UNIVERSIDAD DE CASTILLA - LA MANCHA

ESCUELA DE INGENIERÍA INDUSTRIAL DE TOLEDO

TRABAJO FIN DE GRADO Nº 19-B-225124

SUPERVISIÓN DE OPERACIONES DE MÁQUINAS DE CAFÉ

Autor:

Julián Lozano Moraleda

Director:

Francisco Moya Fernández

Junio 2019

Julián Lozano Moraleda
UCLM— Escuela de Ingeniería Industrial
Campus Universitario de la Real Fábrica de Armas
45071 Toledo – Spain
E-mail: Julian.lozano.92@gmail.com
Web site: https://github.com/Julian92UCLM/cosmos

Agradecimientos

Quiero mostrar mi agradecimiento a la universidad de Castilla la Mancha, a la escuela de Ingeniería Industrial del campus de Toledo, a los profesores y a todos los compañeros que, a lo largo de este periodo tan importante de mi vida, ha formado parte del proceso de aprendizaje, tanto en lo académico como en lo personal.

Agradecer Francisco Moya Fernández que acepto ser mi tutor y me ha ayudado a hacer posible este trabajo, gracias por su ayuda y consejo en este proyecto.

Gracias a mi familia y amigos por su apoyo y comprensión a lo largo de estos meses de trabajo, su aliento nunca me ha faltado.

Gracias a mis compañeros de trabajo que me ayudaron a compaginar el trabajo con la preparación de este proyecto, sin su ayuda tampoco habría sido posible.

Gracias a todos.

A mi familia y a todas las personas

que me han apoyado y han hecho

que el trabajo se realice con éxito.

Resumen

Un problema muy habitual que tienen los usuarios de máquinas expendedoras o de vending es que cuando ocurre algún problema con el servicio de estas máquinas no encuentran a nadie para que les ayude a resolver su incidencia. Únicamente podrán encontrar un numero de atención al cliente al que podrán llamar para avisar de que la maquina no funciona adecuadamente ya sea porque haya sufrido un fallo al servir la bebida, no haya devuelto bien el cambio o directamente no se encuentre operativa.

Este trabajo trata resolver el problema creando una plataforma compuesta por una aplicación, una página web y una base de datos. La aplicación se ha creado utilizando App Inventor y permite que el usuario pueda ingresar las incidencias ocurridas en la base de datos. Esta aplicación estará subida a un repositorio y podrá descargarse desde cualquier máquina de café, ya que se colocará en todas las maquinas un código QR que redireccionará al usuario mediante un enlace al sitio donde se encuentra guardada para poder instalarla. La base de datos nos permitirá almacenar todos los datos relacionados con las incidencias, esta estará compuesta por hojas de Google Sheets y estará gestionada mediante scripts de Google. La página web se encargará de monitorizar todo el sistema ya que contendrá los datos más destacables de las incidencias, así como estadísticas. Desde la página web también podremos ver el tiempo medio de resolución de las incidencias, este es un dato muy importante ya que medirá la calidad del servicio que estas máquinas ofrecen al cliente. Además, esta página también tendrá una sección en la que habrá un espacio desde el que se podrán anular las incidencias.

Las incidencias las registrará el usuario en la base de datos utilizando la aplicación. Este será notificado cuando se inicie el trámite de la incidencia mediante un correo en el que ira su nombre, el motivo de la incidencia, la fecha y un código para poder anular la incidencia y también una vez que se cierre. En todo momento el cliente podrá ver sus incidencias y el estado en el que se encuentran. Estas incidencias podrá anularlas el cliente desde la aplicación una vez vea que el problema ya está resuelto, ya que habrá una sección para hacerlo, también podrá cerrar las incidencias el personal de mantenimiento desde la página web una vez que haya realizado el mantenimiento, esta anulación se podrá hacer introduciendo el id de la máquina que haya revisado, ya que si y la haya dejado puesta a punto todas las incidencias relacionadas con esta máquina estarán resueltas.

La aplicación está formada por tres pantallas:

La primera pantalla contiene un formulario con el que el usuario podrá registrarse, en ella habrá que ingresar el nombre de usuario y la contraseña. Esta aplicación recordará los datos por lo que a menos que el usuario quiera registrarse con otro nombre o correo solo se tendrá que registrar una vez.

La segunda pantalla contiene un formulario para que el usuario pueda registrar su incidencia, en el aparecerán las incidencias más comunes para facilitar la resolución del problema, si la suya no se encontrara entre estas también encontrara un espacio en el que podrá expresar su incidencia. En este formulario también habrá que especificar en qué maquina hemos tenido la incidencia ya que sin ese dato será imposible resolver el problema, cada máquina tendrá un numero perfectamente visible que hará de id y será el número que habrá que introducir en el formulario de incidencias.

En la tercera pantalla se podrán realizar dos funciones, ver nuestras incidencias, y anularlas cuando el usuario lo estime oportuno. Esta pantalla nos permitirá ver todas las incidencias que se hayan registrado con el usuario que este registrado en ese momento. Al pulsar el botón nos aparecerá una tabla en la que aparte de aparecer los detalles de la incidencia también nos aparecerá el estado en el que se encuentra cada una, este podrá ser abierto o cerrado. Desde esta pantalla también se podrán anular las incidencias, hay un espacio dedicado a esta función, en el, el usuario podrá introducir el código que recibió por correo al iniciarse la incidencia, al enviar este código la incidencia se cerrará.

La base de datos contendrá tablas en las que se irán almacenando todos los datos del sistema. Habrá una tabla en la que se guardarán los datos de registro de los usuarios como nombre de usuario, correo o id del usuario, estos datos los enlazaremos con esta tabla mediante un formulario de Google. Otra tabla estará de dedicada a almacenar los datos de las incidencias como fecha, id de la incidencia, tipo de incidencia, cantidad si el tipo de incidencia fuera económico, maquina en la que ha ocurrido la incidencia, código y estado en el que se encuentra, estos datos al igual que los anteriores también los recibiremos mediante la ayuda de un formulario de Google. Encontraremos otra tabla en la que recibiremos el código mandado desde la aplicación, los almacenaremos en esta tabla para poder resolverlos. Al igual que recibimos el código de anulación para anular las incidencias desde la aplicación, también recibiremos en otra tabla el id de la máquina que el operario de mantenimiento haya introducido en la página web después de realizar su puesta a punto. Habrá otra tabla en la que calcularemos el tiempo medio en resolver las incidencias, en ella aparecerán los tiempos de inicio y final de incidencia que habremos mandado de las tablas de registro de incidencia y de la tabla de anulación, calcularemos el tiempo pasado entre ellas y se hará una media entre todas, este será el tiempo medio. También hemos creado otra tabla en la que se enlazaran los datos de las incidencias que se deban a una devolución errónea de las vueltas y que no estén resueltas, con los usuarios que registraron esas incidencias, esta tabla servirá para crear las etiquetas que se pondrán en los sobres en los que se devolverá el dinero a los usuarios afectados, estas etiquetas contendrán el nombre del usuario la fecha y la cantidad no devuelta por la máquina.

La gestión de la base de datos estará llevada por scripts de Google, estos archivos están compuestos por programas que se encargan de automatizar las acciones que queremos que realice el sistema.

Índice general

Capítulo 1
Introducción 13
1.1. Organización de la memoria 13
1.2. Repositorio de información 14
Capítulo 2
Motivación y antecedentes 15
Google Sheets como base de datos 16
Plataforma App Script para desarrolladores. 16
Uso y limitaciones de la plataforma de App Script. 16
Capítulo 3
Objetivos 17
3.1. Simplificación del proceso de reclamación ante fallos 17
3.2. Monitorización del servicio 18
Capítulo 4
Contribuciones 19
Capítulo 5
Procedimiento 21
5.1. Diferencias con Scrum 21
5.1.1. Roles 21
5.1.2. Historias de usuario 21
5.1.3. Planificación de sprints 22
5.1.4. Flujo de trabajo 22
5.1.5. Herramientas de ayuda 23
Capítulo 6
Resultados 25
6.1. App Inventor como mecanismo para crear aplicaciones para móvil 25
6.1.1. El entorno App Inventor 25
6.2. Modelo de base de datos 26
6.3. Aplicación dos formularios: identificación e incidencias. 27
6.4. Relación usuario id 33
6.5. Usuario recordado 34
6.6. Añadir opción de máquinas en la aplicación 36
6.7. No repetir usuario 38
6.8. Ver mis últimas incidencias desde la aplicación 42

6.9. Crear código en la App para anular incidencia..... 44
6.10. Crear lista de incidencias..... 44
6.11. Enviar correo electrónico al iniciar incidencia. 49
6.12. Cerrar incidencias desde la web y enviar correo electrónico 51
6.12.1 Enviar correo al anular incidencia 55
6.13. Cerrar incidencias desde la aplicación 58
6.14. Ver estadísticas desde la página web..... 61
6.15. Crear etiquetas para devolver el dinero 65
6.16. Obtener el tiempo medio de resolución de incidencias 67
6.17. Crear código QR para descargar aplicación 67
Capítulo 7
Discusión de resultados y trabajo futuro 69
Capítulo 8
Bibliografía 71

