la noche a la intemperie para que los equipos de la Fundación puedan asistirlas.”*

Como se puede observar, la tendencia que se está siguiendo es que el ciudadano tenga participación en su ciudad.

Ahora mismo para el proyecto, existe una competencia directa, la aplicación Mopa. Como se puede observar en la imagen (sacada de la página web ’MOPA’), esta aplicación está en uso. Y de mucha utilidad para la ciudad de Maputo.

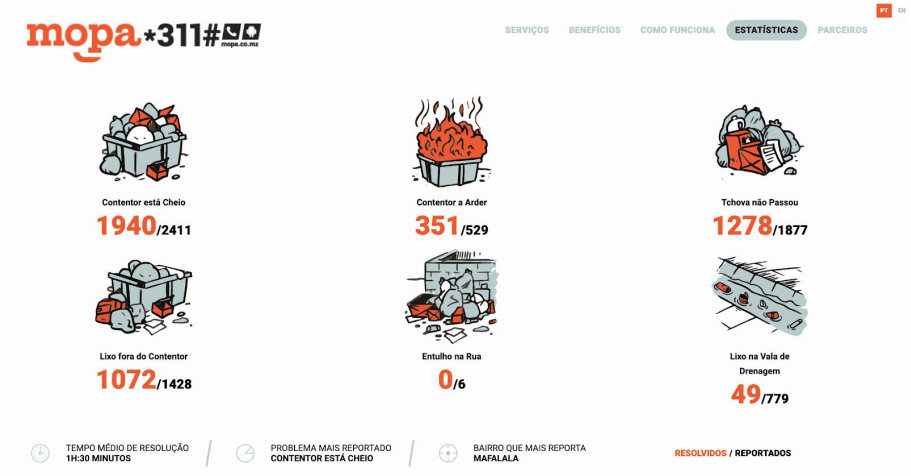


Imagen 1.- Aplicativo Mopa

El problema que tiene esta aplicación es que se encuentra paralizada debido a problemas legales y de financiación con el banco internacional. Por lo cual se ha optado por realizar un nuevo proyecto, independiente. Este nuevo proyecto, como requisito indispensable, es cumplir con todas las funcionalidades de Mopa.

2.1 Punto de partida

Si bien este es un proyecto estaba pensado para que lo realizara un alumno de la UPC, al final lo realizarán dos, por lo que el sistema tendrá más alcance.

Teniendo en cuenta la información proporcionada por parte del Director del proyecto, este trabajo será la primera fase del proyecto que se está llevando a cabo en la ciudad de Maputo.

3. Alcance del proyecto

El objetivo principal del proyecto es tener un sistema que sea capaz de reportar incidencias. Esto se llevará a cabo por dos medios, una webApp o código USSD. Además, un backend. Con fecha de inicio en febrero y fecha fin en junio del 2020

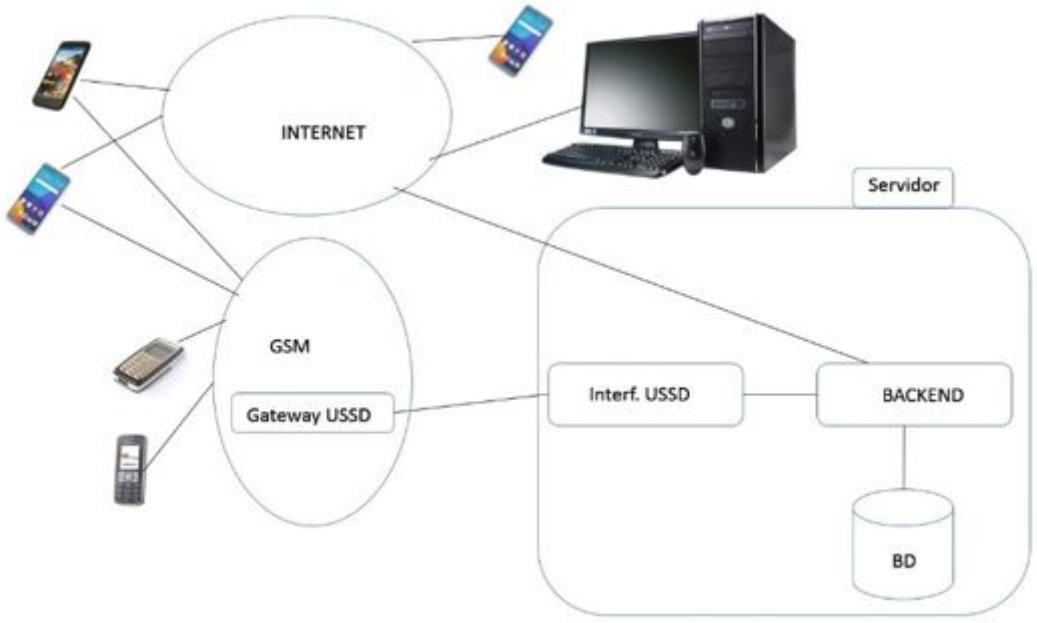


Imagen 2.- Prototipo del sistema

El proyecto finalizará a finales de junio, con la lectura de la memoria. Al tratarse de un prototipo, se espera que el siguiente semestre, otros alumnos den continuidad al proyecto.

3.1 Objetivos

El objetivo principal del proyecto es el desarrollo de un sistema que sea capaz de dar soporte a la ciudad de Maputo, por lo que se desarrollará una webApp y una interfaz USSD en la parte frontEnd y una Api para el backEnd.

- *El sistema debe permitir dar de alta incidencias:* esta funcionalidad es clave en el sistema, ya que a partir de esta funcionalidad, los agentes implicados (perfil ayuntamiento/institución) empiezan a interaccionar con el sistema

    - desde la web
    - desde USSD
- *El sistema debe permitir consultar incidencias:* el usuario tiene que poder realizar una búsqueda de una incidencia, con diferentes tipos de búsqueda (fechas, estados, categorias, etc)
    - *El usuario puede buscar incidencias por número de referencia*
    - *El usuario puede buscar incidencias por estado*
    - *El usuario puede buscar incidencias por categoría*
    - *El usuario puede visualizar incidencias en el mapa*
- *El usuario puede editar/eliminar una incidencia: Por realizar una modificación de una incidencia, dependerá del perfil de usuario, ya que no todos los usuarios podrán realizar esta acción*
    - *El usuario puede poner un comentario a la incidencia y una foto, opcionalmente*
- *El sistema no puede dar de alta fuera de la superficie de Maputo*

3.2 Posibles obstáculos

Los posibles obstáculos identificados en el proyecto son:

- Obtener todos los requisitos para integrarlo en el sistema, ya que al tratarse de un prototipo, no tendrá todo el alcance que se esperaría de un sistema acabado.
- Desconocimiento de la tecnología USSD: no se tiene conocimiento del desarrollo USSD, por lo cual, falta determinar el alcance que pueda llegar a tener en el desarrollo en el sistema.
- En primera instancia se tiene determinado que ellos tienen que proporcionar el servidor. Pero no hay garantía, esto implica tener un servidor en las instalaciones de la FIB.

4. Metodología

4.1 Metodología de trabajo

La metodología de trabajo que se utilizara en el proyecto será, una metodología cascada incremental. En la cual se ha decidido realizar 6 etapas. Se ha de tener en cuenta que el desarrollo del sistema lo llevarán a cabo dos alumnos, por lo cual, habrá partes comunes y partes individuales.

A continuación se describe las tareas ha realizar en cada etapa:

Gestión del proyecto

Etapa en la cual realizarán los alumnos de forma individual el curso de GEP, constan de 4 entregables

Iteración 1

Pensada para realizar el análisis del sistema, esta iteración las tareas la realizarán de forma conjunta los dos alumnos, en base a la información recogida del director del proyecto en el desplazamiento a Maputo.

Se decidirá la tecnología a utilizar para la instalación del servidor, análisis de requisitos(requisitos funcionales, requisitos no funcionales, casos de usuarios) y una primer prototipo de la API para la comunicación entre el frontEnd y el backEnd

Iteración 2-5

Se ha decidido realizar iteraciones cortas (dos semanas) para tener un mejor seguimiento del proyecto. Todas las iteración serán iguales.
Etapas en la que los alumnos realizarán el desarrollo de sus partes asignadas de forma individual.

Iteración 6

Pruebas de integración y preparación de pruebas de usabilidad del sistema
En esta etapa, los dos alumnos de forma conjunta realizarán pruebas de todo el sistema para validar el correcto funcionamiento del sistema.Y preparar pruebas de usabilidad para que lo testeen los ciudadanos de Maputo.
Está previsto que el viaje lo realice el otro alumno (Oriol Fort), el cual tiene como objetivo formar y testear. Realizar una formación del sistema a los trabajadores del ayuntamiento y hacer las pruebas de usabilidad (generadas por los dos alumnos) con personas de Maputo. Estas pruebas que se llevarán a cabo serán útiles de cara a las modificaciones que se puedan realizar en esta versión o versiones posteriores.

Memoria y presentación oral

Los alumnos realizarán de forma individual sus respectivas memorias y preparación de la defensa oral ante el tribunal.

4.2 Validación del proyecto

La validación del proyecto se realizará mediante reuniones periódicas entre el director y los dos alumnos que están desarrollando el proyecto. Las reuniones permitirán resolver las dudas y problemas que surgirán en el desarrollo.

También validar que las funcionalidades implementadas cumplen con las expectativas. Esta validación de funcionalidades primero se realizará entre el director y los alumnos y después en el viaje a Maputo, el alumno Oriol Fort realizará las pruebas con los ciudadanos y personal del ayuntamiento para validar el sistema.

4.3 Implementación

4.3.1.Implementación FrontEnd

Esta parte la tiene asignada el alumno Eynar, será el encargado de implementar la webApp en Ionic.
Siguiendo la metodología cascada incremental, en cada iteración, se asignará tareas (las prioritarias), para así, al finalizar el sprint poder valorar y analizar el ritmo de trabajo.
Ver los obstáculos que se han ido generando a lo largo de cada sprint y saber si la velocidad en la que se está desarrollando es la adecuada para cumplir los objetivos.

4.3.2. Implementación BackEnd

Paralelamente, el alumno Oriol, se encargará de implementar el backEnd.
Seguirá la misma metodología de trabajo que Eynar.

4.4 Herramientas de seguimiento

Las herramientas de control del proyecto serán:

Git

El git permitirá tener el control del trabajo que se vaya realizado a lo largo del proyecto en el apartado de implementación de código.

Se dividirá en dos repositorios: la parte frontEnd y la parte backEnd. Por cada iteración se creará una rama de desarrollo.

Skype

Se ha consensuado entre los dos alumnos y el director realizar reuniones una vez a la semana a lo largo del proyecto. Esto servirá para informar del avance del proyecto y consultar dudas.

Se utilizará la aplicación Skype para realizar reuniones telemáticas en caso de no poderlas realizar de forma presencial.

Google Drive

Para almacenar toda la documentación del proyecto, se utilizará Google Drive.
Una carpeta compartida donde se pondrá toda la información.

5. Estimación temporal

La fecha de inicio del proyecto es el 17 de febrero del 2020 y la fecha de finalización estimada es el 29 de junio, el segundo turno para realizar las presentaciones del trabajo final del grado. Por lo cual, la duración del proyecto es de tres meses.
Como el proyecto lo realizarán dos alumnos, primero se tiene que hacer una estimación de todo el proyecto, en la tabla 1 se muestra la estimación en horas del proyecto.

Etapa	Descripción	Tiempo Aprox	Alumno
Gestión del proyecto	Contexto y alcance Planificación Presupuesto y sostenibilidad Documento final	75h	Eynar
		75h	Oriol
Iteración 1	Análisis y diseño	75h	Eynar
		75h	Oriol
Iteración 2	Implementación FrontEnd Pruebas	55h	Eynar
	Implementación BackEnd Pruebas	55h	Oriol
Iteración 3	Implementación FrontEnd Pruebas	55h	Eynar
	Implementación BackEnd e implementación interfaz USSD Pruebas	55h	Oriol
Iteración 4	Implementación FrontEnd Pruebas	55h	Eynar
	Implementación BackEnd Pruebas	55h	Oriol
Iteración 5	Implementación FrontEnd Pruebas	55h	Eynar
	Implementación BackEnd Pruebas	55h	Oriol
Iteración 6	Implementación Pruebas del sistema y pruebas de usabilidad	60h	Eynar
	Implementación Pruebas del sistema y pruebas de usabilidad	60h	Oriol
Memoria y presentación oral	Redacción de la memoria Preparación de la defensa oral	50h	Eynar
	Redacción de la memoria Preparación de la defensa oral	50h	Oriol
Total		960h	

Tabla 1.- Estimación en horas de las tareas del proyecto

Es importante remarcar, que cada uno de los alumnos tendrá la misma carga de trabajo que el otro, por lo cual, cada alumno tendrá una carga de trabajo de 480h en todo el proyecto.

5.1. Diagrama de Gantt estimación inicial

Tabla 2.- Diagrama de Gantt

5.2. Gestión de riesgos

Durante el desarrollo del proyecto, surgirán problemas, los cuales pueden llegar a ser críticos para el desarrollo.
Los principales problemas que pueden surgir a lo largo del proyecto:

- El tiempo es un punto crítico, no realizar una buena planificación puede poner en peligro la finalización del proyecto. La solución es una buena planificación de las tareas, cumplir las fechas de entregas al finalizar el sprint y dejar un margen de tiempo entre tareas. **Probabilidad baja**
- Otro punto crítico, será en las iteraciones. En el apartado de implementación específicamente. Pueden surgir problemas a la hora de desarrollar los casos de uso, por ello se ha decidido realizar pruebas unitarias. Eso evitará errores en el sistema. **Probabilidad alta**
- El ayuntamiento de Maputo tiene que proporcionar el servidor, pero como no hay garantía, se está realizando documentación de como instalar y arrancar el sistema. **Probabilidad alta**

6. Presupuesto

6.1. Recursos

Los recursos que se utilizarán en este proyecto son:

- Recursos humanos:
    - Los principales responsables del proyecto serán los alumnos que realizarán el sistema, también el director del proyecto, y las personas de la ciudad de Maputo (dos roles, ciudadanos y los trabajadores del ayuntamiento e instituciones).
        - Dos alumnos tendrán las funcionalidades de analizar, desarrollar, testear, documentar y presentar el proyecto
        - Director del proyecto, encargado de supervisar en el transcurso del proyecto.
        - Ayuntamiento de Maputo, será el cliente. Dará los requisitos del sistema
- Recursos hardware
    - 2 Ordenadores
        - MacBook Pro (retina 13 pulgadas, finales 2012). Dispositivo necesario para realizar el código y la memoria.
        - HP 15s-fq1117ns, 15.6" Full-HD, Intel® Core™ i7
- Recursos software
    - ganttpro[5]: Software que se ha utilizado para realizar el diagrama de Gantt
    - Ubuntu 16.04.5 LTS: Software libre que se instalará en las oficinas de Maputo para el mantenimiento del sistema y lugar en el que se instalará el servidor
    - Server: Aws Amazon server: Necesario para la comunicación entre el Backend y FrontEnd
    - Servidor PostgreQSL:
        - psql (PostgreSQL) 10.4 (Ubuntu 10.4-2.pgdg16.04+1)
    - Eclipse java 1.8: Software libre que se utilizará para el desarrollo de la capa de dominio dentro del BackEnd
    - NodeJs 13.9.0: Entorno de desarrollo para desarrollar la webApp (Ionic 4)

6.2. Costes humanos

Este proyecto está pensado para que dos personas lo realicen. Lo que implica que estas dos personas tendrán todos los roles: Analistas, desarrolladores, responsable de pruebas y documentación. En la siguiente tabla se muestra los roles y los respectivos precios por hora.

Se ha hecho una estimación de los costes gracias a la página web:
https://www.tecnoempleo.com/informe-empleo-informatica.php[6]

En la fuente citada, indica máximo/mínimo del salario bruto anual para cada rol; se ha decidido hacer la media. Con esta cifra se ha calculado el precio de la hora. 52 semanas de 40 horas. Con estos datos se ha conseguido el precio de la hora de cada rol.

Rol	Precio por hora (€)
Jefe de proyecto	17
Analista	12
Desarrollador	12
Responsable de pruebas	11,5

Tabla 1 - Precio/hora de los roles identificados

Etapa	Tareas	Horas(horas)	Recurso	Coste (€)
Gestión del proyecto	Contexto y alcance Planificación Presupuesto y sostenibilidad Documento final	150h	Jefe de proyecto	2550
Iteración 1	Análisis y diseño	150h	Analista	1800
Iteración 2	Implementación Pruebas	110h	Desarrollador (100h) Responsable pruebas(10h)	1200 115
Iteración 3	Implementación Pruebas	110h	Desarrollador (100h) Responsable pruebas(10h)	1200 115
Iteración 4	Implementación Pruebas	110h	Desarrollador (100h) Responsable pruebas(10h)	1200 115
Iteración 5	Implementación Pruebas	110h	Desarrollador (100h) Responsable pruebas(10h)	1200 115
Iteración 6	Implementación Pruebas del sistema y pruebas de usabilidad	120h	Desarrollador (80h) Responsable pruebas (40h)	960 230
Memoria y presentación oral	Redacción de la memoria Preparación de la defensa oral	100h	Jefe de proyecto	1700
Total		960h		**12730**

Tabla 2 - Estimación de los costes de los roles del proyecto

6.3. Costes hardware

El hardware necesario para desarrollar el proyecto, es un portátil.
El costo que se calcula, es a partir de la amortización. Donde después de cuatro años el software, quedará obsoleto.

Hardware	Coste(€)	Vida útil (años)	Amortización 1 año (€)	Amortización vida del trabajo (4 meses)
MacBook Pro	1500	4	375	125
Total				**125**

Tabla 3 - Estimación costes hardware

6.4. Coste Software

Los recursos de software en este proyecto tiene que ser software libre, por lo cual, el coste será cero. Se tiene que garantizar este principio en el primer prototipo del sistema.

Software	Coste (€)	Vida útil (años)	Amortización (€)
Ubuntu 16.04.5 LTS	0	-	0
Apache Tmcat/7.0.68	0	-	0
PostgreQSL	0	-	0
Eclipse	0	-	0
NodeJs	0	-	0
Total	0	-	**0**

Tabla 4 - Estimación costes software

6.5 Costes indirectos

El proyecto requiere del uso de la luz eléctrica y de internet para el desarrollo. Por tanto, se ha de tener en cuenta.

Recursos	Coste/mes	Coste total (€)
Internet	30	90
Consumo eléctrico	40	120
Total		210

Tabla 5 - Estimación costes indirectos

6.6. Imprevistos y control de gestión

6.6.1 Control de gestión

6.6.1.1 Imprevistos

Los imprevistos que pueden pasar a lo largo del proyecto pueden venir tanto de los recursos directos como indirectos, por ello se decidió a principio del proyecto tener un margen de maniobra ante posibles problemas en el desarrollo del sistema, como por ejemplo en la demora del arranque del proyecto o la curva de aprendizaje del lenguaje de programación . Por eso, para que la viabilidad del proyecto no se vea comprometido, se ha estimado un 15% respecto al presupuesto calculado.

Imprevisto	Probabilidad (%)	Precio (€)	Coste estimado (€)
Recursos directos	15	12730 + 125	1959,75
Recursos indirectos	15	210	51,5
Total			**2.011,25**

Tabla 6 - Estimación costes imprevistos

6.7 Presupuesto final

Teniendo todos los costes, es posible realizar la estimación del presupuesto del proyecto.

Recurso	Coste (€)
Recursos humanos	12.730
Recursos hardware	375
Recursos software	0
Costes indirectos	210
Costes Imprevistos	2.011,25
Total	**15.329,25**

Tabla 7 - Estimación coste del proyecto

7. Sostenibilidad

Se realizará el informe en base a la matriz de sostenibilidad:
https://www.fib.upc.edu/sites/fib/files/documents/estudis/tfg-informe-sostenibilitat-2018.pdf

7.1 Sostenibilidad ambiental

Para reducir el impacto ambiental, los recursos materiales que se han utilizado para el desarrollo del proyecto no son nuevos, se ha dado utilidad a los ordenadores de uso personal. Respecto al consumo humano siguiendo los datos del documento, una persona cotidiana consume de media 1 kWh. Eso quiere decir que el impacto del proyecto es de 960 kWh entre los dos alumnos.

El sistema está pensado para tener una mejor infraestructura en la ciudad de Maputo, lo cual mejora la huella ecológica. Ya que por ejemplo, es mejor apagar el fuego en un contenedor de basura, recoger la basura, etc.

7.2 Sostenibilidad económica

Para ver la viabilidad del proyecto, al comenzar se realizó el cálculo del presupuesto del proyecto (Apartado presupuesto) en el cual se tuvo en cuenta todos los recursos (humanos y materiales) que afectan en la realización de un proyecto. Para abaratar costes, inicialmente se decidió utilizar software libre, el cual nos permite invertir ese dinero en otros recursos.
Otra medida que se tomó para abaratar costes reduciendo el tiempo de aprendizaje de los alumnos, ya que se fueron asignando responsabilidades en las que se sentían cómodos (‘expertos en ese ámbito’). FrontEnd para Eynar y backEnd para Oriol

Aunque inicialmente se tuvo en cuenta los imprevistos y las contingencias. No se consideró un imprevisto tan grande como el covid, el cual transcoto algunos planes iniciales. Uno de los problemas que surgió a lo largo del desarrollo fue el servidor, que inicialmente se tenía que desarrollar en la FIB(sala de trabajo cedida por el CCD y servidor por tecnología para todos(TxT)), pero por problemas de desplazamiento, se tuvo que optar por un servidor en la nube (Amazon web services). Este problema afectó en costos de aprendizaje y costos monetarios siendo un servidor de pago.

Los riegos que puede tener el proyecto son, la no adaptación del sistema en la ciudad de Maputo o la activación del proyecto que ya está instaurado (Mopa)

7.3 Sostenibilidad social

La aportación que me llevo del proyecto a nivel personal será el cierre a una etapa de mi vida (la universitaria), ya que desde hace años me encuentro trabajando y viviendo de la informática. Y viendo que el proyecto puede ayudar de forma real a las personas de Maputo, es una satisfacción doble.

Teniendo en cuenta las posibilidades que tiene una ciudad como Maputo, el poder optar a tener una plataforma robusta (smart city), no entra dentro de sus prioridades. Y pueden optar a tenerlas (Mopa) pero con los riesgos comentados en el apartado 2.justificación.
La principal ventaja del proyecto, es que es código libre, y sin dependencias de Empresas, pero sí dependiendo de futuros alumnos que quieran realizar su TFG para que el sistema avance.

8. Especificación

8.1 Requisitos funcionales

Se ha decidido definir los requisitos funcionales a través de casos de uso que se definirán en el apartado más adelante (8.3 Casos de uso del sistema). Y se han categorizado para tener una mejor organización:

8.1.1 Usuario ayuntamiento/institución: Acceso a la cuenta

- Iniciar sesión/Cerrar sesión: El sistema tiene que permitir al usuario entrar(introduciendo correo y contraseña válidos en el sistema) y salir del sistema.
- Registro: El sistema tiene que permitir realizar el registro de nuevos perfiles en la aplicación
- Restablecer contraseña: El sistema tiene que permitir al usuario restablecer su contraseña, ya sea por se ha olvidado de su contraseña o por la comodidad del usuario para actualizar su contraseña.
- Editar datos perfil: El sistema tiene que permitir al usuario editar datos del perfil, siendo el correo, uno de los parámetros no modificables

8.1.2 Usuario Ciudadano/Ayuntamiento/Institución: Gestión incidencias

- Crear una incidencia: El sistema tiene que permitir dar de alta una incidencia
- Buscar una incidencia: El sistema tiene que permitir realizar la búsqueda de incidencias, ya sean las incidencias que muestra el sistema o realizando una búsqueda específica(filtro de estado, categoría, gravedad, rango de fechas o número de incidencia)

8.1.3 Usuario Ayuntamiento/Institución: Gestión incidencias

- Editar una incidencia: El sistema tiene que permitir a determinados usuarios(perfil ayuntamiento/institución) realizar cambios en una incidencia
- Notificación de una incidencia: El sistema tiene que permitir a determinados usuarios realizar notificaciones a otros usuarios para informar de las modificaciones producidas en una incidencia

8.1.4 Usuario Ayuntamiento: Gestiones super administrador

- Crear una institución: El sistema tiene que permitir al usuario dar de alta nuevas servicios externos, los cuales se se encargaran de resolver incidencias
- Editar/Eliminar institución: El sistema tiene que permitir al usuario gestionar una institución, actualizar datos de perfil (nuevas competencias, nueva dirección de correo) o eliminar institución
- Alta usuario: El sistema tiene que permitir al usuario dar de alta usuarios perfil Ayuntamiento o institución.

Diagrama de casos de uso

Usuarios del sistema

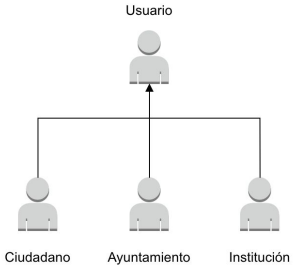


Imagen 4 - Usuarios del sistema

Usuario Ayuntamiento/Institución: Gestión de la cuenta

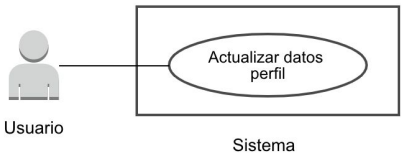


Imagen 6 - Diagrama gestión cuenta

Usuario Ayuntamiento: Casos de uso

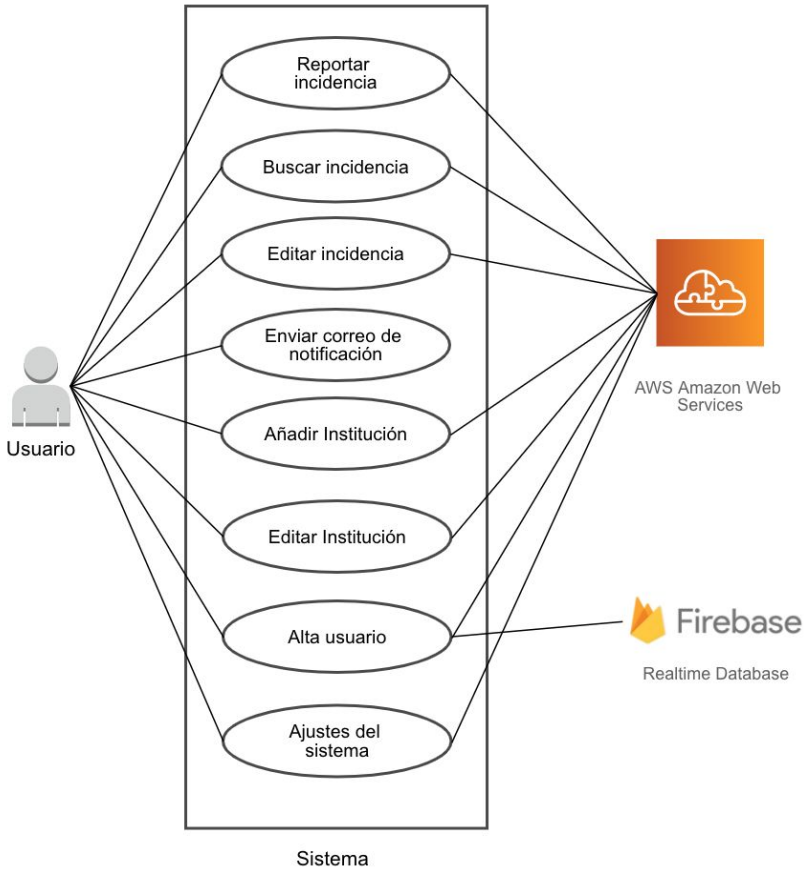


Imagen 7- Diagrama casos de uso usuario ayuntamiento

Usuario Ayuntamiento: Ajustes del sistema

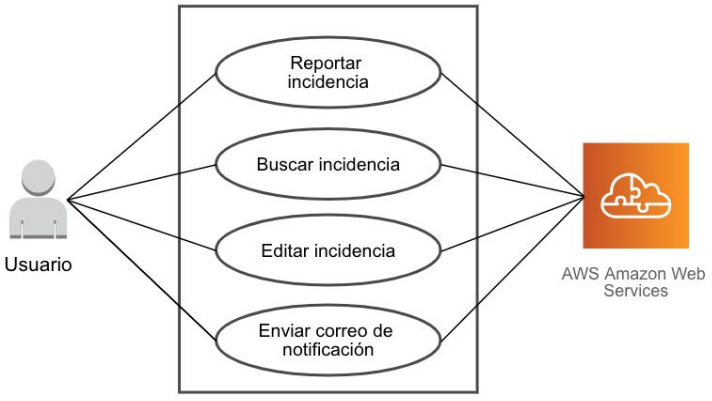


Imagen 8 - Diagrama ajustes sistema

Usuario Ciudadano: Casos de uso

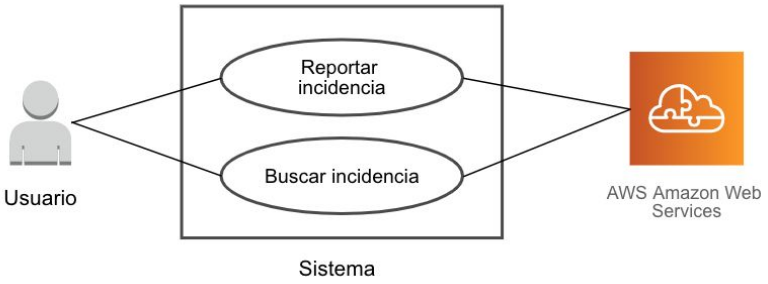


Imagen 9 - Diagrama casos de uso ciudadano

Usuario Institución: Casos de uso

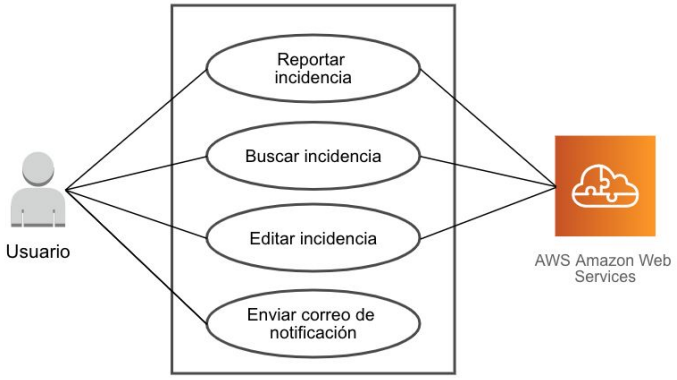


Imagen 10 - Diagrama casos de uso Institución

8.2. Requisitos no funcionales

Tipo de requisito	Personalización e internalización
Número de requisito	1
Descripción	El sistema tiene que estar disponible en español
Justificación de requisito	Al ser un prototipo, se comenzará realizando en la lengua española
Criterio de satisfacción	La webApp estará en castellano

Tipo de requisito	Aprendizaje
Número de requisito	2
Descripción	El sistema no tiene que requerir de ninguna formación previa.
Justificación de requisito	Los usuarios no tienen que conocer todo el sistema para poder utilizarlo
Criterio de satisfacción	Se realizarán pruebas en Maputo del sistema, el 85% de los que realicen la prueba tienen que validar que no se necesita ninguna formación

Tipo de requisito	Fiabilidad y disponibilidad
Número de requisito	3
Descripción	La aplicación tiene que estar siempre operativa
Justificación de requisito	El sistema tiene que estar a disposición de recibir una incidencia
Criterio de satisfacción	El hosting estará en un servidor que se instalará en Maputo.