Índice de figuras

Modelo de base de datos 27
Formulario de Google para registro de usuarios e incidencias 27
Formulario de Google para registro de usuarios 28
Tabla de Google Sheets para registro de usuarios 28
Formulario de Google para registro de incidencias 29
Tabla de Google Sheets para registro de incidencias 29
Pantalla de registro de usuario 30
Pantalla de registro de incidencias 31
Código fuente del formulario de Google. URL 31
Código fuente del formulario de Google. Elementos 32
Bloques del botón registro de pantalla de registro de usuarios 32
Bloques del botón enviar, pantalla de registro de usuarios 33
Bloques de incidencias, pantalla de incidencias 33
Script crear id 34
Bloques del botón registro con TinyDB, pantalla de registro de usuarios. 35
Bloques de inicialización, pantalla de registro de usuarios 35
Pantalla de registro de usuarios con opción de cambiar de usuario 36
Bloques del botón de cambiar de usuario, pantalla de incidencias 36
Pantalla de incidencias, añadido introducir id de máquina 37
Bloque de botón enviar con máquina, pantalla de incidencias 37
Bloque botón enviar con máquina, pantalla de incidencias II 38
Tabla de incidencias con columna para máquinas 38
Pantalla de registro de usuario con botón validar 39
Bloque de botón validar, pantalla de registro de usuarios 39
Bloque respuesta Web, pantalla de registro de usuarios 40
Bloque enviar con máquina, pantalla de incidencias 41
Bloque iniciar PantallaTransicion, de la PantallaTransicion 41
Bloque respuesta Web, PantallaTransicion 42
Pantalla mis_incidencidas 43
Función SELECT 43
Bloque botón histórico, pantalla mis_incidencias 43
Bloque de botón enviar enviando código, pantalla mis_incidencias 44
Tabla de incidencias con columna código 44
Código del archivo index.html 45
Página web. Inicio 46
Tabla de incidencias de Awesome table 47
Tabla de Google Sheets con filtros para Awesome table 47
Tabla de incidencias de Awesome table 48
Código del archivo tablas.html 49
Script de Google Enviar correo 51
Trigger de función onEditEmvioMail 51
Código del archivo mantenimiento.html 53
Formulario página web, mantenimiento 53
Script, enviar id de máquinas 55
Tabla de máquinas actualizadas 55
Script de Google Enviar_correo_resuelto 57
Pantalla mis_incidencias con campo de texto y botón para anular incidencia 58
Bloque BotonAnular, pantalla mis_incidencias 59
Tabla de código recibido 59
Script anular_desde_aplicacion 61

Gráfico de incidencias más habituales *61*
Gráfico de máquinas con más incidencias *62*
Gráfico de incidencias en la última semana *62*
Gráfico de incidencias en la última semana II *63*
Código del archivo estadisticas.html *65*
Consulta para filtrar los datos de la tabla de formulario de incidencias *66*
Función VLOOKUP para filtrar añadir los datos de usuario *66*
Plantilla para tabla de incidencias *66*
Tabla de etiquetas, Awesome table *66*
Fecha inicial y final, tabla tiempo_medio *67*
Función MINUS *67*
Función AVERAGE *67*
Código QR de acceso a la aplicación *68*

Capítulo 1

Introducción

Este proyecto trata de resolver un problema habitual que ocurre con uno de los servicios que la escuela tiene para los estudiantes o trabajadores del centro.

La universidad licita un servicio de máquinas expendedoras de café, este es un servicio de vending que ofrece café y otras bebidas a estudiantes y trabajadores del centro.

Como todos los servicios, este tiene algún problema o punto de mejora, y es que al no servirle la bebida una persona, si durante el servicio ocurre algún problema como que no te sirva la bebida, no te ponga vaso o no te devuelva bien el cambio tendríamos que buscar por alguna parte el número del servicio técnico para que subsanen el problema y la máquina no continúe realizando mal el servicio o que nos devuelva el dinero.

Este trabajo resuelve el problema proporcionando una interfaz entre el cliente y el servicio de mantenimiento. Se trata de una aplicación que le ayudará a dar a conocer sus incidencias, esta puede descargársela el cliente en su smartphone de una manera cómoda. También ofrece una página web con la que la universidad podría controlar la calidad de servicio que ofrece en sus espacios.

1.1. Organización de la memoria

La organización de este documento es conforme al anexo TFG-08b de la Normativa Trabajo Fin de Grado de la Escuela de Ingeniería Industrial de Toledo, de la Universidad de Castilla-La Mancha, aprobada en Junta de Escuela el 23 de junio de 2016. Se descompone en los siguientes capítulos.

Capítulo 2 Analiza los antecedentes y estado del arte en relación con el tema del proyecto.

Capítulo 3 Enumera y justifica los objetivos del proyecto y establece los límites intrínsecos y extrínsecos de ejecución del TFG.

Capítulo 4 Enumera las principales contribuciones y aportaciones personales del autor en este TFG.

Capítulo 5 Describe la metodología de desarrollo empleada durante la ejecución del TFG.

Capítulo 6 Describe en detalle las pruebas realizadas y los resultados obtenidos.

Capítulo 7 Discute los resultados en relación con los objetivos del proyecto y recopila las principales conclusiones del proyecto.

1.2. Repositorio de información

Todo el material generado durante la ejecución de este proyecto está disponible en https://github.com/Julian92UCLM/cosmos. El repositorio incluye los archivos del presente documento, el código fuente de los programas realizados o modificados, y todos los datos generados en la evaluación de resultados.

Capítulo 2

Motivación y antecedentes

Un problema muy común que puede encontrar un cliente de una máquina expendedora de café es la falta de monedas suficientes en el interior de la maquina parar recibir el cambio correctamente.

Para este caso o para cualquier otra incidencia ocurrida con la maquina la solución proporcionada es llamar al número de servicio de mantenimiento para reclamar el problema.

Esta solución puede suponer muchas molestias para el cliente cuando la cantidad no devuelta por la maquina es pequeña.

Hoy existen muchas tecnologías que podrían proporcionar alguna solución más cómoda para el cliente ya que siempre se encuentra en una situación de desventaja.

Algunas de las tecnologías que estudiaremos su utilización en este proyecto:

Bluetooth: es un estándar global inalámbrico para una conectividad simple y segura. La tecnología Bluetooth soporta billones de productos electrónicos de consumo en todo el mundo. Consigue avanzar tanto en la automatización del hogar como en todo tipo de dispositivos.

Es una tecnología de conectividad inalámbrica de baja potencia utilizada para transmitir audio, transferir datos y difundir información entre dispositivos.

Wifi: es una de las tecnologías de comunicación inalámbrica mediante ondas más utilizada hoy en día. También conocida como WLAN(Wireless LAN, red inalámbrica) o estándar IEEE 802.11.

Además, el Wifi tiene algunas ventajas comparando con el bluetooth como, por ejemplo, la posibilidad crear una “red de dispositivos”, sin embargo, esto no es posible con el Bluetooth. Por eso, si usamos Bluetooth sólo será posible emparejar dispositivos por pares, mientras que, con una red Wifi, aunque tendremos que usar un router, podremos conectar una red de equipos a la vez.

La pega principal que encontramos en la tecnología Wifi está en el consumo de energía, que es bastante más elevado que en el caso del Bluetooth.

Código QR: es un código de barras bidimensional cuadrado que puede almacenar los datos codificados. La mayoría de las veces es usada para almacenar un enlace a un sitio web (URL).

Esta tecnología permite y acelera el uso de servicios web para móviles: se trata de una herramienta digital muy creativa.

Raspberry Pi: Raspberry Pi es una placa computadora de bajo precio, se puede decir que es un ordenador del tamaño de una tarjeta de crédito. El concepto es el de un ordenador sin accesorios. Está formada por una placa que se puede conectar a algunos dispositivos de ordenador y comportarse como tal.

Google Sheets como base de datos

Las hojas de Google evolucionan día a día ganando cada vez más poder convirtiéndose hoy en día en una de las mejores aplicaciones de hoja de cálculo.

Google Sheets tiene aplicaciones como Apps Script, Form Triggers, Google Sheets API, Timed Triggers, que lo convierte en una herramienta más completa si lo comparamos con Excel.

Hablaremos de las características de Google Sheets, las limitaciones que presenta esta aplicación, las soluciones alternativas que pueden utilizarse ante los problemas que se presentan y como usar Google Sheets como base de datos.

Plataforma App Script para desarrolladores.

Google contiene muchas aplicaciones integradas que pueden interactuar con Google Sheets App Script como MailApp , DriveApp, DocumentApp, SlidesApp, CalendarApp etc.

Hay muchas integraciones disponibles para trabajar con Google Sheets pero la mayoría de las funciones no se suelen usar debido a que muchos de los usuarios y desarrolladores ni siquiera saben todo lo que Google App Script puede proporcionarles.

Uso y limitaciones de la plataforma de App Script.

Como hemos dicho anteriormente, App script tiene muchas integraciones disponibles dentro de todo en entorno de Google. App Script es muy útil cuando deseamos compartir y modificar datos entre las aplicaciones de Google. Como, por ejemplo.

-Podríamos responder automáticamente a algún usuario cuando envíe un formulario de Google usando MailApp.

-Envío periódico a usuarios que se encuentren en una hoja de cálculo de Google con MailApp y Timed triggers.

-Sincronizar eventos entre hojas de cálculo de Google y Google Calendar.

Estos son solo algunos de los ejemplos entre todas las posibilidades que nos ofrecen las aplicaciones de Google y con la que podríamos montar una base de datos.

Capítulo 3

Objetivos

Los objetivos en este proyecto son dos:

3.1. Simplificación del proceso de reclamación ante fallos

3.1.1. Conseguir una solución a problemas reales que pueden ocasionarse en una máquina expendedora de café, como, no devolver el cambio correctamente al consumidor, que la máquina se encuentre fuera de servicio o algún otro fallo que observe el cliente de la máquina.

3.1.2. El consumidor afectado por un fallo de la máquina tendrá una interfaz a su disposición para poder comunicarse con el servicio de mantenimiento.

3.1.3. Se dará una solución sencilla, cómoda, moderna y con total garantías para el consumidor.

3.1.3.1. El sistema creado estará compuesto por el menor número de elementos posible.

3.1.3.2. Se podrá acceder a la plataforma que solucione el problema desde cualquier smartphone.

3.1.4. Ofrecer una plataforma en la que el consumidor afectado dé a conocer la incidencia al servicio de mantenimiento de la máquina con los mínimos tramites y datos personales necesarios.

3.1.4.1. Crear un formulario en el que el usuario podrá detallar de manera precisa el problema que ha tenido con la máquina expendedora.

3.1.4.2. Registrar de forma sencilla y automática el fallo que ha tenido la máquina, así como el nombre del cliente afectado y la fecha de la incidencia.

3.1.4.3. Almacenar todos los datos en una base de datos y tramitarlos mediante la plataforma creada.

3.1.4.4. El cliente podrá consultar en todo momento el estado en el que se encuentra su reclamación, desde que registra la incidencia hasta que el problema de la maquina este resuelto o el dinero sea devuelto al consumidor.

3.1.5. Cerrar el trámite correspondiente a cada incidencia una vez solucionado el problema.

3.1.5.1. Se dejará en secretaria a disposición del del cliente un sobre con el nombre del consumidor y el dinero correspondiente a su incidencia cuando el problema declarado se trate de error en la devolución en el cambio de la máquina una vez que se haya resuelto.

3.1.5.2. Se enviará una notificación al usuario anunciando el correcto funcionamiento de la maquina cuando su problema se haya solucionado.

3.1.5.3. Se dará por acabado el problema una vez sea enviada la notificación con la resolución de su reclamación satisfactoriamente.

3.2. Monitorización del servicio

3.2.1 Se creará una página web desde la que se podrá realizar un seguimiento del servicio de las máquinas de café.

3.2.1.1. Desde la página web se podrán ver los datos de incidencias clasificados por fecha, tipo de incidencia o estado en la que se encuentre.

3.2.1.2. Habrá una sección donde aparecerán estadísticas relacionadas con las incidencias de las incidencias ocurridas, así como de las máquinas de café.

3.2.1.3. Podrá verse desde la web el tiempo de resolución de las incidencias.

3.2.1.4. La página web creará las etiquetas que irán en el sobre con el dinero devuelto del cliente.

Tras la consecución de los objetivos anteriores se habrán alcanzado competencias específicas del título de Ingeniería Electrónica Industrial y Automática relacionadas con el conocimiento aplicado de informática industrial y comunicaciones.

Capítulo 4

Contribuciones

Este trabajo trata de resolver el problema que ocurre cuando un usuario de una máquina, expendedora de café en este caso, tiene algún problema durante la ejecución del servicio.

La forma de solucionar este tipo de problemas era incómoda para el cliente. La solución que aporta este trabajo resuelve el problema mediante una app que puede llevarse en cualquier smartphone, esta es una forma más actualizada y ajustada a este tiempo ya que hoy en día todo el mundo lleva consigo un smartphone o una tablet. La aplicación permite registrar la incidencia, ver en que estado se encuentra y anularla una vez se ha resuelto.

La solución dada en este trabajo está enfocada a solucionar los problemas de las máquinas expendedoras de café, pero esta solución podría aplicarse a cualquier tipo de máquina.

Este trabajo resuelve el problema explicado anteriormente de una forma diferente a la que se suele seguir, ya que se utiliza como base de datos Google Sheets así como otras aplicaciones de Google que evolucionan día a día con nuevos avances que bien utilizados ayudan a dar solución a multitud de problemas.

Capítulo 5

Procedimiento

En el desarrollo de este TFG se ha utilizado una metodología ágil basada en Scrum, definida por el director. El trabajo se ha dividido en iteraciones de dos semanas denominadas sprints. Las unidades de trabajo se presentan en forma de historias de usuario (user stories) que definen mini-proyectos de muy corta duración que aportan valor al proyecto. Es decir, cada historia de usuario cumple o ayuda a cumplir alguno de los objetivos. Medir el valor percibido corresponde al propietario del producto (Product Owner), que participa activamente en la planificación del proceso priorizando las unidades de trabajo.

La utilización de una metodología ágil permite equilibrar la cantidad de trabajo y los objetivos alcanzados. Los 12 créditos ECTS del TFG se reparten según el criterio del director para que los resultados aporten el máximo valor posible, incluso en presencia de imprevistos.

5.1. Diferencias con Scrum

Scrum es una metodología estrictamente centrada en el cliente. El cliente es el responsable de priorizar y, en cierto modo, planificar las iteraciones. Esto garantiza que la ejecución del proyecto responde al máximo con las expectativas del cliente, aún cuando los imprevistos impidan alcanzar alguno de los objetivos iniciales. Esta característica de Scrum es la única que se ha intentado mantener inalterada. Sin embargo, el TFG es un proyecto individual, lo que ha requerido modificar significativamente otros aspectos de la metodología.

5.1.1. Roles

La única remuneración que se obtiene con la ejecución de un TFG es la calificación de los distintos aspectos (anteproyecto, valoración del director, valoración del tribunal, etc.). Por tanto, el cliente del TFG se compone por el director y el tribunal de la defensa. Desgraciadamente no es posible conocer a priori el tribunal. Por este motivo el director es el único representante del cliente en el proceso de desarrollo (Product Owner).

El TFG debe ser realizado de manera individual. Por tanto, el equipo de trabajo (Team Member) se compone exclusivamente por el autor. La labor de dirección del TFG se asimila a la de dirección del proyecto y, por tanto, el director también actúa como coordinador del proceso, o Scrum Master. Nótese que hay dos roles representados por la misma persona. Desde un punto de vista purista esto implica que puede haber conflicto de intereses y los intereses del cliente pueden estar insuficientemente representados. Es una limitación extrínseca, que no es posible solucionar con el proceso actual. Aun así, el uso de una metodología ágil centrada en el cliente debe mejorar el alineamiento de intereses cuando sobrevienen problemas que afectan o pueden afectar a la consecución de alguno de los objetivos iniciales.

5.1.2. Historias de usuario

Scrum, como la mayoría de los métodos ágiles, está enfocada al desarrollo de proyectos en entornos de alta incertidumbre por equipos multidisciplinares bien formados. El desarrollo de un TFG, al tratarse de un primer proyecto profesional, también está sometido a gran cantidad de incertidumbre. Sin embargo, no siempre se cuenta con la formación previa necesaria para

abordar todos los problemas. Esto implica que, en ocasiones, se necesita aprender o leer, sin repercusión medible en el valor percibido por el Product Owner. En esos casos se planifican unidades de trabajo que no corresponden estrictamente a historias de usuario en el sentido de Scrum. Se ha intentado mantener al mínimo este tipo de historias de usuario para tener el proceso lo más controlado posible.

Puntualmente ha sido necesario planificar historias de usuario que solo pretenden explorar opciones. Este tipo de historias de usuario están contempladas en Scrum, se denominan spikes. Sin embargo, en la ejecución de este TFG se ha procurado reducir al mínimo para que la exploración de alternativas no domine en el tiempo dedicado al TFG.

5.1.3. Planificación de sprints

Para la planificación y el seguimiento se ha utilizado un tablero Trello. Los tableros Trello permiten agrupar tarjetas en una serie de listas con nombre.

El autor ha sido responsable de añadir la mayoría de las historias de usuario a la lista Backlog. Se trata de un proceso continuo, durante toda la ejecución del proyecto. El director, como product owner, prioriza las historias, moviendo las tarjetas dentro de la lista Backlog. Justo antes de cada iteración se realiza una reunión presencial o virtual para revisar la iteración pasada y planificar la siguiente iteración.

Usando la técnica de planning poker se dimensionan las historias de usuario en días de trabajo. Esta técnica consiste en un proceso de generación de consenso entre el autor y el director sobre el tiempo requerido para la ejecución de cada historia de usuario. La unidad empleada ha sido de un día.

El director, como product owner traslada las tarjetas correspondientes a las primeras historias de la lista Backlog a la lista ToDo hasta completar los 10 días de trabajo de la iteración.

5.1.4. Flujo de trabajo

El flujo de trabajo diario del autor corresponde a la siguiente secuencia:

-Dentro de la lista ToDo puede elegirse cualquier tarjeta para trabajar en ella. Antes de comenzar el trabajo se arrastra la tarjeta a la lista Doing. Esto proporciona información en tiempo real al director del progreso de la iteración.

-Al terminar una historia de usuario la tarjeta correspondiente se arrastra a la lista QC (quality control).

-El director, como Scrum Master, revisa que la historia está realmente acabada y, si así es, la traslada a la lista Done. En caso contrario la traslada a la lista Doing otra vez, añadiendo un comentario que lo justifica.

-Si en el transcurso del trabajo se encuentra un obstáculo que impide progresar con una historia, se traslada a la lista Blocked, añadiendo un comentario que lo justifica.

En todo momento es posible ver el estado global de ejecución del proyecto. Al finalizar, la lista Done contiene todas las historias de usuario ejecutadas por orden de terminación. Y las listas Blocked y Backlog contienen (en este orden) todas las historias de usuario que corresponderían a trabajo futuro, ya priorizadas por el director.

5.1.5. Herramientas de ayuda

El proceso de desarrollo está fuertemente ligado a la herramienta Trello. Se trata de una herramienta colaborativa en línea, que permite mantener una serie de tarjetas agrupadas en listas con nombre. Cada tarjeta puede tener un título, una descripción, un conjunto de adjuntos, y un conjunto de comentarios. Trello se ha usado con éxito en la planificación de proyectos de nivel de complejidad muy variable. Por ejemplo, Epic Games utiliza un tablero Trello para planificar las características a incorporar a cada nueva versión de Unreal Engine. Por otro lado, un problema de Trello es el manejo limitado de la historia de modificaciones en las tarjetas y en los movimientos entre listas de tarjetas. Esto dificulta en cierto modo el seguimiento de los cambios y, sobre todo, la corrección de errores en el proceso. Por este motivo, Trello solo se ha empleado en la coordinación del trabajo, mientras que toda la gestión de cambios se ha delegado en otra herramienta.