Tipo de requisito	Seguridad
Número de requisito	4
Descripción	El usuario administrador necesitará un usuario y contraseña
Justificación de requisito	Para poder gestionar las incidencias, se necesitará tener perfil administrador
Criterio de satisfacción	Solo los perfiles administradores podrán gestionar las incidencias

Tipo de requisito	Seguridad
Número de requisito	5
Descripción	Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.
Justificación de requisito	Solo los administradores podrán gestionar las incidencias
Criterio de satisfacción	Perfil administrador podrá modificar los estados de las incidencias

Tipo de requisito	Look and feel
Número de requisito	6
Descripción	El sistema tiene que tener la misma base que la aplicación Mopa
Justificación de requisito	Los usuarios de Maputo ya conocimiento de esta aplicación
Criterio de satisfacción	Se realizarán pruebas con usuarios de Maputo, el 90% de los usuarios tienen que realizar las mismas funcionalidades que la aplicación sin cometer error

Tipo de requisito	Interfaz
Número de requisito	7
Descripción	El sistema tiene que ser multiplataforma
Justificación de requisito	Se tiene que poder acceder al sistema desde cualquier plataforma
Criterio de satisfacción	El sistema tiene que garantizar ser responsive dependiendo de la dimensión del dispositivo des del cual se accede

Tipo de requisito	Implantación
Número de requisito	8
Descripción	El sistema no se tiene que instalar
Justificación de requisito	Al tratarse de una webApp, no requiere descargar (Play Store, IOS) e instalar
Criterio de satisfacción	El sistema tiene que ser accesible desde cualquier navegador (últimas versiones y estables). No requiere ninguna instalación previa

Tipo de requisito	Implantación
Número de requisito	9
Descripción	El servidor y componentes deben ser portables en plataformas GNU/Linux y Windows, con máquinas que presentan arquitecturas de 64 bits
Justificación de requisito	El sistema que se está desarrollando tiene que ser implantado y tener un mantenimiento por personas en Maputo
Criterio de satisfacción	El sistema que se desarrollará tendrá que ser instalada en las oficinas de Maputo

Tipo de requisito	Implantación
Número de requisito	10
Descripción	Las interfaces de comunicación deben contener los estándares Web y fundamentalmente se deben basar en protocolos HTTP
Justificación de requisito	Se tiene que definir un estándar para la comunicación entre la capa presentación y dominio
Criterio de satisfacción	Las peticiones seguirán los protocolos HTTP, las respuestas tienen que ser JSON

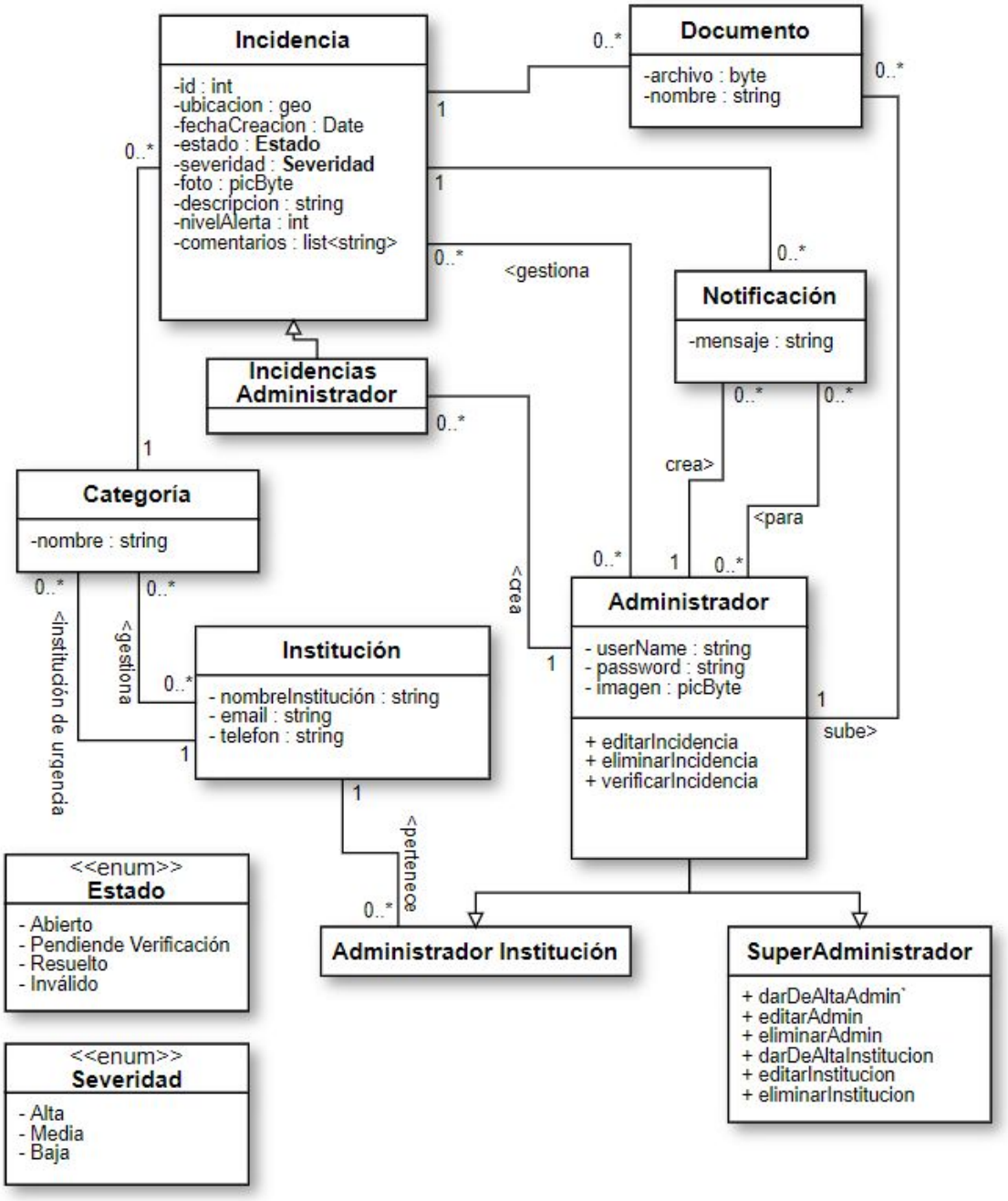
Tipo de requisito	Usabilidad
Número de requisito	11
Descripción	El sistema debe proporcionar mensajes de error que sean informativos y orientados al usuario final.
Justificación de requisito	El sistema tiene que informar de los errores que va cometiendo el usuario
Criterio de satisfacción	El sistema validará los inputs que va introduciendo el usuario, en caso de error, mostrar mensaje de error

Tipo de requisito	Usabilidad
Número de requisito	12
Descripción	El sistema debe contar con manuales de usuario
Justificación de requisito	
Criterio de satisfacción	El sistema tiene que tener un apartado ‘guía de usuario’

Tipo de requisito	Rendimento
Número de requisito	13
Descripción	El sistema no debe tardar más de diez segundos en mostrar los resultados de una búsqueda
Justificación de requisito	El sistema no tiene que tardar más de 10 segundos en dar respuesta al usuario
Criterio de satisfacción	Las peticiones que se realizarán al BackEnd esperarán como mucho 10 segundos

8.3 Modelo conceptual

Se ha realizado el modelo conceptual en base a los requisitos funcionales que se ha realizado en el apartado anterior. Esto servirá para tener más claro los conceptos del dominio del problema.



-Un usuario administrador de institución tiene permisos para editar incidencias según las categorías de incidencia gestionadas por su institución
-Al gestionar una incidencia (actualizar, crear, eliminar) se le pregunta al Administrador a que usuarios quiere notificar
-Cuando se crea una incidencia con nivel de alerta Alto, se notifica a la institución de urgencia asignada a la categoria de la incidencia.

8.4 Diagrama de estados

Este es el flujo que se ha definido en el sistema para una incidencia.

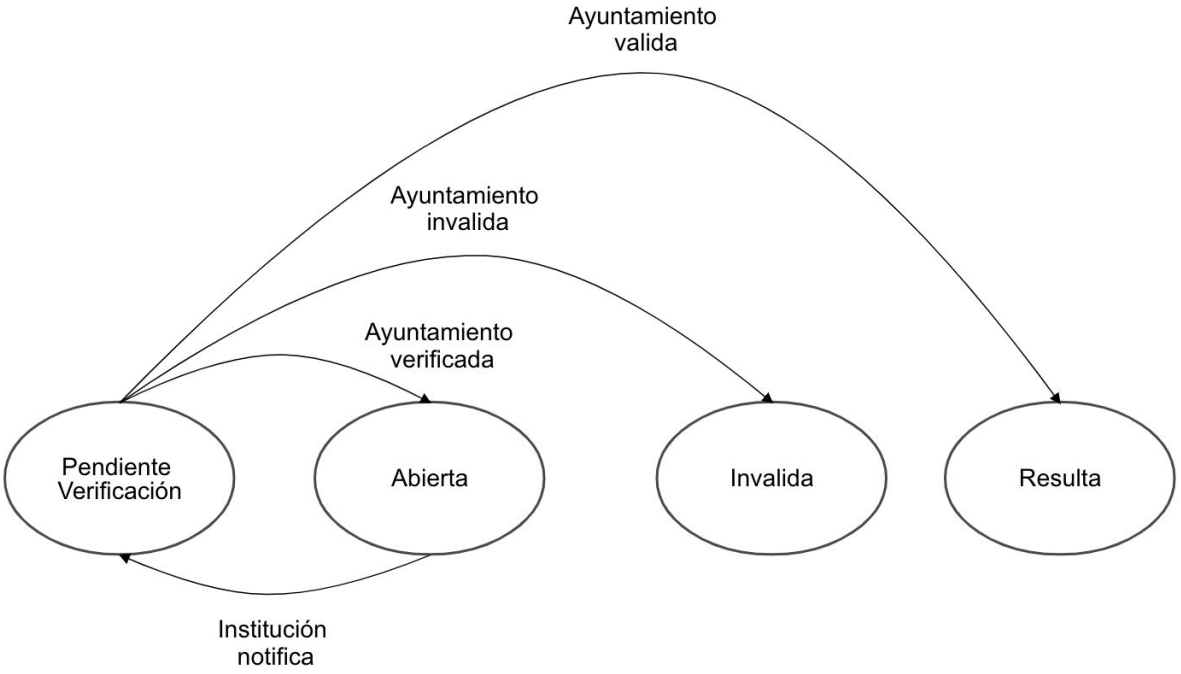


Imagen 3 - Diagrama de estados de una incidencia

- Pendiente verificación: El ayuntamiento está validando los datos, ya sea porque se ha dado de alta o se ha modificado una incidencia.
- Abierta: La incidencia está en proceso de resolución.
- Invalida: Incidencia no cumple con los requisitos mínimos establecidos por el ayuntamiento
- Resuelta: Incidencia resuelta con éxito.

8.6 Casos de usos detallado

Los casos de uso del sistema tendrán la siguiente estructura:

- Nombre del caso de uso: Titulo del caso de uso.
- Actor principal. usuario o usuarios que realizarán la funcionalidad.
- Precondiciones: el estado en el que debe estar el sistema para que se ejecute el caso de uso.
- Disparador: Acción que activa el caso de uso.
- Escenario principal: Los pasos que se producen en caso de que todo funcione de forma correcta.

- Extensiones: Los pasos alternativos que se producirían en caso de que no vaya bien.

Caso de uso	#1 Iniciar sesión perfil administrador/servicio externo
Actor principal	Perfil administrador
Precondiciones	-
Disparador	El perfil administrador quiere iniciar sesión en el sistema
Escenario principal	1. El administrador introduce correo/identificador y contraseña 2. El sistema valida las credenciales 3. El sistema redirige al usuario accede a la pantalla principal
Extensiones	1. El sistema informa los datos incorrectos 2. El sistema da alternativas para recuperar la contraseña

Caso de uso	#2 Registro al sistema perfil administrador
Actor principal	Perfil administrador
Precondiciones	-
Disparador	El usuario quiere crear una cuenta en el sistema
Escenario principal	1. El usuario rellena el formulario 2. El sistema valida los datos introducidos 3. El sistema guarda los datos introducidos 4. El sistema informa que se ha creado la cuenta 5. El sistema redirige a la página de iniciar sesión
Extensiones	1. El sistema informa de los campos introducidos son incorrectos 2. El sistema valida los datos introducidos por el usuario y el administrador principal no valida la cuenta

Caso de uso	#3 Recuperar contraseña
Actor principal	Perfil administrador
Precondiciones	-
Disparador	El perfil administrador quiere recuperar su contraseña
Escenario principal	1. El usuario quiere recuperar su contraseña 2. El sistema le informa que tiene que introducir correo asociado a la cuenta 3. El sistema valida que la cuenta exista 4. El sistema envía un correo de recuperación de contraseña al correo asociado 5. El sistema genera una url de recuperación de correo 6. El usuario clica sobre la url y accede a la página de cambiar contraseña 7. El usuario cambia contraseña 8. El sistema valida los datos introducidos 9. El sistema informa que se ha cambiado la contraseña
Extensiones	1. El sistema informa que el correo no tiene cuenta asociada 2. El usuario no introduce una contraseña válida.