Todo el proyecto ha sido gestionado desde su inicio con una herramienta de control de versiones distribuido en un repositorio público de GitHub, disponible en https://github.com/Julian92UCLM/cosmos. Cada vez que se completa con éxito una historia de usuario se notifica mediante un comentario en la tarjeta correspondiente. Este comentario tan solo contiene el identificador del paquete de cambios (commit) que da por concluida la historia. Todos los stakeholders pueden consultar la evolución del proyecto en todo momento desde la propia página del repositorio.

Capítulo 6

Resultados

6.1. App Inventor como mecanismo para crear aplicaciones para móvil

Los desafíos a los que se enfrenta la enseñanza hacen que se busquen continuamente nuevos métodos para hacer frente a estos problemas.
Con la intención de facilitar la programación de aplicaciones móviles surge App Inventor mediante un lenguaje visual basado en bloques.
App Inventor para Android, es un entorno visual de programación en bloques, que posibilita el desarrollo de aplicaciones para dispositivos móviles Android de forma bastante sencilla sobre todo si es comparado con los lenguajes de programación tradicionales.

6.1.1. El entorno App Inventor

App Inventor es un lenguaje visual de programación para la creación de aplicaciones para dispositivos móviles, desarrollada conjuntamente por Google y MIT (Instituto Tecnológico de Massachusetts). App Inventor está basado en un entorno de desarrollo compuesto de bloques, sin embargo, la diferencia fundamental radica en permitir a los usuarios construir aplicaciones que disponen de servicios basados en la web, lectura de códigos de barras, interacción con redes sociales, interacción con sensores de orientación y geolocalización.
La creación de aplicaciones con App Inventor es bastante intuitiva y no requiere conocimientos previos avanzados en programación. Además, las aplicaciones creadas se pueden utilizar en cualquier dispositivo con plataforma Android. App Inventor está basado en componentes los cuales el usuario los manipula mediante la interacción con la aplicación.
La construcción de una aplicación en esta plataforma se realiza a través de dos espacios: App Inventor Designer y Blocks Editor. App Inventor Designer puede ejecutarse desde cualquier navegador, gracias a él podemos crear visualmente el interfaz de la aplicación simplemente haciendo clic y arrestando componentes como botones, cuadros de texto, figuras, sonidos o animaciones que se encuentran en Palette.
En la ventana de Bock Editor, se encuentran bloques que podemos conectar, estos bloques también se pueden añadir a la pantalla haciendo clic y a arrastrando hacia el espacio de trabajo, con ellos podemos controlar de qué manera se comportaran los componentes que previamente se han definido en la ventana de App Inventor Designer.
La aplicación nos permite realizar pruebas mediante un emulador o instalar la aplicación y ejecutarla directamente desde un dispositivo Android (smartphone o tablet).
Ventajas de utilizar App inventor respecto a otros entornos.
Utilizar App inventor trae consigo ventajas respecto a otras plataformas que podrían dar resultados similares al que se busca en este trabajo.
La ventaja principal de crear una App nativa (es una aplicación desarrollada para smartphones en el lenguaje nativo del propio terminal) respecto a una Web App es que la App nativa una vez descargada e instalada en el smartphone estará siempre a disposición del usuario sin necesidad de acceder a internet cada vez que quiera acceder a ella.

Otras ventajas de que la aplicación esté instalada en el dispositivo móvil son la posibilidad de acceder a elementos como la cámara o la ubicación además de ser más rápida que una web App.
Las ventajas de utilizar App inventor respecto a otras plataformas de programación de App nativas está en que el lenguaje de programación de App inventor consiste en un entorno visual de programación con bloques. Esto hace más sencillo la creación de la aplicación ya que no es necesario saber otros lenguajes de programación como Java, JavaScript, HTML5…
Estas ventajas hacen que utilizar App inventor para realizar la aplicación sea la mejor opción.

6.2. Modelo de base de datos

La base de datos está compuesta por tablas relacionadas entre sí:

-La tabla de registro de usuario se utiliza para almacenar los datos necesarios de los clientes para su registro.

-La tabla de incidencias guardará los datos de las incidencias registradas desde la aplicación.

-En la tabla tipos de incidencias aparecerán las incidencias más comunes, cada una tendrá un ID asignado.

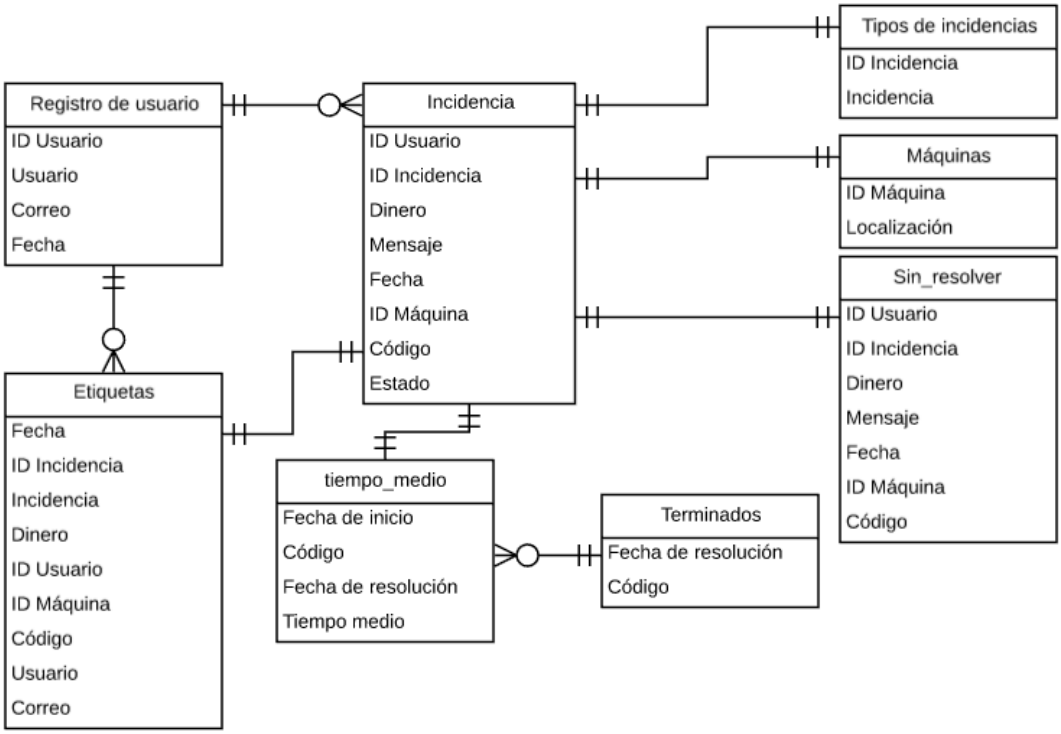
-La tabla máquinas contendrá todas las maquinas relacionadas con esta aplicación y habrá un ID único para cada una.

-La tabla *Sin_resolver* almacenará las incidencias que no estén cerradas, es decir, las que no tengan el valor *cerrado* en la columna de estado.

-La tabla *terminado* recibirá de la aplicación el código de finalización de la incidencia junto con la fecha en la que se ha cerrado.

-La tabla *tiempo_medio* enlazara los datos de código y fecha de finalización de la tabla *terminado* con la fecha de inicio de la tabla de registro de incidencias.

-la tabla *etiquetas*, almacenará los datos de incidencias de la tabla de incidencias cullo valor en la columna de estado no sea cerrado y que la incidencia se deba a una devolución errónea en el cambio, estos datos los enlazará mediante el ID de usuario con los de la tabla de registro de usuario obteniendo el correo y el usuario.

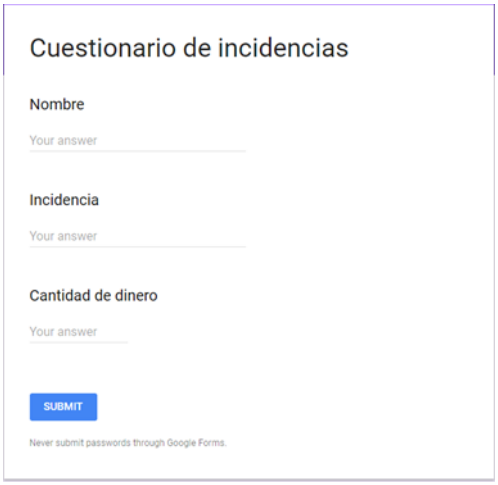


Modelo de base de datos

6.3. Aplicación dos formularios: identificación e incidencias.

Para recibir las incidencias a la base de datos es necesario disponer de alguna plataforma en la que poder publicar las preguntas que está buscando el cliente afectado para que este pueda responder con su problema. Para esto Google Sheets ofrece la posibilidad de crear formularios y almacenar las respuestas a cada pregunta en columnas.

Podríamos crear un cuestionario con los apartados para nombre, incidencia, cantidad de dinero por si ese fuera el problema y quedaría de la siguiente forma.

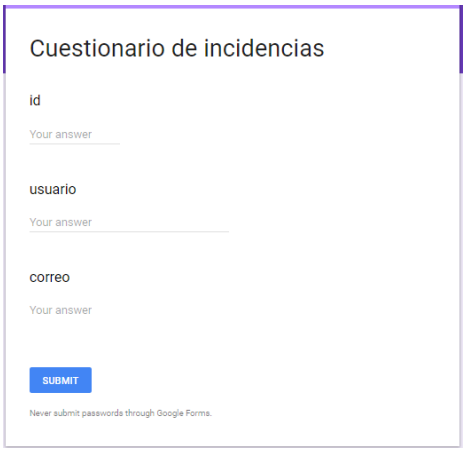


Formulario de Google para registro de usuarios e incidencias

Pero este resultado es bastante primitivo aparte de que lo más correcto sería que en la base de datos estuvieran en tablas separadas los datos del usuario y las incidencias.

Por eso crearemos dos formularios, uno para almacenar los datos del usuario, y otro para registrar las incidencias. El objetivo es subir estos dos cuestionarios de alguna manera a la aplicación para almacenar los datos.

El cuestionario elegido tendrá secciones para almacenar: id, usuario y correo. Quedando de la siguiente manera.



Formulario de Google para registro de usuarios

El id será un numero único para cada usuario y será asignado por la base de datos, el usuario será el nombre con el que el cliente afectado quiera que le nombremos, el correo electrónico será necesario para informarle del estado en el que se encuentra su incidencia y la contraseña será una medida de seguridad, para que no pueda entrar un usuario en la cuenta de otro.

A	B	C	D	E
Marca temporal	id	usuario	correo	Contraseña

Tabla de Google Sheets para registro de usuarios

Como vemos se ha creado una columna para cada sesión del cuestionario, ademas, podemos observar que aparte de crearse estas tres columnas también se ha creado una cuarta con la fecha en el que se envía el formulario. Ya tenemos creada la tabla con los campos necesarios para el registro en la App.

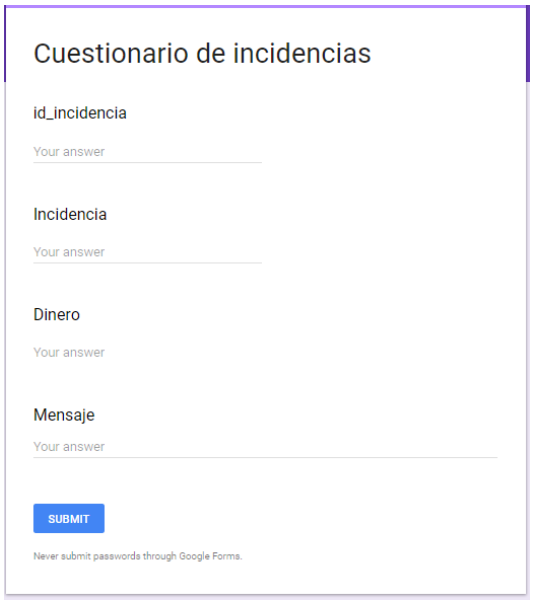
El cuestionario para almacenar las incidencias contendrá los siguientes campos:

-*Id_incidencia*, al igual que para el usuario cada incidencia tendrá un id único e irrepetible con el cual se hará referencia al conjunto de la incidencia cuando sea necesario.

-*Incidencia*, en este espacio el usuario podrá detallar el problema que ha tenido con la máquina.

-*Dinero*, cuando el contratiempo se deba a una devolución incorrecta de la máquina al cliente, el usuario introducirá en este lugar la cantidad no devuelta por la máquina.

-Además, aparecerá otra cuestión llamada *mensaje* en la que la persona que rellene el formulario podrá añadir más detalles de la incidencia si fuera necesario.



Formulario de Google para registro de incidencias

A	B	C	D	E
Timestamp	id_incidencia	Incidencia	Dinero	Mensaje

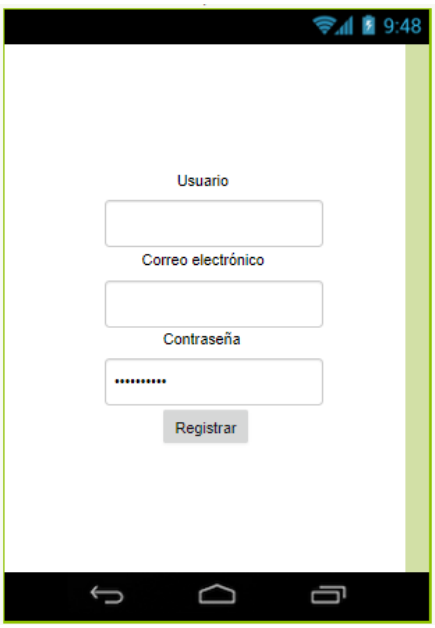
Tabla de Google Sheets para registro de incidencias

Al igual que antes podemos observar que aparte de crearse los encabezados de las columnas que hemos incluido en el formulario también se ha creado una columna con la fecha en la que se envía la incidencia.

Este dato resulta de gran interés ya que si conocemos la fecha en la que se registra la incidencia podremos clasificarla por tiempo o calcular la duración en resolver el problema.

Como dijimos en un punto anterior, utilizaremos App Inventor para crear la aplicación, para ello accedemos a la página web y creamos un nuevo proyecto.

Para crear el formulario de registro de usuario desde el apartado de diseño, arrastraremos a la pantalla tres cajas de texto, tres etiquetas de texto para identificar cada caja, un botón para enviar los datos del formulario y también arrastraremos el elemento Web que se encuentra en el apartado de conectividad, este es un componente no visible que permite gestionar tanto solicitudes HTTP como GET, POST, PUT y DELETE. Una vez hecho quedara de la siguiente forma.



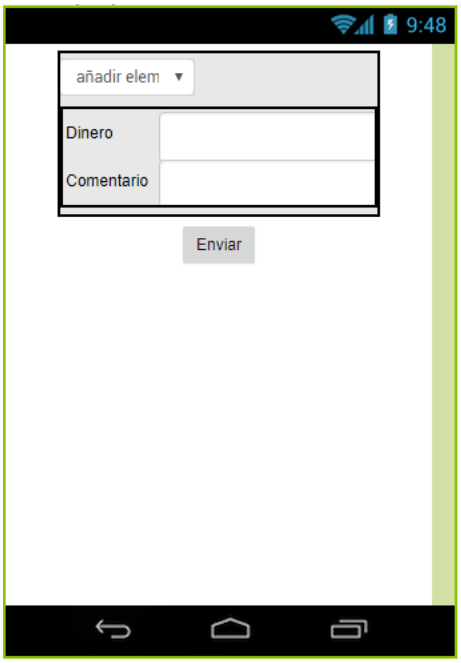
Pantalla de registro de usuario

Ahora crearemos otra pantalla en la que construiremos otro formulario, este será par a introducir los datos referentes a la incidencia.

Añadiremos a la pantalla un desplegable en el que introduciremos las incidencias más comunes (no me ha devuelto el cambio, no hay agua, no hay café, no hay vaso, no hay azúcar, el líquido está frío, maquina no operativa y cantidad insuficiente) y la opción otro por si no estuviera entre estas.

Agregaremos también a la pantalla dos cajas de texto, una para introducir la cantidad económica que no ha devuelto la maquina si este fuera el problema y la otra para introducir la incidencia si la opción fuera otra a las que aparecen en la lista. Estas cajas de texto dependerán de la selección de la incidencia, inicialmente no serán visibles y se verán o no en función de la selección del usuario.

También al igual que en el formulario de registro añadiremos la opción web de conectividad y un botón para enviar el formulario.



Pantalla de registro de incidencias

Desde la aplicación enviaremos los datos a la base de datos mediante un POST, es decir enviaremos los datos desde un enlace.

Para ello lo primero que haremos será ir al formulario en vivo y mirar el código fuente del formulario. Encontraremos la URL de publicación en la etiqueta acción del formulario y los elementos de la lista.

```
<html class>
▶<head>…</head>
▼<body dir="ltr" itemscope itemtype="http://schema.org/CreativeWork/FormObject" class="freebirdLightBackground" data-is-prepopulate-mode="false" jscontroller=
"OkF2xb" jsaction="rcuQ6b:npT2md; click:KjsqPd;n6dfDc:GrC6Ef;oDrQPd:19NwTb;UizPIf:zbvklb;wINJic:.CLIENT;JPpwNb:.CLIENT">
   <meta itemprop="name" content="Registro">
   <meta itemprop="faviconUrl" content="https://ssl.gstatic.com/docs/spreadsheets/forms/favicon_qp2.png">
   <meta itemprop="url" content="https://docs.google.com/forms/d/e/1FAIpQLSclLujpPS-oegUuh4EVf9gB4JH6i4eQtq2Yc892pr1Vumzk-Q/viewform?usp=embed_googleplus">
   <meta itemprop="embedURL" content="https://docs.google.com/forms/d/e/1FAIpQLSclLujpPS-oegUuh4EVf9gB4JH6i4eQtq2Yc892pr1Vumzk-Q/viewform?
   embedded=true&usp=embed_googleplus">
   <meta itemprop="thumbnailUrl" content="https://ssl.gstatic.com/docs/forms/social/social-forms-big-2.png">
   <meta itemprop="image" content="https://ssl.gstatic.com/docs/forms/social/social-forms-big-2.png">
   <meta itemprop="imageUrl" content="https://ssl.gstatic.com/docs/forms/social/social-forms-big-2.png">
  ▼<div class="freebirdFormviewerViewFormContentWrapper">
   ▶<div class="freebirdFormviewerViewFormBanner freebirdHeaderMast">…</div>
   ▼<div class="freebirdFormviewerViewCenteredContent">
    ▼<form action="https://docs.google.com/forms/d/e/1FAIpQLSclLujpPS-oegUuh4EVf9gB4JH6i4eQtq2Yc892pr1Vumzk-Q/formResponse" target="_self" method="POST" id=
    "mG61Hd"> == $0
     ▼<div class="freebirdFormviewerViewFormCard exportFormCard">
       <div class="freebirdFormviewerViewAccentBanner freebirdAccentBackground"></div>
      ▼<div class="freebirdFormviewerViewFormContent ">
       ▶<div class="freebirdFormviewerViewNoPadding">…</div>
       ▼<div class="freebirdFormviewerViewItemList" role="list">
        ▼<div class="freebirdFormviewerViewNumberedItemContainer">
         ▼<div role="listitem" class="freebirdFormviewerViewItemsItemItem freebirdFormviewerViewItemsTextTextItem" jsname="ibnC6b" jscontroller="rDGJeb"
         jsaction="sPvj8e:e4JwSe,Dlbleb;O22p3e:e4JwSe,F0zejc;" data-item-id="1350201718">
          ▶<div class="freebirdFormviewerViewItemsItemItemHeader">…</div>
          ▶<div class="freebirdFormviewerViewItemsTextItemWrapper">…</div>
           <div class="freebirdFormviewerViewItemsItemGradingGradingBox freebirdFormviewerViewItemsItemGradingFeedbackBox" jsname="R7fTud"></div>
           <div jsname="XbIQze" class="freebirdFormviewerViewItemsItemErrorMessage" id="i.err.1350201718" role="alert"></div>
          </div>
```