Caso de uso	#4 Cerrar sesión perfil administrador
Actor principal	Perfil administrador
Precondiciones	Estar logueado en el sistema
Disparador	El perfil administrador quiere cerrar sesión en el sistema
Escenario principal	1. El usuario hace clic en cerrar sesión 2. El sistema desloguea la sesión del usuario 3. El sistema redirige a la pantalla principal del aplicativo
Extensiones	

Caso de uso	#5 Acceder al sistema en perfil usuario
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El cliente quiere acceder al aplicativo
Escenario principal	1. El cliente accede a la url de la aplicación 2. El sistema muestra el contenido de la página principal de la aplicación
Extensiones	

Incidencia

Caso de uso	#6 Dar de alta una incidencia a través de la web
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El usuario quiere reportar una incidencia desde la web
Escenario principal	1. El usuario quiere reportar una incidencia desde el navegador 2. El usuario rellena el formulario de alta de incidencia 3. El sistema valida los datos introducidos por el usuario 4. El sistema genera un nuevo número de referencia (incidencia) 5. El sistema da de alta la incidencia 6. El sistema envía un correo notificando que se ha realizado el alta de la incidencia con su correspondiente número de referencia 7. El sistema notifica a la central que se ha dado de alta una incidencia
Extensiones	3. El sistema informa que los datos introducidos por el usuario son incorrectos 4. El sistema comprueba que una incidencia del mismo tipo en la misma zona está abierta. a. Caso de uso # Incidencia repetida 6. El usuario no recibe correo de alta de referencia a. El usuario clica sobre ‘volver a enviar correo’ b. El sistema vuelve a enviar correo con el número de referencia

Caso de uso	#7 Dar de alta una incidencia a través de código USSD
Actor principal	Perfil cliente
Precondiciones	-
Disparador	El usuario quiere reportar una incidencia des del móvil,
Escenario principal	1. El usuario quiere reportar una incidencia des del móvil vía código USSD 2. El usuario envía datos vía código USSD 3. El sistema responde al usuario vía USSD 4. La interacción entre el usuario y sistema terminará hasta realizar el alta de la incidencia 5. El sistema notifica a la central que se ha dado de alta una incidencia
Extensiones	

Caso de uso	#8 Dar de alta una incidencia con prioridad
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El usuario quiere reportar una incidencia
Escenario principal	1. El usuario quiere reportar una incidencia con prioridad alta 2. El usuario introduce los datos 3. El sistema notifica a la central que se ha dado de alta la incidencia y notifica al servicio externo que se ha reportado una incidencia
Extensiones	3. La incidencia introducida es incorrecta, el administrador actualiza la incidencia a no valida y lo notifica al servicio externo

Caso de uso	#9 Categorizar incidencia
Actor principal	Perfil administrador
Precondiciones	Incidencia creada
Disparador	Al crear una incidencia, categorizar nivel de prioridad
Escenario principal	1. El sistema reporta que se ha creado una incidencia 2. El usuario corrobora que la incidencia creada tiene los parámetros estándares 3. El usuario valida la incidencia 4. El usuario categoriza el nivel de prioridad 5. El sistema estima la el tiempo en el que se podrá resolver la incidencia
Extensiones	1. El usuario no valida la incidencia creada 2. La incidencia pasa ha estado no válido 3. El sistema actualiza en base de datos el estado de la incidencia

Caso de uso	#10 Notificar incidencia a operadores externos
Actor principal	Incidencia
Precondiciones	La incidencia es validado por el perfil administrador o se actualiza incidencia
Disparador	El sistema notifica a los operadores externos para que resuelvan la incidencia
Escenario principal	1. La incidencia cambia de estado 2. El sistema genera una notificación al servidor externo apropiado
Extensiones	

Caso de uso	#11 Gestionar incidencia
Actor principal	Perfil cliente/administrador
Precondiciones	El usuario tiene que estar logueado
Disparador	El usuario quiere gestionar incidencia
Escenario principal	1. El usuario busca incidencia que quiere modificar 2. El sistema muestra todos los datos de la incidencia 3. El usuario actualiza la incidencia 4. El sistema actualiza en base de datos las modificaciones 5. El sistema notifica a servicios externos las actualizaciones producidas
Extensiones	1. El usuario hace un cambio incorrecto 2. El sistema notifica que ese cambio no se puede realizar

Caso de uso	#12 Consultar listado de incidencias
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El usuario quiere buscar el listado de incidencias
Escenario principal	1. El usuario accede a la pestaña de incidencias 2. El sistema devuelve el listado de incidencias disponibles
Extensiones	1. El sistema no devuelve el listado

Caso de uso	#13 Consultar incidencia por categoría
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El usuario quiere buscar incidencias por categoría
Escenario principal	1. El usuario accede a la pestaña de incidencias 2. El sistema filtra por categoría 3. El sistema devuelve listado de incidencias por el filtro de categoría
Extensiones	1. El sistema no devuelve el listado

Caso de uso	#14 Consultar incidencia por estado
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El usuario quiere buscar incidencias por estado
Escenario principal	1. El usuario accede a la pestaña de incidencias 2. El sistema filtra por estado 3. El sistema devuelve listado de incidencias por el filtro de estado
Extensiones	1. El sistema no devuelve el listado

Caso de uso	#15 Consultar incidencia por fecha
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El usuario quiere buscar incidencias por fecha
Escenario principal	1. El usuario accede a la pestaña de incidencias 2. El sistema filtra por fecha introducida 3. El sistema devuelve listado de incidencias por el filtro de fecha
Extensiones	1. La sistema informa que la fecha introducida es incorrecta 2. El sistema no devuelve el listado

Caso de uso	#16 Consultar incidencia por rango de fechas
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El usuario quiere buscar incidencias por rango de fecha
Escenario principal	1. El usuario accede a la pestaña de incidencias 2. El sistema filtra por rango de fechas introducida 3. El sistema devuelve listado de incidencias por el filtro de fecha
Extensiones	1. La sistema informa que el rango de fechas introducidas son incorrectas 2. El sistema no devuelve el listado

Caso de uso	#17 Consultar incidencia por referencia de incidencia
Actor principal	Perfil cliente/administrador
Precondiciones	-
Disparador	El usuario quiere buscar incidencias por referencia
Escenario principal	1. El usuario accede a la pestaña de incidencias 2. El sistema muestra información relativa a la incidencia.
Extensiones	1. El sistema informa que los datos introducidos son incorrectos

Caso de uso	#18 Detalle incidencia
Actor principal	Perfil cliente/administrador
Precondiciones	Pestaña búsqueda incidencia
Disparador	El usuario quiere visualizar el detalle de una incidencia
Escenario principal	1. El usuario selecciona la incidencia de la que quiere ver el detalle 2. El sistema busca la información del detalle 3. El sistema muestra los datos de la incidencia
Extensiones	

Caso de uso	#19 Comentario incidencia con aviso
Actor principal	Perfil cliente/administrador
Precondiciones	Pantalla detalle incidencia
Disparador	El usuario quiere comentar una incidencia
Escenario principal	1. El usuario introduce un comentario y añade un medio por el cual le llegarán notificaciones respecto a esa incidencia 2. El sistema valida los datos introducidos 3. El sistema muestra mensaje de comentario añadido correctamente 4. La incidencia notificará al usuario cada vez que cambie de estado
Extensiones	

Caso de uso	#20 Comentario incidencia sin aviso
Actor principal	Perfil cliente/administrador
Precondiciones	Pantalla detalle incidencia
Disparador	El usuario quiere comentar una incidencia
Escenario principal	5. El usuario introduce un comentario 6. El sistema valida los datos introducidos 7. El sistema muestra mensaje de comentario añadido correctamente
Extensiones	

Caso de uso	#21 Eliminar incidencia
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado
Disparador	El usuario quiere eliminar una incidencia
Escenario principal	1. El usuario accede al listado de incidencias 2. El usuario elimina una incidencia 3. La incidencia es eliminada del sistema

Caso de uso	#22 Notifica nueva incidencia
Actor principal	Sistema
Precondiciones	-
Disparador	Se crea una incidencia
Escenario principal	1. El usuario crea una incidencia 2. El sistema verifica el tipo de incidencia 3. Según el tipo de incidencia el sistema notifica a la institución correspondiente 4. La institución recibe una notificación de una nueva incidencia sobre la que debe actuar

Caso de uso	#23 Notificar incidencia cerrada/solucionada
Actor principal	Sistema
Precondiciones	El usuario introdujo su correo electrónico al abrir la incidencia
Disparador	Se cierra una incidencia
Escenario principal	1. Un administrador cierra una incidencia. 2. El sistema manda un correo al usuario que abrió la incidencia

Caso de uso	#24 Adjuntar informe
Actor principal	Perfil administrador/servicio externo
Precondiciones	Existe incidencia
Disparador	Añadir informe en un incidencia
Escenario principal	1. Acceder a una incidencia 2. Pulsar botón, Añadir informe 3. Introducir documento (formato pdf) 4. Pulsar botón, Guardar 5. El sistema añade el documento en la incidencia

Caso de uso	#25 Reportar incidencia resuelta
Actor principal	Perfil servicio externo
Precondiciones	Existe incidencia
Disparador	Incidencia solucionada
Escenario principal	1. El usuario reporta que la incidencia ha sido solucionada 2. El sistema actualiza el estado de la incidencia 3. El sistema notifica al administrador de la institución y en caso de que la incidencia tenga usuario registrado, informar al usuario

9. Diseño e implementación

9.1 Arquitectura

El sistema está dividido en dos partes. La del cliente, que dispone de una webApp y una interface USSD. Y por el otro lado, una Api en un servidor, con la funcionalidad de procesar todas los requerimientos del cliente.

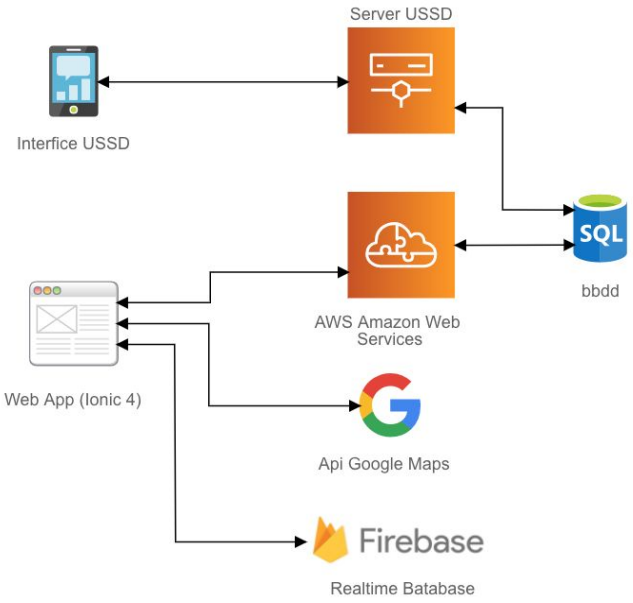


Imagen 11 - Visión general del sistema

Los elementos del sistema son:

- Interfaz USSD: Diseñado para la comunicación del sistema con dispositivos móviles antiguos
- Web App: Aplicación que permite al cliente interaccionar con el sistema
- Server Aws Amazon Web Services y servidor USSD: Servidor en la nube que se encarga de comunicar entre el frontEnd y BackEnd.
- Firebase: Plataforma que la utilizaremos para el control del usuario: Registrar/crear/modificar y restauración de la contraseña
- bbdd: Base de datos de nuestro sistema

9.2 Patrones utilizados

Al utilizar el framework Angular,se decidió utilizar el patrón MVC (modelo-vista-controlador).

Este patrón tiene la finalidad de separar los datos y la lógica de negocio de la interfaz del usuario y la interacción del sistema con el usuario.

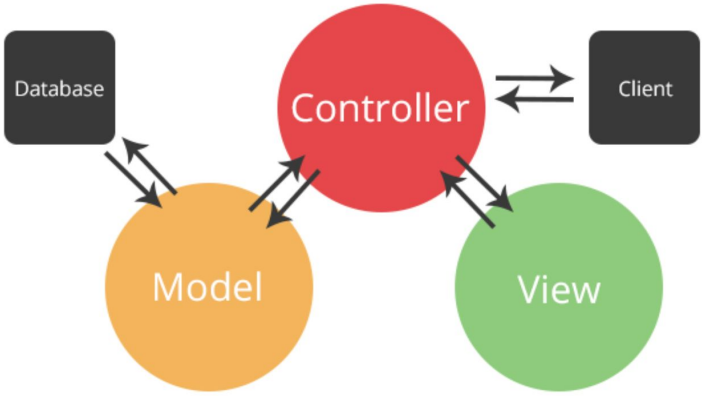


Imagen -12 Patrón MVC

- Modelo: Contiene una representación de los datos que maneja el sistema, su lógica de negocio y los mecanismos de persistencia. Los modelos del sistema se obtienen en base al diagrama de clases, ya que ahí se definen las clases con sus respectivos atributos, los cuales serán utilizados en la webApp.
- Vista: En el sistema, la interfaz de usuario, el cual muestra la información(modelo) en un formato adecuado, capaz de interaccionar con el.
- Controlador: Es el que actúa como intermediario entre el modelo y la vista, se encarga de gestionar el flujo de información entre ellas y realizar las transformaciones para adaptar los datos a las necesidades de cada uno.

9.3. Diseño de la interface

El diseño del sistema está adaptado para tener un comportamiento responsive, es decir, una correcta visualización de una misma página en distintos dispositivos (web, móvil).

9.3.1. Navegabilidad

Uno de los componentes más importantes de la webApp, es el menú de navegación, ya que es el encargado de acceder a todas las vista del sistema. En la visión web se ha optado por crear el componente desde cero (html y css básico), en cambio, en la visión móvil, la navegación tendrá un menú desplegable lateral, utilizando un componente de Ionic 4.

Tener diferentes menús de navegación dependiendo del dispositivo, es debido a la usabilidad que pueda tener el usuario. Es importante que el usuario sea capaz de tener un aprendizaje rápido de la webApp.

- web

Imagen 13 - Captura pantalla navegación web

- movíl

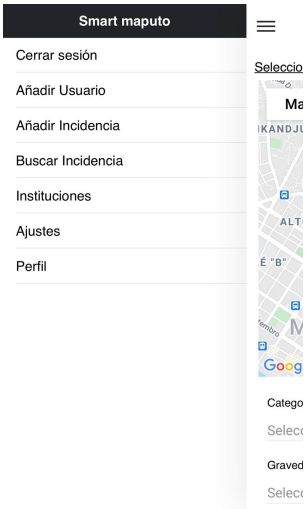


Imagen 14- Captura pantalla navegación móvil

9.3.2. Añadir incidencia

La vista añadir incidencia será la primera vista que el usuario verá, independientemente del perfil (ayuntamiento, institución, ciudadano) y la forma en la que se haya accedido (url pública, url privada) a la webApp. La única funcionalidad y fundamental de esta vista, es dar de alta una incidencia. El sistema se nutre de esta vista.
Esta vista dispone del componente de Google Maps y un formulario.

El sistema cumple la funcionalidad de no permitir dar de alta una incidencia fuera de la zona de Maputo. Este requisito se cumple gracias a un método que nos proporciona la API de google maps, el cual tranzando un polígono en la zona de Maputo, validamos si la localización seleccionada es correcta.

- web

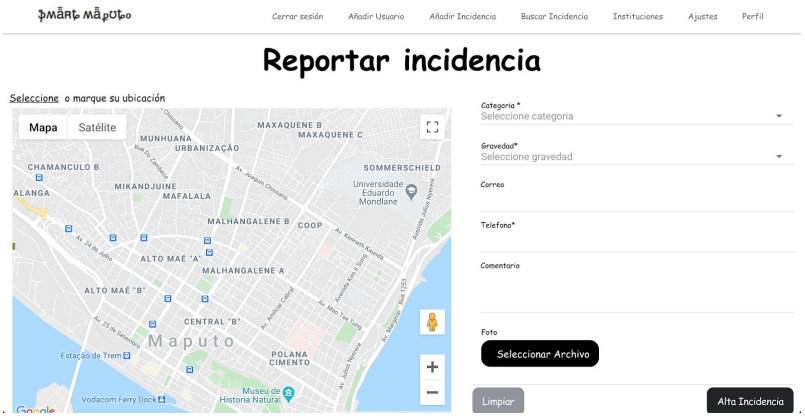


Imagen 15 - Captura pantalla Reportar incidencia - web

- movíl

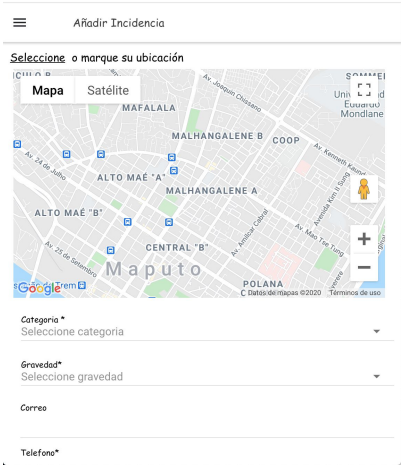


Imagen 16 - Captura pantalla Reportar incidencia - movíl

9.3.3. Vista Buscar Incidencia

La vista Busqueda incidencia, es visible para todos los usuarios, pero dependiendo del perfil dispondrá de funcionalidades visibles (editar/eliminar incidencias).