Código fuente del formulario de Google. URL

Para obtener los elementos de la lista, buscaremos la etiqueta de nombre hasta localizar un valor como “entry.16715522”.

```
▼<form action="https://docs.google.com/forms/d/e/1FAIpQLSclLujpPS-oegUuh4EVf9gB4JH6i4eQtq2Yc892pr1Vumzk-Q/formResponse" target="_self" method="POST" id=
"mG61Hd">
  ▼<div class="freebirdFormviewerViewFormCard exportFormCard">
     <div class="freebirdFormviewerViewAccentBanner freebirdAccentBackground"></div>
   ▼<div class="freebirdFormviewerViewFormContent ">
    ▶<div class="freebirdFormviewerViewNoPadding">…</div>
    ▼<div class="freebirdFormviewerViewItemList" role="list">
     ▼<div class="freebirdFormviewerViewNumberedItemContainer">
      ▼<div role="listitem" class="freebirdFormviewerViewItemsItemItem freebirdFormviewerViewItemsTextTextItem" jsname="ibnC6b" jscontroller="rDGJeb"
      jsaction="sPvj8e:e4JwSe,Dlbleb;O22p3e:e4JwSe,F0zejc;" data-item-id="1350201718">
       ▼<div class="freebirdFormviewerViewItemsItemItemHeader">
        ▶<div class="freebirdFormviewerViewItemsItemItemTitleContainer">…</div>
        </div>
       ▼<div class="freebirdFormviewerViewItemsTextItemWrapper">
        ▼<div class="quantumWizTextinputPaperinputEl freebirdFormviewerViewItemsTextShortText freebirdThemedInput modeLight" jscontroller="pxq3x"
        jsaction="clickonly:KjsqPd; focus:Jt1EX; blur:fpfTEe; input:Lg5SV;" jsshadow jsname="W85ice">
         ▼<div class="quantumWizTextinputPaperinputMainContent exportContent">
          ▼<div class="quantumWizTextinputPaperinputContentArea exportContentArea">
           ▼<div class="quantumWizTextinputPaperinputInputArea">
             <input type="text" class="quantumWizTextinputPaperinputInput exportInput" jsname="YPqjbf" autocomplete="off" tabindex="0" aria-label=
             "correo" aria-describedby="i.desc.1350201718 i.err.1350201718" name="entry.16715522" value dir="auto" data-initial-dir="auto" data-
             initial-value> == $0
             <div jsname="LwH6nd" class="quantumWizTextinputPaperinputPlaceholder exportLabel">Your answer</div>
           </div>
```

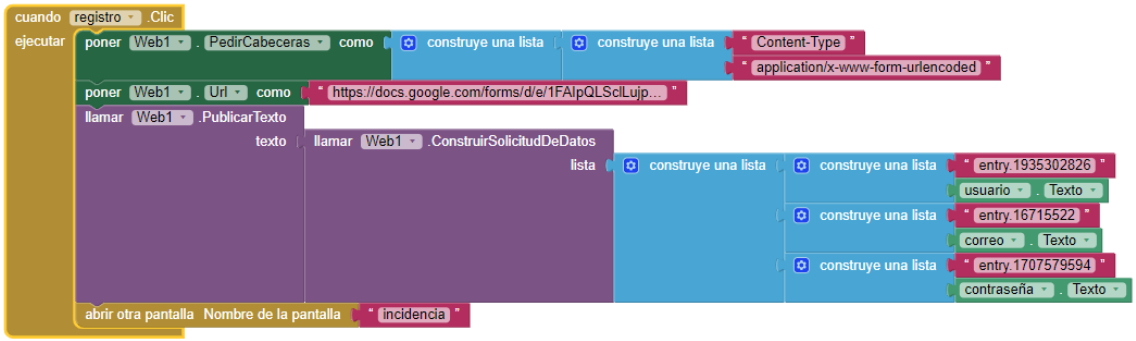
Código fuente del formulario de Google. Elementos

Esto lo tendremos que hacer tanto en el formulario de registro como en el de envío de incidencias.

Después iremos a la hoja de cálculo y seleccionaremos Archivo – Compartir, hacemos clic en *avanzado* y cambiamos el acceso de privado a cualquiera con enlace y cambiamos el tipo de acceso de *puede ver* que viene por defecto a *se puede editar* y hacemos clic en guardar.

Ahora seleccionamos archivo – publicar en la web, en la siguiente pantalla hacemos clic en publicar y luego en aceptar y cerrar.

Hecho esto ahora construiremos los bloques. Realizaremos el envío de los datos de registro del usuario mediante un POST utilizando el enlace obtenido del código fuente del formulario y los nombres de los elementos los pondremos la siguiente forma.

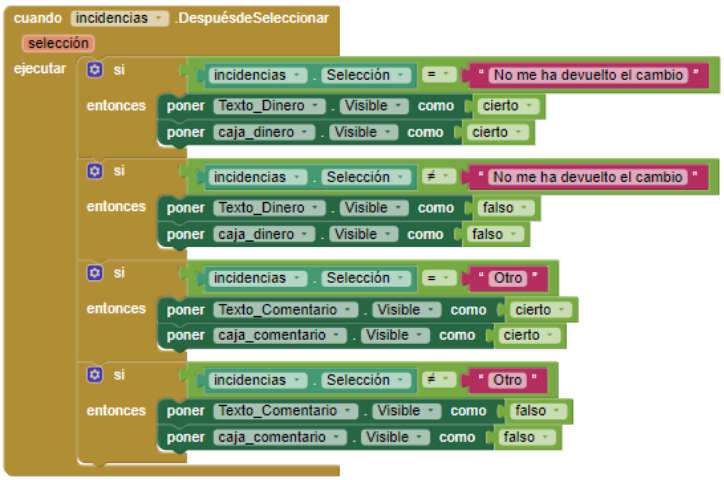


Bloques del botón registro de pantalla de registro de usuarios

Los datos referentes a las incidencias los enviaremos de la misma forma que los de registro de usuario, obtendremos el enlace del formulario y el nombre de los elementos que queremos conseguir.

Bloques del botón enviar, pantalla de registro de usuarios

En este formulario sin embargo veremos las cajas de texto de dinero y de comentarios en función de la selección como podemos ver en la imagen.



Bloques de incidencias, pantalla de incidencias

6.4. Relación usuario id

Anteriormente tanto en la tabla de registro de usuarios como en la de incidencias hemos creado un campo identificado como *id*. Este campo será muy útil para poder identificar cada incidencia o cada usuario por un número que será único en cada caso.

Este campo queremos que se cree automáticamente, para ello crearemos un script con un programa que nos realizará este trabajo.

Para ello desde la hoja de cálculo iremos a herramientas, script editor y crearemos un nuevo script.

```
function onFormSubmit(e) {
  Logger.log('Form submitted');
  return;
  var uniqueID = getUniqueID(e.values);
  Logger.log('Unique Id: %d', uniqueId);
  recordResponseID(e.range, uniqueID);
```

```
}
function recordResponseID(eventRange, uniqueID) {
  var row = eventRange.getRow();
  var column = eventRange.getLastColumn() + 1;
  var sheet = SpreadsheetApp.getActiveSheet();
  sheet.getRange(row, column).setValue(uniqueID);

}
function getConnectedForm() {
  var formUrl = SpreadsheetApp.getActiveSpreadsheet().getFormUrl();
  var form =  FormApp.openByUrl(formUrl);
  return form;
}
function getUniqueID(eventValues) {
  return getConnectedForm().getResponses().length;
}
```

Script crear id

Este programa lee el último dato de la columna de *id* y le suma +1. Este proceso se hará automáticamente cada vez que se envíe un formulario de usuario o de incidencias.

De este modo ya tenemos automatizada la creación del id para cada incidencia o usuario.

6.5. Usuario recordado

Para que el uso de la aplicación sea lo más cómodo posible para el usuario lo ideal es que una vez registrado, la aplicación sea capaz de recordar sus datos.

Las aplicaciones creadas con App Inventor se inicializan cada vez que son ejecutadas. Esto quiere decir que si una aplicación fija el valor de una determinada variable y el usuario cierra la aplicación el valor de esta variable no será recordado la próxima vez que se inicialice la aplicación.

Para ello App Inventor dispone de TinyDB, este es un componente invisible que sirve para almacenar datos para la aplicación.

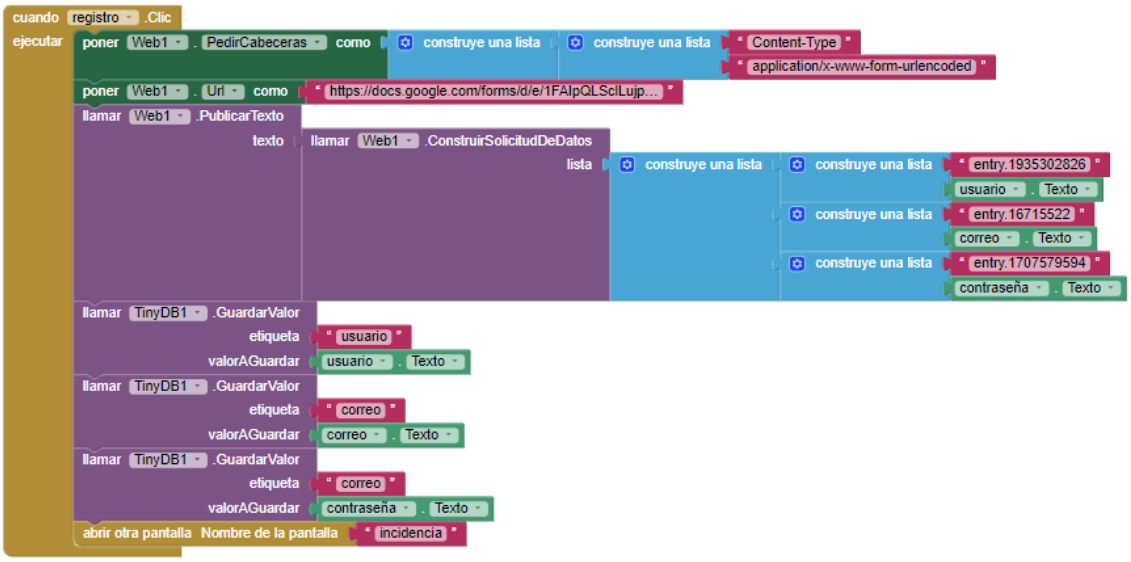
Los datos almacenados en TinyDB estarán disponibles cada vez que se abra la aplicación.

Los datos se guardan mediante valores con etiquetas. Para guardar un elemento hay que especificar la etiqueta con la que quiere guardarse. Luego para recuperar el valor solo hay que poner la etiqueta con la que hemos guardado ese dato.

Para añadir este componente simplemente lo arrastramos desde los componentes de almacenamiento hasta nuestra pantalla, una vez añadido el componente nos vamos a los bloques donde lo configuraremos.

Añadiremos dos bloques de TinyDB en los cuales el valor a guardar será el texto escrito en los campos de texto de usuario, contraseña y correo con las etiquetas usuario, contraseña y correo que serán los nombres con los que se identificarán estas variables.

Cuando el usuario haga clic en el botón de registrar automáticamente los valores de usuario, correo y contraseña quedaran guardados en el dispositivo.



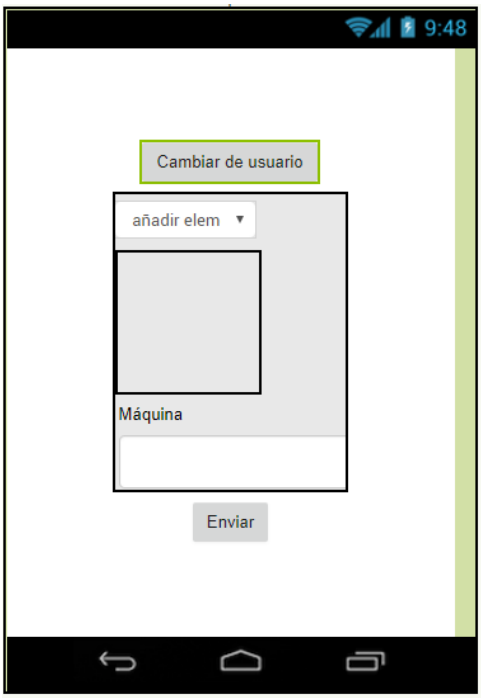
Bloques del botón registro con TinyDB, pantalla de registro de usuarios.

Ahora que ya podemos guardar los datos en el dispositivo crearemos un bloque para recordar los datos por defecto al abrir la aplicación, de este modo no será necesario registrarse una y otra vez, sino que la aplicación se abrirá con los datos que nos hemos registrado.

Bloques de inicialización, pantalla de registro de usuarios

Estos datos nos aparecerán siempre hasta una vez que los cambiemos. Para ello pondremos un acceso a la página de registro desde la página en la que registramos las incidencias.

Para ello arrastramos un botón desde la Paleta de componentes hasta nuestra pantalla y añadimos un bloque para realizar la función de pasar a la pantalla de registro de usuario.



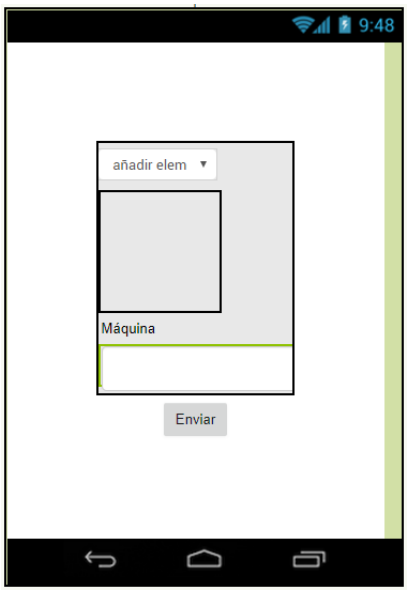
Pantalla de registro de usuarios con opción de cambiar de usuario

Bloques del botón de cambiar de usuario, pantalla de incidencias

6.6. Añadir opción de máquinas en la aplicación

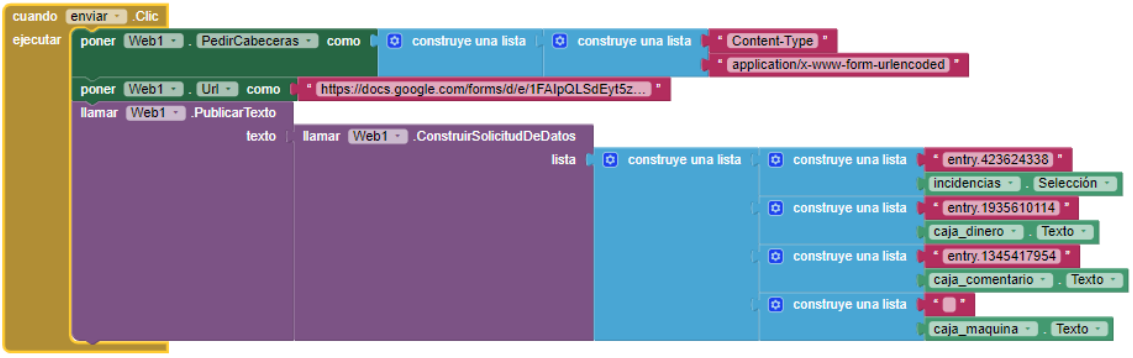
Para especificar mejor el problema ocurrido al usuario de las máquinas y añadir detalles que precisen mejor la incidencia para poder facilitar su resolución, añadiremos la opción para elegir la máquina que en concreto a tenido el problema.

Para ello volveremos a la página de App Inventor para ampliar el formulario de las incidencias. En el apartado de diseño añadiremos una etiqueta y una caja de texto para añadir el numero de la máquina. Renombramos los componentes añadidos a la pantalla y en las propiedades de la caja de texto marcamos la opción solo números, ya que el identificativo de cada máquina será un número.



Pantalla de incidencias, añadido introducir id de máquina

Una vez añadidos los componentes pasamos a añadirles funcionalidades en la zona de bloques. Añadimos a la lista de los datos que queremos enviar el dato que se introduzca en la caja de texto de *caja_maquinas.*

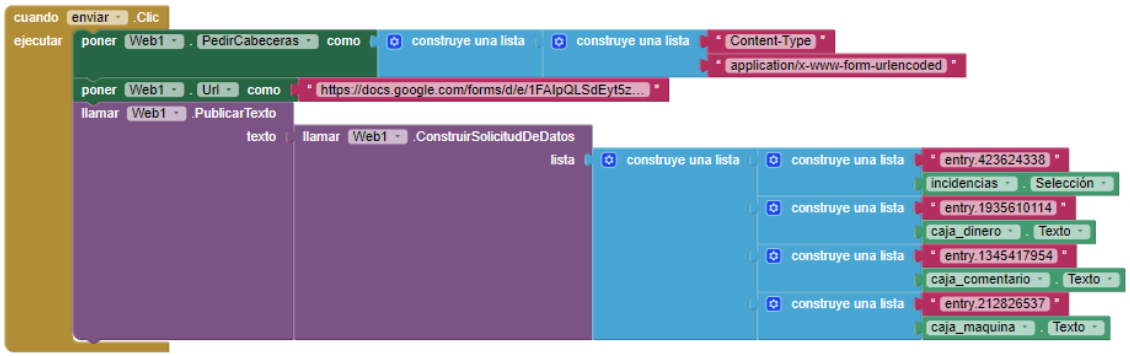


Bloque de botón enviar con máquina, pantalla de incidencias

El hueco que queda en blanco será la etiqueta de acción del formulario de la pregunta correspondiente a la opción máquina que crearemos.

Una vez editado el formulario de Google vemos el código fuente y copiamos la acción del formulario del apartado máquina.

Una vez pegado quedara de la siguiente forma.



Bloque botón enviar con máquina, pantalla de incidencias II

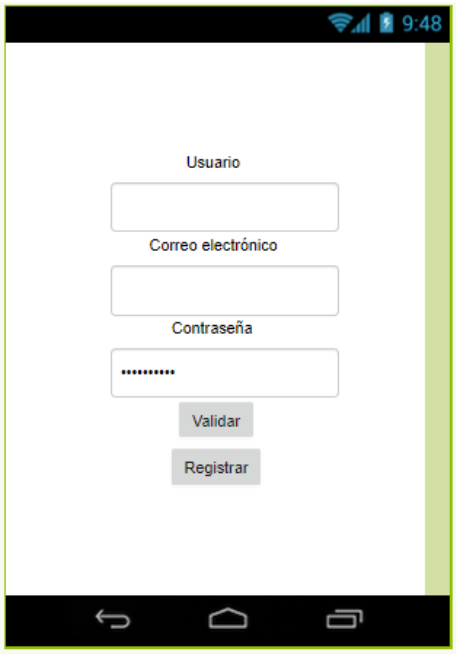
Ahora ya tenemos la tabla con la columna Máquina en la que se irán almacenando los id de las máquinas relacionadas con algún fallo que se introduzcan en la aplicación.