Para una mejor usabilidad, el usuario podrá ver las incidencias, de dos formas:

- ➔ Mapa: El usuario verá todas las incidencias en el mapa y el color marker dependerá del estado de la incidencia. La información de los colores, está explicada en la leyenda
- ➔ Listado: El usuario visualizará el listado de las incidencias por orden de creación

- web

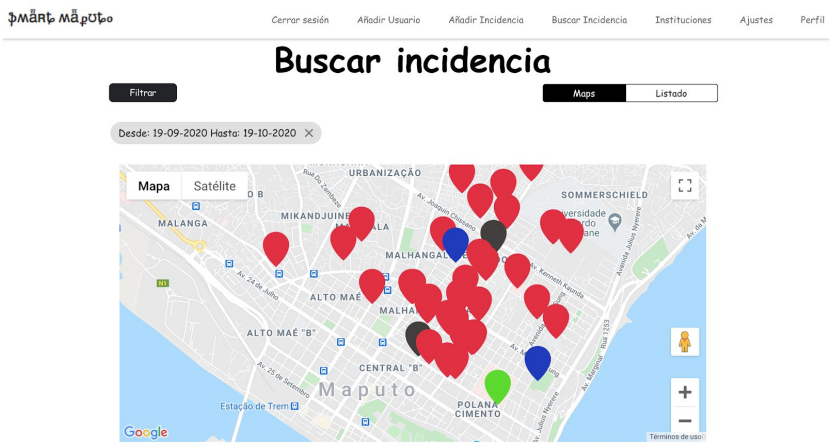


Imagen 17 - Captura pantalla búsqueda incidencia - web

- movíl

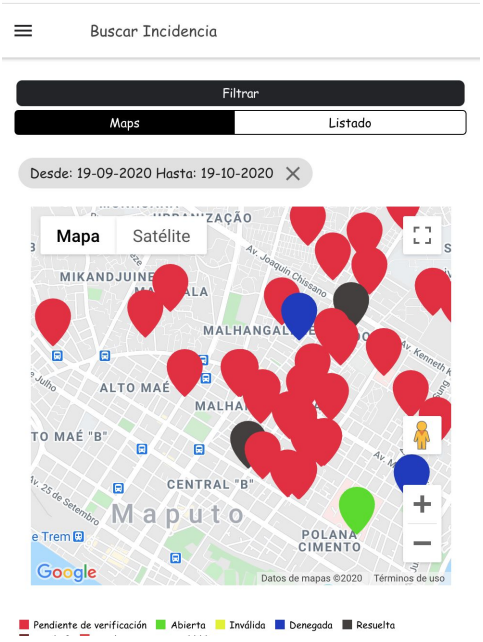


Imagen 18 - Captura pantalla búsqueda incidencia - móvil

9.3.4. Vista Instituciones

Esta vista sólo será visible para el perfil ayuntamiento. Y contiene el listado de las instituciones. Permite las funcionalidades de añadir/modificar/eliminar una institución

- web

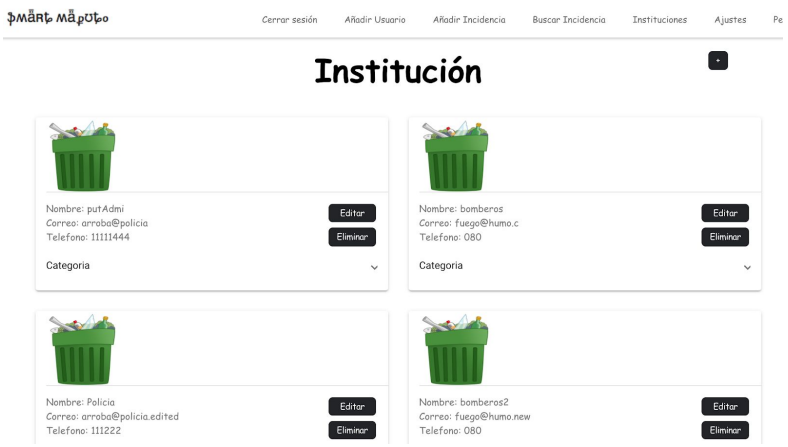


Imagen 19 - Captura pantalla instituciones - web

- movíl

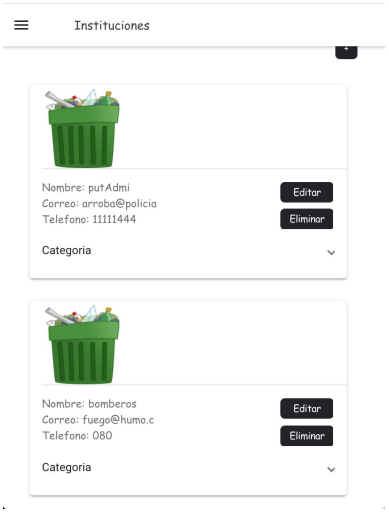


Imagen 20 - Captura pantalla instituciones - móvil

9.3.5. Iniciar Sesión

Modal de iniciar sesión que será accesible desde la web privada. Introducir usuario y contraseña para poder iniciar sesión y acceder al sistema.

- web

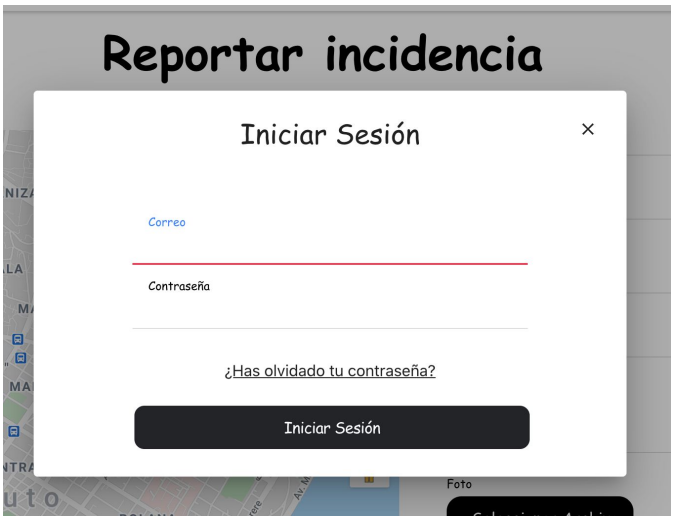


Imagen 21 - Captura pantalla modal iniciar sesión - web

- movíl

Imagen 22 - Captura pantalla modal iniciar sesión - movíl

9.3.6. Editar Incidencia

Modal editar incidencia, será visible para los perfiles ayuntamiento/institución. Muestra la información del detalle, con opción de editar los parámetros que considere el usuario. Esta vista contiene otra de las funcionalidades principales del sistema.

- web

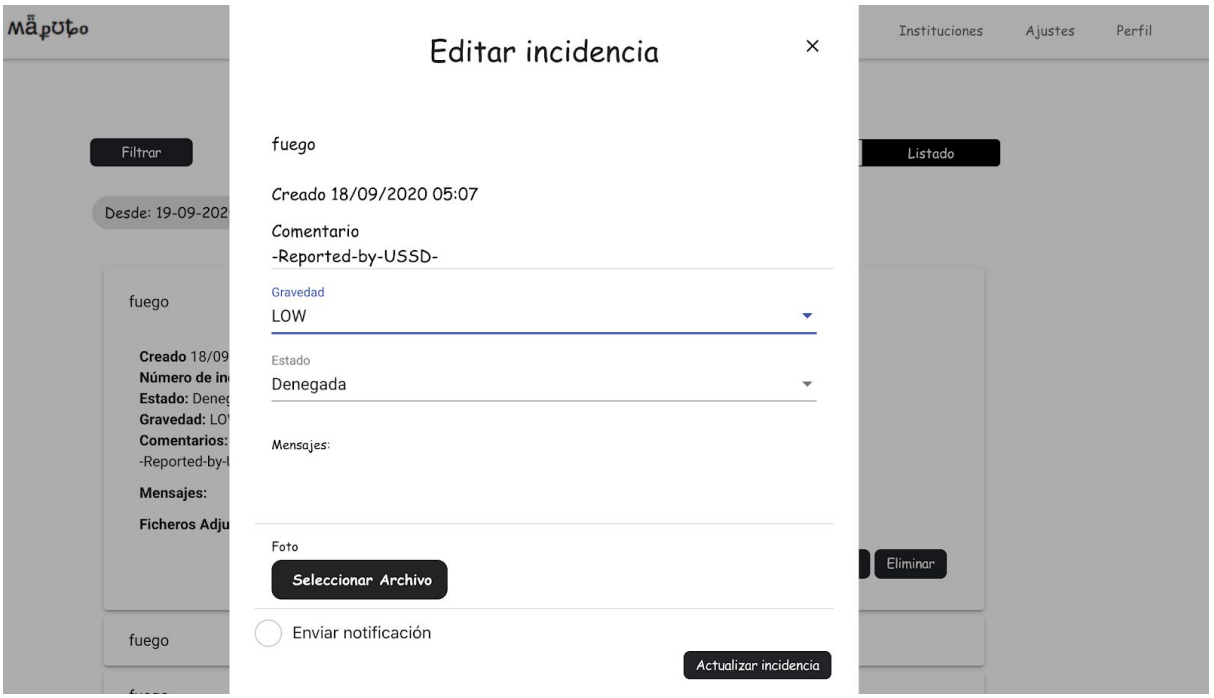


Imagen 23 - Captura pantalla modal editar incidencia - web

- movíl

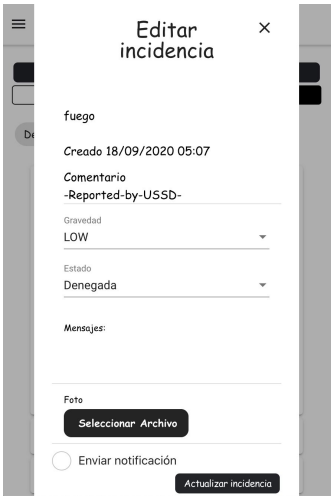


Imagen 24 - Captura pantalla modal editar incidencia - movíl

9.3.7. Ajustes

Modal ajustes, sólo es accesible con el perfil Ayuntamiento. Se creó con el propósito de que el usuario sea capaz de añadir/editar/eliminar elementos del sistema, ya sea estados, categorías o localizaciones.

- web

Imagen 24-25 - Captura pantalla modal ajustes - web

- móvil

Imagen 26 - Captura pantalla modal ajustes - movíl

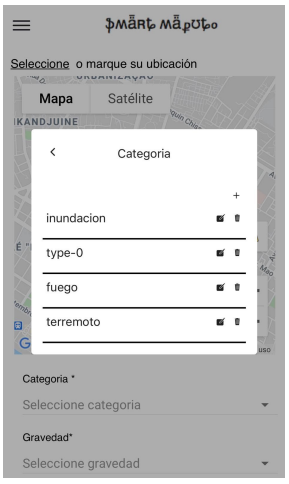


Imagen 27 - Captura pantalla modal ajustes - movíl

9.4. Implementación webApp

Repositorio Git:

- https://github.com/aaeynar/maputo

Recursos utilizados

9.4.1. Lenguaje de programación

TypeScript

Es un lenguaje de programación libre y de código abierto, siendo un superconjunto de JavaScript, el cual puede ser usado para desarrollar aplicaciones que se puedan ejecutar en el lado del cliente o del servidor (node.js) y está mantenido por Microsoft (TypeScript se incluye como lenguaje de programación de primer nivel en Microsoft Visual Studio).

Al extender de una sintaxis de JavaScript, utilizando un compilador TypeScript, se puede traducir a código JavaScript original y al revés, se puede utilizar cualquier código JavaScript debería de funcionar sin problemas.

9.4.2. Tecnologías

Angular

Angular es un framework multiplataforma de código abierto desarrollado en TypeScript y mantenido por Google. Con el objetivo de tener aplicaciones web en una sola página, Single Page Application. Utiliza el patrón MVC (Modelo-Vista-Controlador) para que sea más fácil el desarrollo y realizar pruebas.

También Angular ejecuta la primera vista de la aplicación en node.js, .NET, PHP, y otros servidores para renderizar de forma casi instantánea obteniendo solo HTML y CSS.

Angular es la evolución de AngularJS aunque incompatible con su predecesor.

Ionic 4

Ionic es un framework para construir aplicaciones móviles híbridas basada en otro Framework JavaScript, AngularJS y Apache Cordova. Para desarrollar Ionic, se utiliza HTML, CSS, TypeScript, proporcionando aplicaciones web progresivas.

Utilizando Cordova o Capacitor, se pueden crear aplicaciones nativas capaces de distribuir en sus plataformas.

Node.js

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor utilizando JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Express.js

Express.js es un framework de desarrollo de aplicaciones web minimalista y flexible para Node.js. Robusto, flexible y muy simple. Diseñada para crear aplicaciones web y API y lanzado como software gratuito y de código abierto bajo la licencia MIT

Apache cordova

Cordova es un entorno de desarrollo de aplicaciones móviles, el cual permite utilizar las tecnologías estándar web como HTML, CSS y JavaScript. Esto nos permite utilizar para el desarrollo multiplataforma.

Apache consigue utilizar APIs nativas dependiendo el entorno de ejecución del dispositivo como los sensores, datos, cámara, etc.

Heroku

Heroku es una plataforma en la nube que permite construir, entregar y supervisar aplicaciones y alojarlas en la nube, esto implica que el desarrollador no se tiene que preocupar de la infraestructura (un servidor) sino, sólo del desarrollo.

Soporta el desarrollo en una gran variedad de lenguajes de programación, Ruby, Java, PHP, NodeJS

AWS (Amazon Web Services)

Amazon Web Services, es una plataforma de servicios de nube que proporciona variedad de servicios, ofrecidas a través de internet por Amazon.com. Puede dar servicios de almacenamiento, base de datos, servicios de aplicaciones, servicios móviles, inteligencia artificial. Es considerado como un pionero en su campo.

Google Maps Platform

Google Maps Platform es una API que recupera datos de Google Maps, esta utilidad lo aprovechan los desarrolladores para añadirlas en su aplicaciones móviles o páginas web. Dependiendo de las necesidades, el desarrollador utiliza las APIs y SDK que crea conveniente

FireBase

Plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por Google. Se encuentra ubicada en la nube, integrada con Google Cloud Platform. Utiliza herramientas que hacen que se integre fácilmente en aplicaciones web o aplicaciones móviles. Sincroniza los datos de los proyecto sin tener que administrar las conexiones y no hay necesidad de utilizar un servidor, el desarrollador utiliza la SDK que van publicando para poder utilizarla.

9.4.3. Herramientas

Git

Git es un software de control de versiones, es decir, fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones si estas disponen de un gran número de archivos de código fuente

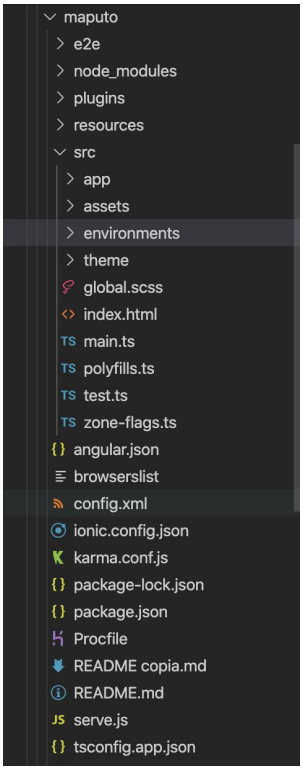
Visual Studio Code

Visual Studio Code es un editor de código fuente para que lo puedan utilizar Windows, Linux y macOS, desarrollado por Windows. El cual incluye muchas funcionalidades muy útiles para los programadores, como la depuración, control integrado de Git, finalización inteligente de código, etc.

9.4.4 Desarrollo FrontEnd

Estructura del proyecto

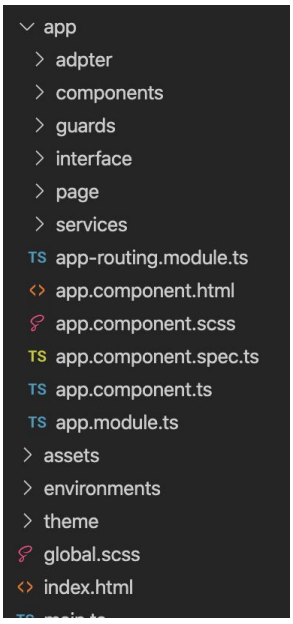
En el desarrollo del sistema, es importante saber cómo se estructura el framework que se utilizara en el proyecto, ya que esto ayudará al alumno a desarrollar y desenvolverse mejor. Por ello es bueno tener conceptos relevantes al utilizar el framework de angular, como:



- **Package.json:** Fichero que contiene metadatos sobre su aplicación o módulo, así como la lista de dependencias para instalar desde npm cuando se ejecuta npm install
- **src:** Carpeta que contiene todo el código fuente del proyecto
- **node_modules:** Directorio que se crea en la carpeta raíz del proyecto al instalar los paquetes o dependencias mediante npm

Imagen 27- Estructura proyecto Ionic 4

Y en específico, la carpeta src, en la que se como se ha mencionado antes, es la que contiene el código fuente del proyecto. Y también se ha realizado una categorización de carpetas.



Módulos: Son los contenedores para almacenar los componentes y servicios de la aplicación. A partir de un módulo raíz se enlazan otros módulos. El principal es el app.module.ts

Componentes: Son los bloques básicos de la construcción de las páginas web en Angular. Contiene una parte visual en html (Vista) y una funcional en typeScript (Controlador).

Servicios: Son los proveedores de datos, que mantiene lógica de acceso a ellos y operativa relacionada con el negocio. Los servicios serán consumidos por los componentes

Imagen 28 - Captura carpeta src proyecto Ionic 4

Enrutamiento del aplicativo

Como se había mencionado en apartados anteriores, se han creado dos accesos al aplicativo, el acceso para el usuario ciudadano y el acceso para los usuarios institución y ayuntamiento. El enrutamiento se encuentra en el fichero app-routing.module.ts

app-routing.module.ts

git > subidaMaputo > maputo > src > app > app-routing.module.ts > ...

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  { path: '', redirectTo: 'public', pathMatch: 'full' },
  { path: 'public', loadChildren: () => import('./page/home/home.module').then( m => m.HomePageModule)},
  { path: 'private/home', loadChildren: () => import('./page/private/home/home.module').then( m => m.HomePagePrivateModule)},
  { path: 'not-found', loadChildren: () => import('./page/not-found/not-found.module').then( m => m.NotFoundPageModule)},
  { path: '**', redirectTo: 'not-found',},
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules, useHash: true })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Imagen 29 - fichero app-routing-module.ts

Pages

Son los accesos a nuestras pantallas, en el proyecto se ha decidido tener la página pública y la página privada, que serán los controladores de los componentes genéricos que se han creado

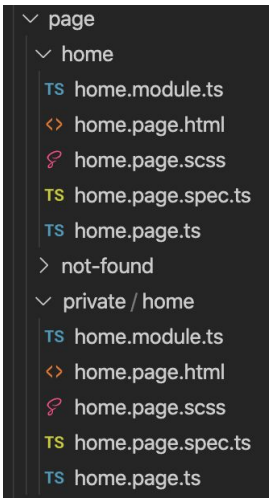


Imagen 30- Page public y page private del sistema

Componentes genéricos

Una de las buenas prácticas en Angular, es la generación de componentes genéricos para no duplicar código. Por ello, se decidió, crear un módulo compartido, del cual se podrían nutrir las dos páginas creadas (Pagina publica y privada).

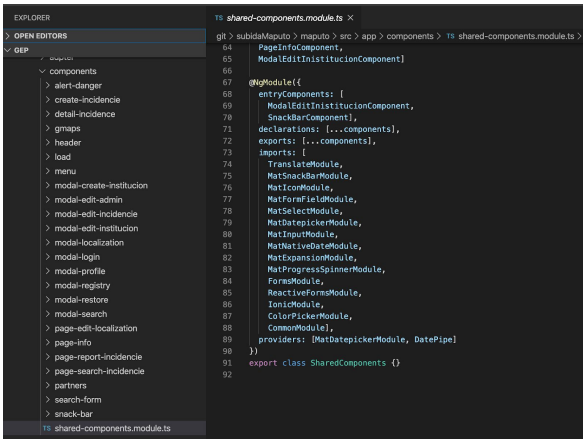


Imagen 31 - carpeta componentes

Dentro de los componentes genéricos, encontramos todas las las vistas del sistema (repostar incidencia, buscar incidencia), los modales (iniciar sesión, editar incidencia, editar institución).

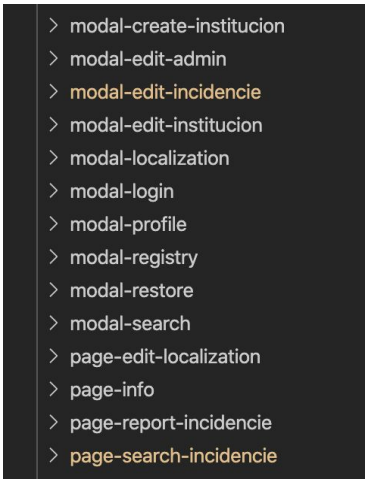


Imagen 32 - Ejemplo de componentes del aplicativo

Todos estos componentes se encuentran en el módulo *SharedComponents*. Y para poder utilizar los componentes, lo tenemos que importar en el módulo del Page.

```
@NgModule({
  entryComponents: [
    ModalLocalizationComponent,
    ModalSearchComponent
  ],
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    MatMenuModule,
    MatToolbarModule,
    SharedComponents,
    MatDialogModule,
    RouterModule.forChild([
      {
        path: '',
        component: HomePage
      }
    ])
  ],
  declarations: [HomePage]
})
export class HomePageModule {}
```

Imagen 33 - Visualización del import del módulo ShareCompontes en la página HomePage - public

Servicio

Para la manipulación de datos, se han utilizado varios servicios, pero uno de los más importantes, es el servicio para la conexión a la Api(BackEnd). Este servicio se hizo mediante peticiones http, de forma bidireccionales

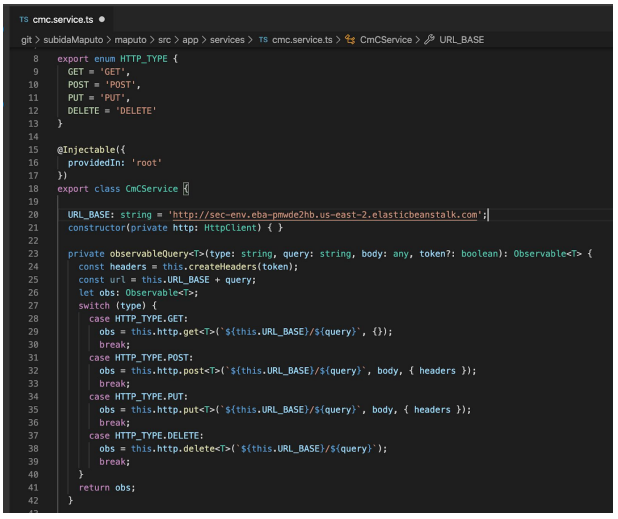


Imagen 34- Servicio llamadas backEnd

9.5.Implementación backEnd

Recursos utilizados en la implementación del backEnd

9.5.1 Lenguaje de programación

Java

Es un lenguaje de programación y una plataforma informática. Una de las características principales es que es una programación orientada a objetos.

Maven

Maven es una herramienta de software para la gestión y construcción de proyectos Java. Tiene un modelo de configuración de construcción simple, basado en un formato XML

Hibernate

Hibernate es una herramienta de mapeo objeto-relacional para Java que facilita el mapeo de atributos entre una base de datos relacional (MySQL) y el modelo de objetos de una aplicación.

MySQL

9.5.2 Herramientas

IntelliJ

IntelliJ IDE, es un entorno de desarrollo integrado para el desarrollo de programas informáticos, en esta versión se ha utilizado la versión gratuita

Postman:

Herramienta que permite crear peticiones sobre APISs, en el proyecto, para peticiones HTTP

9.5.3 Desarrollo backEnd

Api Maputo

El encargado de la implementación del backEnd fue el alumno Oriol Fort. Se encargó de crear la base de datos y generar los endpoints necesarios para el consumo desde la parte frontEnd.
Para ello se utilizó los métodos de petición HTTP

- GET: Solicita una representación de un recurso específico. El método GET sólo debe recuperar datos
- POST: Se utiliza para enviar datos a un recurso específico, pudiendo producirse un cambio. Está orientado a la creación de nuevos contenidos.
- PUT: Se utiliza para enviar datos al servidor, pudiendo producirse un cambio. Está orientado a la actualización del contenido contenido.
- DELETE: Borra el recurso específico

La url base es: http://sec-env.eba-pmwde2hb.us-east-2.elasticbeanstalk.com/
Se pueden catalogar por diferentes funcionalidades

- Usuario: Los endpoints(recursos) creados para usuario, tienen estas funcionalidades: alta/consultar/editar/borrar usuario y también autentificarse

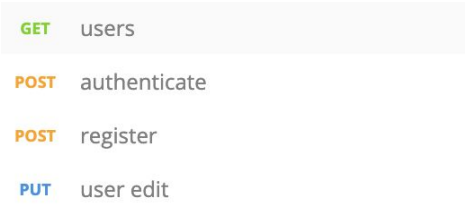


Imagen 35 - Endpoints user

- Incidencia: Los endpoints(recursos) creados para la manipulación de datos para las incidencias son: crear/buscar/editar/eliminar incidencia y un endpoint para añadir comentarios

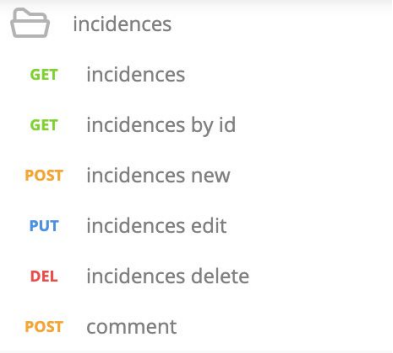


Imagen 36 - Endpoints incidencia

- Institución: Los endpoints(recursos) creados para la manipulación de datos para las instituciones son: crear/buscar/editar/eliminar institución.

Imagen 37 - Endpoints institución

- Ajustes: Los endpoints(recursos) creados para la manipulación de datos para los ajustes: crear/editar/buscar/eliminar estados, categorías, gravedad y localizaciones.

Imagen 38 - Endpoints ajustes

Api Notificaciones

Para el envío de notificaciones (correos), en un primer momento, se desarrolló un endpoint en la Api Maputo y funcionaba de forma correcta en local. Pero, se encontraron dificultades en la subida al AWS server (servidor de Amazon) y no se consiguió subirlo a producción
Por ello se decidió implementar una Api en otro servidor(Heroku) y en otro lenguaje de programación(Express.js en Node.js con JavaScript).
En esta Api se desarrolló un endpoint.
Url base: https://mail-maputo.herokuapp.com/

- Post /sendMail: recibe el cuerpo del mensaje en el body y se encarga de enviar los correos. La parte frontEnd se encarga construir todo el correo y envía una plantilla dependiendo de la funcionalidad que se requiera:
    - Notificar a un usuario ayuntamiento que se ha dado de alta una incidencia
    - Notificar usuario ayuntamiento/institución que se ha modificado una incidencia
    - Notificar usuario institución que se ha resuelto con éxito una incidencia

9.5.4 Desarrollo interfaz USSD

El desarrollo de la interfaz USSD ha sido una de las funcionalidades más críticas a lo largo del desarrollo del sistema. La búsqueda de información para poder desarrollar la interfaz consumió muchas horas de trabajo.

En primera instancia, esta tarea la tenía asignada el alumno Oriol Fort, pero a medida que pasaba el tiempo, la no obtención de resultados hizo que el alumno Eynar Arnez se involucre en en la búsqueda de información.

En la búsqueda, se encontró African’s Talking, un SDK encargado de simular la interfaz USSD. Pero no tuvo éxito la implementación del servicio African’s Talking, ya que influyó, el desconocimiento y la poca información de la herramienta.

Finalmente, se decidió desarrollar toda la lógica en una Api en Java y para poder simular la interfaz un ejecutable, el cual se encargará de simular la comunicaciones entre el ciudadano y el sistema para poder dar de alta una incidencia.

Esta decisión se toma para que en futuros evolutivos, la integración de la interfaz USSD, solo sea la búsqueda de información (pedir ayuda a los colaboradores en Maputo) ya que la lógica ya esta programada.

10. Pruebas

10.1. Pruebas FrontEnd

Pruebas de la interice del usuario

Para asegurarnos de que lo que estamos programando cumplen con unos criterios, se decidió realizar pruebas unitarias a los componentes que se iban creando.

La herramienta de testeo que se ha utilizado ha sido, Jasmine, que es la que Angular utiliza por defecto. Siendo una ventaja utilizarla, ya que no es necesario realizar ningún paso o configuración adicional para habilitar las pruebas unitarias.

Es importante saber que en Angular, los componentes para que tengan su prueba, han de usar la nomenclatura estándar , 'nombre archivo' + '.spec'.

Ejemplo:

Componente: app.component.ts

Test Unitario: app.component.spec.ts

Esto nos servirá para que cada componente cumpla los requisitos funcionales y de cara a posibles actualizaciones del sistema, se tenga mayor control de cada componente.

Pruebas con usuarios reales

Estaba previsto realizar pruebas de usabilidad del sistema con ciudadanos de Maputo.

En el viaje que tenía previsto realizar Oriol Fort, realizar varias sesiones de usabilidad con usuarios que no conocían el sistema para recoger datos útiles de cara a los siguientes evolutivos, pero no se ha podido realizar esta parte debido al Covid-19, pero igualmente se ha realizado unas pruebas de usabilidad, no a la escala que se quería, pero temas de seguridad, los test los hemos realizados los miembros de del proyecto.

Las pruebas que el usuario ha realizado: modo web y mobile

Página pública:

- Navegación menú principal
- Añadir incidencia
- Navegar en la pantalla búsqueda incidencia
    - Acceder al detalle de la incidencia
    - Buscar una incidencia con filtro
    - Navegar en el mapa
- Navegar en la pantalla Como funciona

Página privada

- Navegación menú principal
- Añadir incidencia
- Navegar en la pantalla búsqueda incidencia
    - Acceder al detalle de la incidencia
    - Buscar una incidencia con filtro
- Iniciar Sesión (administrador/institución)
- Añadir Usuario
- Acceder pantalla Instituciones
- Información detalle usuario
- Realizar Ajustes como administrador (estado, categoría)
- Actualizar/eliminar incidencia, dependiendo los permisos de usuario
- Actualizar/eliminar institución, dependiendo los permisos de usuario

Una vez realizada las pruebas, se pudo obtener feedback para poder mejorar el aplicativo en el transcurso del desarrollo y de cara a las siguientes versiones.

Ejemplo:

- Color del aplicativo
- Color, fuente de los textos
- Tamaño, color de los botones
- Formato del menú principal
- Notificaciones del aplicativo
- etc

11. Seguimiento del proyecto

11.1 Planificación

Al iniciar el proyecto, se encomendó realizar una planificación temporal, catalogada como una tarea principal. Esta planificación la cogeria el alumno como hoja de ruta, pero en el transcurso del proyecto, se produjeron cambios importantes a causa de estos factores:

- El Covid 19, sería el imprevisto/contingencia más importante, el cual ha influido directa o indirectamente en los demás factores.
- La instalación/configuración de un servidor en las oficinas de la fib, se tuvo que cambiar por un servidor en la nube. Este trabajo ha requerido más tiempo del esperado.
- Han surgido nuevos casos de uso que no estaban estaban previstas inicialmente, que se detallarán en el apartado 11.2.1
- Modificación de casos de uso, con la finalidad de adecuar mejor el sistema
- Estaba previsto el viaje de un alumno a Maputo para enseñar/testear el sistema. Al no poder realizar el viaje, no se pudo realizar el testeo de la usabilidad del sistema con los ciudadanos de Maputo como estaba previsto.
- USSD, desarrollarlo llevó más tiempo del que se tenía previsto.

En la planificación inicial, se marcó dos fechas claves. 17 de febrero, fecha de inicio y 22 de junio, fecha de finalización del proyecto (fin de desarrollo y documentación de la memoria). Por lo cual, la duración del proyecto era de tres meses.

A finales de mayo, se tomó la decisión de alargar el proyecto y presentar el TFG en el turno de otoño del 2020. Esta decisión se tomó para cumplir todos los objetivos del proyecto, tener más tiempo, era un recurso necesario. Porque en caso contrario, se tendría que haber quitado funcionalidades al sistema. Con todos estos factores, vimos que alargar el proyecto era la mejor opción.

11.2 Casos de usuario

Al seguir la metodología cascada incremental, vimos que teníamos que ajustar y realizar modificaciones en el sistema. Ya que, al tratarse de un prototipo, había algunas funcionalidades que se vieron alteradas respecto a la planificación inicial. Con la supervisión del Director, se fueron añadiendo, modificando o eliminando casos de uso y a la vez priorizando cada una de ellas, con el objetivo de conseguir un sistema robusto.

Al iniciar el proyecto, una de las funcionalidades pensadas por el equipo (director y los dos alumnos), era que el usuario pueda tener su cuenta.
Que el usuario tenga su perfil, le daba más interacción con el sistema, ya que así podría tener el seguimiento de sus incidencias pero, se vio que era más prioritario tener el perfil de institución (servicio externo) y se decidió desarrollar este nuevo perfil.

Se vio que era más necesario tener este perfil, ya que, este tendría más interacción con el sistema, sería el encargado de solucionar las incidencias. Por lo cual, se pensó que para este primer prototipo, vendría mejor tener este rol. El no tener el perfil ciudadano, no tiene impacto en el sistema, ellos podrán interactuar y crear las incidencias como se tenía previsto, consultar el estado de sus incidencias.

Para ello se decidió crear dos accesos al aplicativo:

- Acceso a la web pública: Esta web pública, será visible a los ciudadanos, los que podrán interactuar con el sistema.
- Acceso a la web privada: Acceso solo para el personal del ayuntamiento y las instituciones, en el que se les permitirá loguearse. Al loguearse, tendrán todas las funcionalidades que le permita su perfil.

Se espera que para futuros evolutivos(TFGs), se pueda añadir el perfil ciudadano.

11.2.1. Casos de uso añadidas respecto a la planificación inicial

- Añadir perfil usuario ayuntamiento

Caso de uso	#21 Dar de alta un usuario Institución
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado
Disparador	El usuario quiere dar de alta un usuario perfil Institución
Escenario principal	1. El administrador rellena los datos de nuevo usuario que quiere dar de alta 2. El sistema valida las credenciales 3. El sistema da de alta el nuevo usuario 4. El sistema notifica que se ha dado de alta el nuevo usuario
Extensiones	1. El sistema informa los datos incorrectos

- Añadir/editar una localización

Caso de uso	Alta de una nueva localización (casa)
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado
Disparador	El usuario quiere añadir una nueva localización
Escenario principal	1. El usuario accede a la pantalla de edición de localización 2. El usuario rellena los datos para dar de alta una nueva casa 3. El sistema valida la nueva localización 4. El sistema de de alta la nueva localización 5. El sistema notifica al usuario que se ha dado de alta correctamente la nueva localización
Extensiones	2. El sistema informa los datos incorrectos

Caso de uso	Editar de una nueva localización (casa)
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado
Disparador	El usuario quiere añadir una nueva localización
Escenario principal	1. El usuario accede a la pantalla de edición de localización 2. El usuario selecciona la casa que quiere editar 3. El usuario edita la localización 4. El sistema valida y edita la localización 5. El sistema notifica al usuario que se ha editado correctamente la localización
Extensiones	3. El sistema informa los datos incorrectos

- Añadir/editar/eliminar estados

Caso de uso	Alta nuevo estado
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado
Disparador	El usuario quiere dar de alta un nuevo estado
Escenario principal	1. El usuario accede a la pantalla de edición de estados 2. El usuario rellena los datos para dar de alta un nuevo estado 3. El sistema valida y da de alta el nuevo estado 4. El sistema notifica al usuario que se ha dado de alta correctamente el nuevo estado
Extensiones	2. El sistema informa los datos incorrectos

Caso de uso	Editar de un estado
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado - El estado tiene que existir
Disparador	El usuario quiere añadir editar un estado
Escenario principal	1. El usuario accede a la pantalla de edición de estados 2. El usuario selecciona el estado que quiere editar 3. El usuario edita el estado 4. El usuario edita el estado 5. El sistema valida y edita el estado 6. El sistema notifica al usuario que se ha editado correctamente el estado
Extensiones	2. El sistema informa los datos incorrectos

Caso de uso	Eliminar un estado
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado - El estado tiene que existir
Disparador	El usuario quiere eliminar un estado
Escenario principal	1. El usuario accede a la pantalla de edición de estados 2. El usuario selecciona el estado que quiere eliminar 3. El sistema elimina el estado 4. El sistema notifica al usuario que se ha eliminado correctamente el estado
Extensiones	2. El sistema notifica que no se puede eliminar el estado seleccionado

- Añadir/editar/eliminar categorías

Caso de uso	Alta nuev categoría
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado
Disparador	El usuario quiere dar de alta una nueva categoría
Escenario principal	1. El usuario accede a la pantalla de edición de categorías 2. El usuario rellena los datos para dar de alta una nueva categoría 3. El sistema valida y da de alta la nueva categoría 4. El sistema notifica al usuario que se ha dado de alta correctamente la nueva categoría
Extensiones	2. El sistema informa los datos incorrectos

Caso de uso	Editar de una categoría
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado - El estado tiene que existir
Disparador	El usuario quiere añadir editar una categoría
Escenario principal	7. El usuario accede a la pantalla de edición de categorías 8. El usuario selecciona la categoría que quiere editar 9. El usuario edita la categoría 10. El sistema valida y edita la categoría 11. El sistema notifica al usuario que se ha editado correctamente la categoría
Extensiones	3. El sistema informa los datos incorrectos

Caso de uso	Eliminar una categoría
Actor principal	Perfil administrador
Precondiciones	- El administrador tiene que estar logueado - La categoría tiene que existir
Disparador	El usuario quiere eliminar una categoría
Escenario principal	5. El usuario accede a la pantalla de edición de categorías 6. El usuario selecciona la categoría que quiere eliminar 7. El usuario elimina la categoría 8. El sistema notifica al usuario que se ha eliminado correctamente la categoría
Extensiones	3. El sistema notifica que no se puede eliminar la categoría seleccionado

11.2.3. Casos de uso modificados

- perfil usuario

En la planificación inicial se decidió implementar el perfil usuario ciudadano con cuenta en el aplicativo, pero vimos que había una mayor necesidad de que el perfil usuario institución sea desarrollado, por ello se modificaron los casos de uso de registro/modificación de un perfil usuario en el sistema

11.2.3. Casos de uso pendientes para siguientes evolutivos

- Perfil usuario ciudadano

No se implementó el perfil ciudadano para este evolutivo, pensado que era más prioritario tener el perfil institución, ya que este nuevo perfil institución tendría más interacción con el sistema.

Es decir, comparando un usuario ciudadano de Maputo con un usuario perfil institución, la probabilidad de que un ciudadano interactúe con el sistema es menor, que la de un usuario institución. El ciudadano depende de que la incidencia esté en su cuarterao. En cambio, un perfil institución tendrá más probabilidad de utilizar el sistema porque, le pueden asignar incidencias a resolver o incluso, crear una incidencia.

En base a éste criterio, se decidió no implementar el perfil ciudadano para este primer prototipo

11.3 Ejecución tiempo real

Como se decidió hacer una prórroga del proyecto, la estimación temporal que se hizo a principio del proyecto tuvo cambios importantes. Se decidió añadir dos iteraciones de 60 horas en el proyecto.

La decisión de añadir dos iteraciones en el proyecto, se basó en los backlogs que teníamos pendiente por desarrollar a finales de mayo (toma de decisión del turno de lectura del TFG).

Estimando el ritmo de trabajo y los casos de uso pendientes en el sistema, se valoró que era necesario invertir aproximadamente 120 horas.

Etapa	Descripción	Tiempo Aprox	Alumno
Gestión del proyecto	Contexto y alcance Planificación Presupuesto y sostenibilidad Documento final	75h	Eynar
		75h	Oriol
Iteración 1	Análisis y diseño	75h	Eynar
		75h	Oriol
Iteración 2	Implementación FrontEnd Pruebas	55h	Eynar
	Implementación BackEnd Pruebas	55h	Oriol
Iteración 3	Implementación FrontEnd Pruebas	55h	Eynar
	Implementación BackEnd Pruebas	55h	Oriol
Iteración 4	Implementación FrontEnd Pruebas	55h	Eynar
	Implementación BackEnd Pruebas	55h	Oriol
Iteración 5	Implementación FrontEnd Pruebas	55h	Eynar
	Implementación BackEnd Pruebas	55h	Oriol
Iteración 6	Implementación FrontEnd Pruebas	60h	Eynar
	Implementación BackEnd Pruebas	60h	Oriol
Iteración 7	Implementación FrontEnd Pruebas	60h	Eynar
	Implementación BackEnd Pruebas	60h	Oriol
Iteración 8	Implementación Pruebas del sistema y pruebas de usabilidad	60h	Eynar
	Implementación Pruebas del sistema y pruebas de usabilidad	60h	Oriol
Memoria y presentación oral	Redacción de la memoria Preparación de la defensa oral	50h	Eynar
	Redacción de la memoria Preparación de la defensa oral	50h	Oriol
Total		1200h	

Tabla 10.- Estimación en horas de las tareas del proyecto

11.4 Diagrama de Gantt tiempo final

Imagen 39 - Diagrama Gantt tiempo real

A las iteraciones 7 y 8 se les asignaron largos periodos de tiempo (un mes cada uno) en comparación a las iteraciones anteriores pero con la misma carga de trabajo que las otras iteraciones. Esto fue porque en medio se acercaba el periodo de vacaciones(Agosto).

11.5. Trabajo realizado

Gestión del proyecto

Iteración en la que se realizó el curso de GEP, se realizaron 4 entregables

Iteración 1

Realización del análisis del sistema a la vuelta del viaje a Maputo del director de forma conjunta

- Requisitos funcionales y no funcionales
- Casos de uso del sistema
- Búsqueda de información para instalar el servidor

Iteración 2

Iteración crítica, ya que se modificaron todos los planes debido al Covi-19. Se replanteó y tomamos la decisión de optar por un servidor en la nube (Amazon Web Services) debido a la imposibilidad de desplazamiento y acceso a la oficina.

Definición del diagrama de clases y creación de los proyectos en las partes del frontEnd y backEnd. Y el comienzo del desarrollo en la parte frontEnd, en esta iteración se comenzó con el diseño de las diferentes pantallas que tendría el aplicativo

Iteración 3

Implementación del menú de navegación responsive, ya que tendrá diferente diseño debido al mejor aprovechamiento de espacios según el dispositivo. A continuación, primera conexión entre el backend Y frontEnd con el endpoint de creación incidencia y la realización de la pantalla de reportar incidencia, el cual incluía el formulario y la integración de la Api de google maps y finalizó con la pantalla de como funciona, siendo una pantalla estática.

Se canceló el viaje de Oriol Fort a Maputo, lo cual cambió la previsión de realizar las pruebas de usabilidad y formar a personas en Maputo, por ello se decidió que las pruebas las realizaremos entre los miembros del equipo.

Iteración 4

Implementación de la pantalla búsqueda de incidencia y además la visualizar/editar/eliminar incidencia.

Búsqueda de información para el desarrollo de la interfaz USSD.

Iteración 5

Iteración en la que se decidió la lectura de la memoria en Octubre
Se implementó la pantalla Institución, en la cual estaban las funcionalidades de visualización/edición/eliminación de una institución y se decidió añadir el perfil institución en el sistema.

Iteración 6

Implementación de las funcionalidades de ajustes del sistema

- Añadir/Editar/Eliminar estado
- Añadir/Editar/Eliminar categoría
- Añadir/Editar/Eliminar localización

Iteración 7

Implementación de la Api de notificaciones e integración en el sistema. El sistema tiene tres plantillas de correos:

- Notificación de creación de una incidencia
- Notificación edición de una incidencia, estos correos, los podrán realizar los usuarios ayuntamiento e institución
- Notificación resuelta correctamente

Iteración 8

Pruebas de integración y preparación de pruebas de usabilidad del sistema por parte del director. Lo cual llevó a realizar ajustes en temas de diseño y correcciones en temas funcionales del sistema.

Memoria y presentación oral

Los alumnos realizarán de forma individual sus respectivas memorias y preparación de la defensa oral ante el tribunal.

11.6. Presupuesto

Hubo un incremento en el presupuesto respecto a la previsión que se había realizado al iniciar el proyecto, esto fue por:

- Recursos humanos: Al cambiar la fecha de entrega se tuvo que invertir en tiempo de programación de los alumnos. Los roles que se han utilizado en las dos iteraciones son: Desarrollador (100h) Responsable pruebas(10h)

- Recursos de software: Al no poder tener un servidor propio debido a que no se pudo instalar como se tenía previsto. Se tuvo que buscar otros medios para publicar el sistema, en este caso, un servidor en la nube.
    - Los servidores en la nube que se buscaron, fueron gratuitos o de suscripción gratuita el primer año
- Recursos indirectos: Incrementar el tiempo del proyecto, requiere de utilizar los recursos indirectos en dos meses adicionales. Eso implica 60€(luz) + 80€ (internet) con un total de 140€ incremento respecto a los costes indirectos iniciales

Después de todos los factores que se vieron implicados, el presupuesto final de proyecto es:

Recurso	Coste (€)
Recursos humanos	15.360
Recursos hardware	187,5
Recursos software	0
Costes indirectos	350
Total	**15.897,5 €**

Tabla 11 - Coste del proyecto

11.7 Satisfacción de los objetivos

De los objetivos marcados inicialmente podemos concluir que se han llevado a cabo con éxito

- *El sistema debe permitir dar de alta incidencias: Se ha conseguido implementar el prototipo y se encuentra disponible para todos los usuarios*
- *El sistema debe permitir consultar incidencias: La webApp permite realizar búsquedas a todos los usuarios*
- *El usuario puede editar/eliminar una incidencia: La webApp permite realizar las modificaciones a los usuarios ayuntamiento/institución*
- *El sistema no puede dar de alta fuera de la superficie de Maputo: El sistema no permite dar de alta fuera de la zona de Maputo.*

11.8 Satisfacción de los requisitos

Se han satisfechos los siguientes requisitos

Requisitos funcionales:

- Usuario ayuntamiento/institución: Acceso a la cuenta
    - Iniciar sesión/Cerrar sesión
    - Registro de nuevos perfiles
    - Restablecer contraseña
    - Editar datos perfil
- Usuario Ciudadano/Ayuntamiento/Institución: Gestión incidencias
    - Crear una incidencia: El sistema tiene que permitir dar de alta una incidencia
    - Buscar una incidencia: El sistema tiene que permitir realizar la búsqueda de incidencias, ya sean las incidencias que muestra el sistema o realizando una búsqueda específica(filtro de estado, categoría, gravedad, rango de fechas o número de incidencia)
- Usuario Ayuntamiento/Institución: Gestión incidencias
    - Editar una incidencia
    - Notificación de una incidencia

- Usuario Ayuntamiento: Gestiones super administrador
    - Crear una institución
    - Editar/Eliminar institución
    - Añadir/editar una localización
    - Añadir/editar/eliminar estados
    - Añadir/editar/eliminar categorías

Requisitos no funcionales:

- El sistema tiene que estar disponible en español
- El sistema no tiene que requerir de ninguna formación previa.
- La aplicación tiene que estar siempre operativa
- El usuario administrador necesitará un usuario y contraseña
- Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.
- El sistema tiene que tener la misma base que la aplicación Mopa
- Se tiene que poder acceder al sistema desde cualquier plataforma
- El sistema no se tiene que instalar
- El servidor y componentes deben ser portables en plataformas GNU/Linux y Windows, con máquinas que presentan arquitecturas de 64 bits
- Las interfaces de comunicación deben contener los estándares Web y fundamentalmente se deben basar en protocolos HTTP
- El sistema debe proporcionar mensajes de error que sean informativos y orientados al usuario final.
- El sistema debe contar con manuales de usuario

Queda realizar una mejora en este requisito no funcional: *El sistema no debe tardar más de diez segundos en mostrar los resultados de una búsqueda*, ya que ahora mismo en la petición http de búsqueda de incidencia, nos acercamos al límite establecido.

Teniendo como referencia el listado de este apartado, se puede observar que han alcanzado las bases fijadas inicialmente en el apartado de requisitos del sistema.

12. Conclusiones

El trabajo realizado en este proyecto está orientado en la obtención de un producto, el cual pretende solventar una necesidad, por lo cual, las conclusiones se basarán en la progresión y el aprendizaje del alumno durante la ejecución de dicho sistema.

12.1 Conclusiones personales

Se ha desarrollado el sistema con el objetivo de dar soporte a la ciudad de Maputo, siendo capaz de gestionar las incidencias que se van produciendo en esta comunidad. Cabe remarcar que, en la realización del proyecto, se han ido encontrando distintos problemas (requisitos del sistema) que con la colaboración del director se han resuelto satisfactoriamente.

El haber escogido la parte frontEnd me ha resultado realmente gratificante, ya que me ha servido para afianzar los conocimientos previos del framework de Angular que disponía antes de iniciar el proyecto, darme cuenta que es un sector del cual me quedan muchísimas cosas por aprender y sacar partida.

Lamentar que, de todos los nuevos conocimientos adquiridos a lo largo de la implementación, no haber sacado más partida a algunas herramientas utilizadas; por ejemplo, el descubrimiento tardío de Firabase, ya que dispone de muchas funcionalidades que podrían haber sido útiles en el sistema.

Y a nivel personal, darme cuenta de que uno de los puntos donde he de mejorar bastante, es en el apartado del diseño de una aplicación. Determinar cómo se tendría que ver el aplicativo fue un reto constante al cual me tuve que enfrentar a lo largo del proyecto.

Otro punto a destacar, es el hecho de que los TFG’s se suelen hacer individualmente. En nuestro caso no fue así, lo desarrollamos dos alumnos, lo cual también suponía un reto personal, ya que las decisiones que se tomaban, adecuarse al ritmo de trabajo, y otros factores, son circunstancias a tener en cuenta. Después también estaba el hecho de tener un rol cliente (director) bastante definido, el cual validaba las funcionalidades del sistema en el transcurso del proyecto. Con todos estos factores, se podría decir que, esta experiencia vivida se acerca al trabajo que realiza un programador en el ámbito laboral.

12.3 Trabajo futuro

Una vez finalizado este primer prototipo del sistema, la dependencia del Covid-19 ha sido enorme, ya que, la imposibilidad de que un alumno o el director no se hayan podido

desplazar a Maputo. Ya que era fundamental tener el feedback de la usabilidad del sistema pero no se pudo obtener en este primer prototipo.
Por lo cual, un primer punto de partida de cara a los siguientes evolutivos, sería las pruebas de usabilidad que quedaron pendientes en Maputo para ver si se cubren todas las necesidades. Una vez se tenga el feedback, realizar el nuevo alcance del sistema en función a las necesidades de la Ciudad.

13. Referencias

[1] Mopa [en línea]. [Consultado 16 de marzo 2020]. Disponible en internet: https://www.mopa.co.mz/

[2]USSD [en línea]. [Consultado 16 de marzo 2020]. Disponible en internet: https://es.wikipedia.org/wiki/USSD

[3] compromisoempresarial [en línea]. [Consultado 16 de marzo 2020]. Disponible en internet: https://www.compromisoempresarial.com/transparencia/2018/02/las-10-mejores-apps-que-impulsan-la-participacion-ciudadana-y-la-transparencia/

[4] “gitHub” [En liniea] [Consultado 16 de marzo 2020]. Disponible en internet: https://es.wikipedia.org/wiki/GitHub

[5]. ganttpro [en línea]. [Consultado 01 de marzo 2020]. Disponible en internet: <https://app.ganttpro.com/>

[6]. tecnoempleo [en línea]. [Consultado 09 de marzo 2020]. Disponible en internet: <https://www.tecnoempleo.com/informe-empleo-informatica.php/>

14. Anexo

14.1 Anexo 1: Documentación técnica - frontEnd

Arranca proyecto local

En la planificación inicial se acordó con el Director del proyecto realizar una documentación técnica, la cual facilite a los informáticos de Maputo o al alumno que tenga la intención de continuar con el proyecto.

Los requisitos mínimos de la persona, es tener nociones básicas de HTML, CSS y JavaScript y saber el significado del patrón de diseño MVC para sacar partido al framework de Angular.

En caso de no haber trabajado o no dominar el framework de Angular, tenemos la ventaja de que es un framework que tiene mucha documentación en Internet. Esto puede llegar a ser beneficioso, ya que en caso de duda, las posibilidades de consultar o encontrar la solución aumenta. Cabe destacar que en la página principal de angular dispone de bastante información útil para el programador. [https://angular.io/]

El primer paso a realizar es la instalación del Node.js [https://nodejs.org/es/].

Descargar el ejecutable y realizar la instalación de node.js

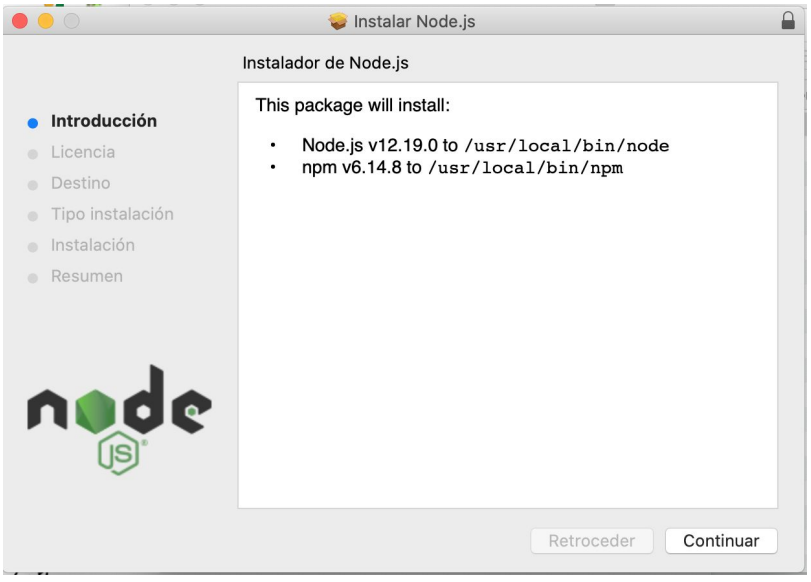


Imagen 40- Instalación node.js

Una vez se hayan seguido los pasos, comprobar que se ha instalado con éxito

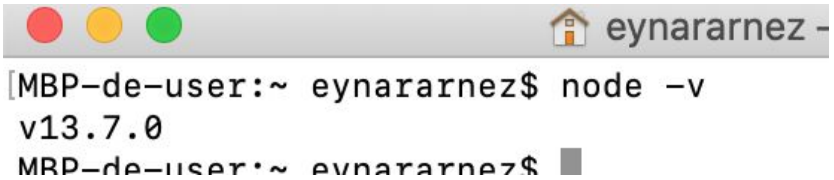


Imagen 41- versión node

Una vez instalado node.js, dispondremos del sistema de gestión de paquetes (npm) que nos servirá para realizar las demás instalaciones.

El siguiente paso es abrir una terminal para realizar la instalación de Ionic Cli y Angular Cli

- npm i @ionic/angular@4.11.2
- npm i @angular/cli@8.3.23

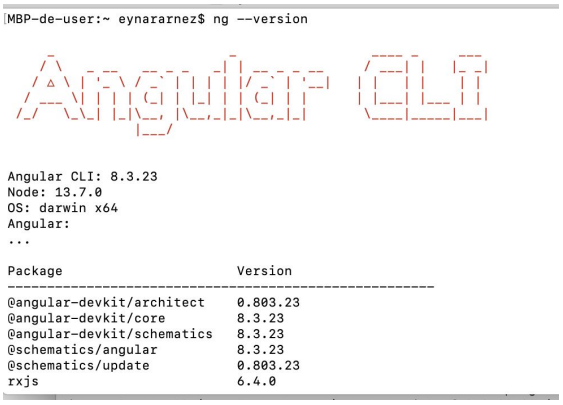


Imagen 42 - Versión angular

Es importante realizar la descarga de estas versiones, ya que en caso de utilizar versiones superiores, el sistema puede llegar a tener comportamientos rayo o en el peor de los casos, no poder arrancar el proyecto en local.

El siguiente paso es descargar el proyecto, que se encuentra en el repositorio de git:

- https://github.com/aaeynar/maputo

Ponerse en la carpeta del proyecto y ejecutar

- npm install

Ejecutar este comando significa la descarga de todas las librerías(node_modules) necesarias para que se pueda arrancar el aplicativo en local

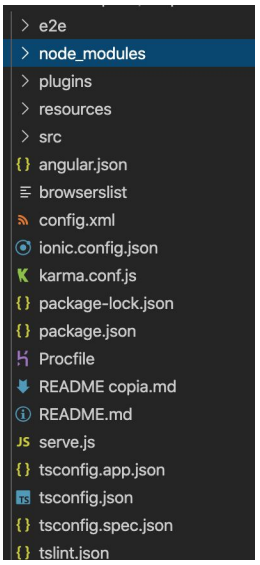


Imagen 43 - Carpeta node_modules

El último paso es arrancar lo en local con el comando

- ionic serve

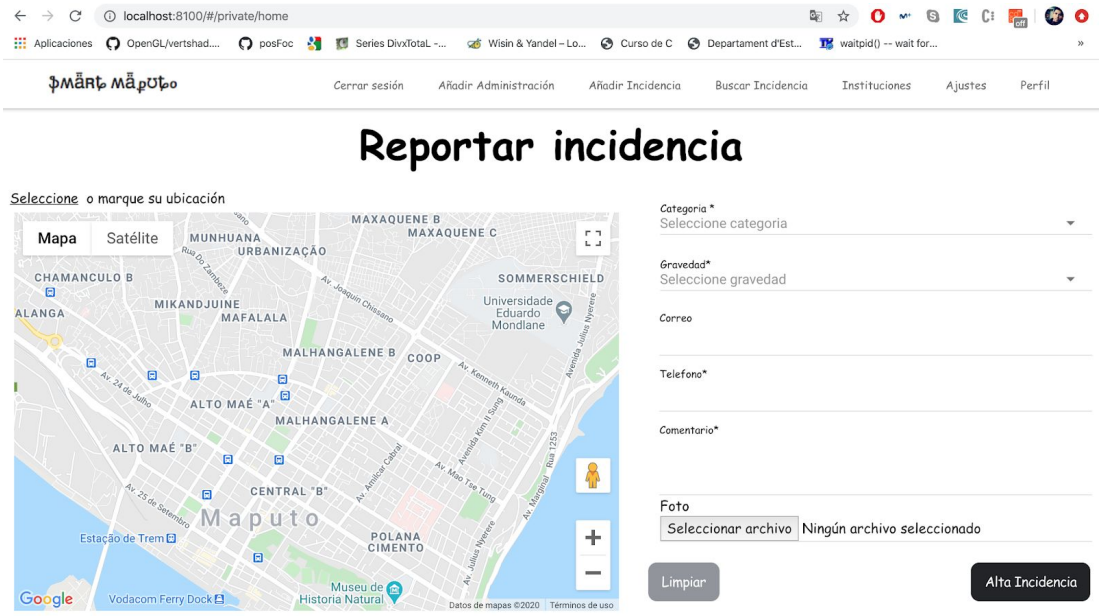


Imagen 44 - Proyecto local

Build webApp

Una vez se haya conseguido arrancar el proyecto en local, el siguiente paso es generar un código transpilado que sea útil para poder subir nuestra app a un servidor. El término transpilar significa: teniendo nuestro código en TypeScript generamos un código en otro lenguaje JavaScipt, es decir, el programa que tenemos produce otro programa en otro lenguaje cuyo comportamiento es el mismo que el original

Para poder construir estos transpilados hemos decidido utilizar Apache Cordova, que se encarga de ofrecer aplicaciones híbridas, esto significa que no es una aplicación móvil nativa ni una página web ya que tiene acceso a las APIs nativas del dispositivo.
.
Estas versiones se han ido generado a lo largo del desarrollo del proyecto (versiones del sistema, ya sea arreglando o añadiendo nuevas funcionalidades), pero la intención final es realizar una guía de uso para que los programadores de Maputo o estudiantes de la FIB que quieran seguir con el proyecto.

Como se ha dicho, Apache Cordova, ofrece aplicaciones híbridas, eso quiere decir que se pueden construir los transpilados para poder añadir nuestra aplicación tanto a Play Store como Apple Store, pero el alcance de nuestro proyecto se estimó en subir la aplicación a un servidor y poder visualizarla en una página web.

[https://ionicframework.com/docs/cli/commands/cordova-build]

Para añadir Cordova, en la terminal ejecutar

- npm i cordova-browser

Una vez haya terminado de descargar las librerías y lo tengamos en nuestro proyecto, hemos decidido añadido el script *ionic cordova platform add browser && ionic cordova build browser --prod* en el packaje.jon para generar los traspilados

Imagen 45 - package.json

Se ha decidido realizar este script para que con un solo comando en la terminal, seamos capaces de crear la carpeta browser y hacer el build de producción

- npm run browser

Este comando se encarga de crear las carpetas platforms y www dentro de la carpeta raíz como se puede observar en la siguiente imagen: tenemos todos los ficheros y se han creado estas dos nuevas carpetas

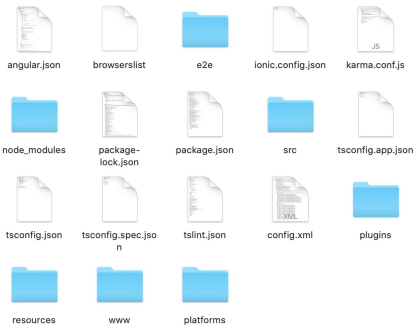


Imagen 46 - Carpeta raíz proyecto

- Contenido de la generación del código transpilado dentro de la carpeta www

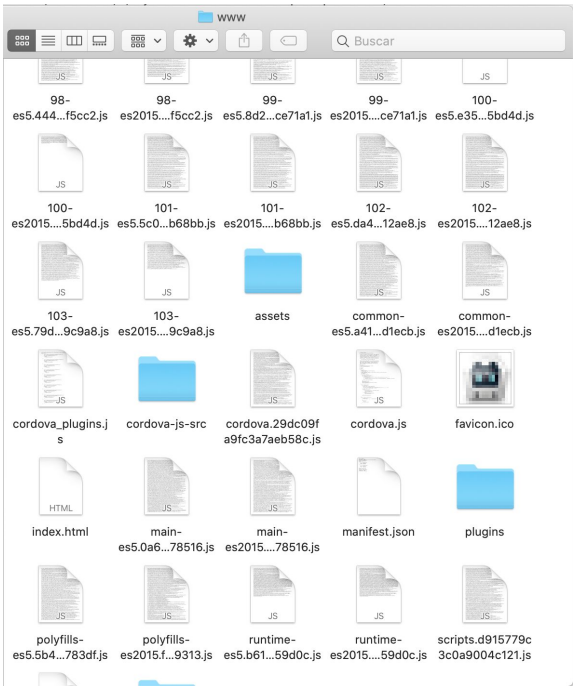


Imagen 47 - Carpeta www transpilados generados

Una vez tengamos estos minimizados estamos en disposición de subir al servidor (Amazon Web Services) que deseemos