	A	B	C	D	E	F
1	Timestamp	id_indciencia	Incidencia	Dinero	Mensaje	Máquina
2						
3						

Tabla de incidencias con columna para máquinas

6.7. No repetir usuario

Para evitar que haya usuarios repetidos, pero con diferente id habrá que evitar que un mismo usuario se escriba dos veces en la base de datos. Para ello añadimos a la pantalla un botón al que llamaremos validar y deshabilitaremos el botón registrar modificando sus propiedades. La intención es que no nos deje registrar el usuario, contraseña y el correo sin antes comprobar si está ya en la base de datos o no.



Pantalla de registro de usuario con botón validar

Después de hacer esta modificación en el diseño de la pantalla modificaremos los bloques para que realicen la función deseada.

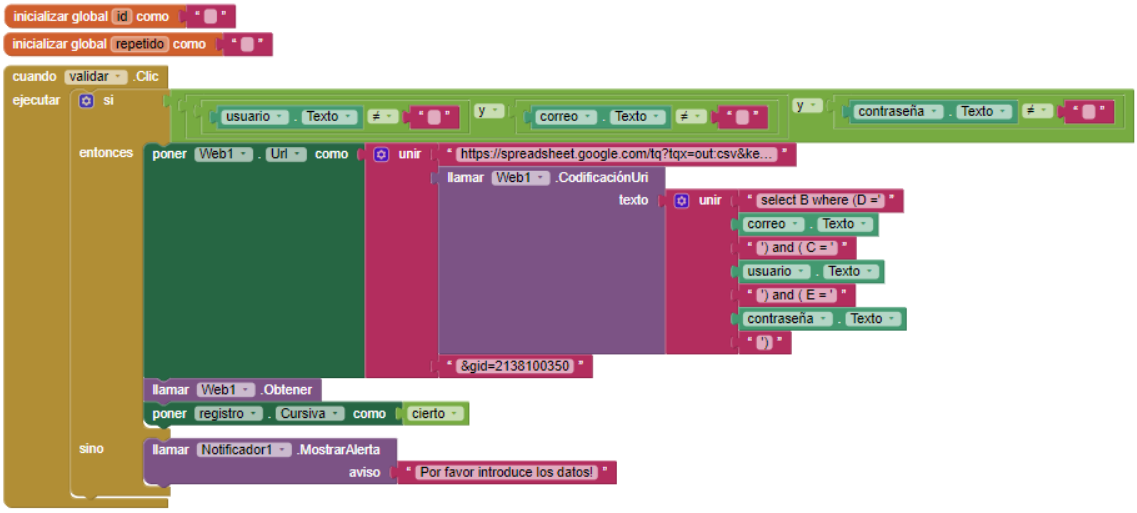
Crearemos dos variables a las que inicializaremos sin ningún valor a las que llamaremos id y repetido.

> -La variable *id* obtendrá el valor de id de usuario en el caso de que el usuario y correo que introduzcamos ya se encuentre en la base de datos.
>
> -La variable *repetido* podrá tener dos valores, verdadero si el correo y usuario ya se encuentra registrado o falso si por el contrario el usuario y correo que escribimos en las cajas de texto son nuevos y aun no se han registrado.

Cuando hacemos clic en validar con algún campo de texto vacío no nos dejará continuar y nos saltará una alerta que indicará que introduzcamos todos los datos.

Una vez que hemos introducido los datos y hecho clic en el botón validar el siguiente bloque realizará una consulta en nuestra base de datos mediante una URL. Esta consulta se realizará uniendo un enlace da acceso a la tabla en la que queremos hacer la consulta y el código que nos resulte del transformador de código URI de la consulta que queremos realizar.

Para terminar con este bloque habilitaremos el botón registrar que hasta este momento se encontraba deshabilitado.

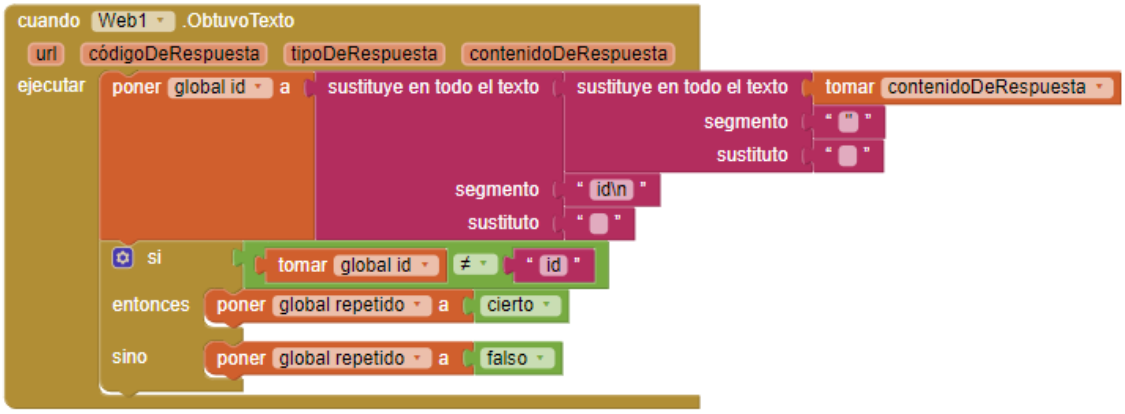


Bloque de botón validar, pantalla de registro de usuarios

Una vez mandada la consulta nos dará una respuesta que será el id de los datos si estos ya se encuentran en la base de datos o nada si estos datos aún no se han registrado.

El problema es que la respuesta vendrá en forma de columna y con los datos entre comillas, por ello sustituiremos las comillas y la palabra id por un texto vacío mediante una combinación de bloques para quedarnos con el valor de id si este ya existe o la palabra id si este aun no existiera.

Para finalizar con este bloque pondremos la variable repetido en *falso* si el resultado de la combinación de bloques anteriores nos hubiera quedado *id* o en *cierto* si fuera diferente, en este caso el resultado habrá sido el id de los datos ya existentes en la base de datos.

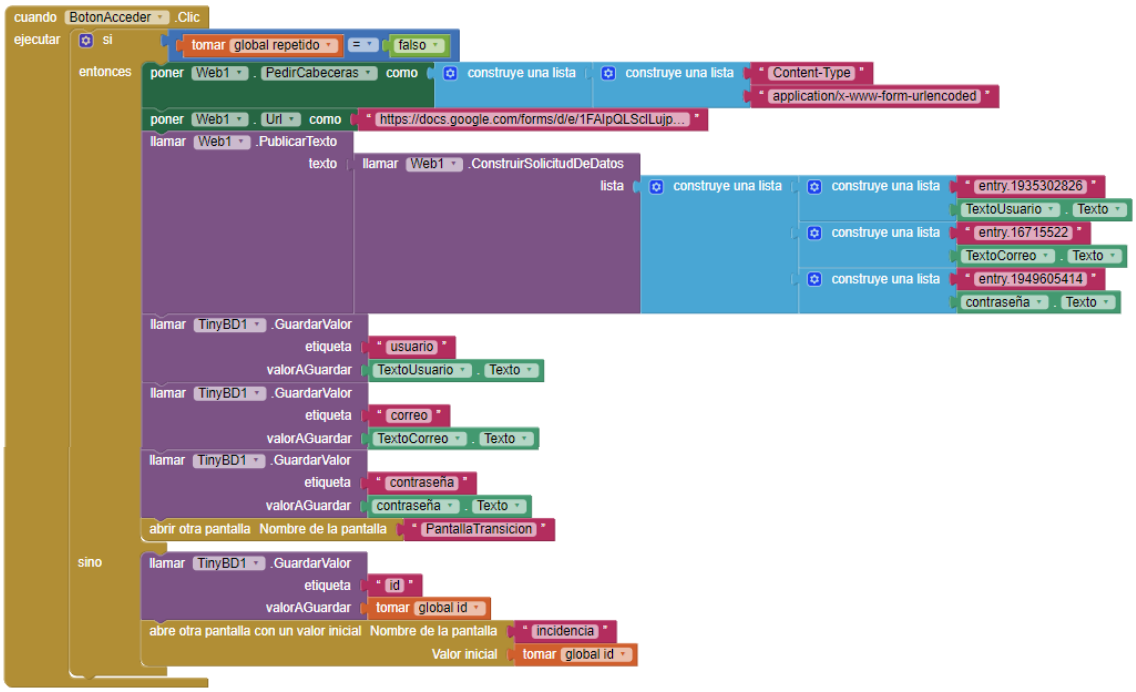


Bloque respuesta Web, pantalla de registro de usuarios

Modificaremos el bloque de las funciones del botón registrar realizado en un punto anterior, ahora al hacer clic en el botón registrar podremos encontrarnos con dos casos:

Si la variable *repetido* tiene el valor *falso* enviaremos los datos a la base de datos tal y como está hecho. La diferencia es que tras enviar los datos iremos a una página que crearemos a continuación que llamaremos *PantallaTransicion* con la intención de hallar el id de los datos que acabamos de mandar.

Si la variable *repetido* tiene el valor *cierto* almacenaremos el valor de id en TinyDB e iremos directamente al formulario de incidencias.



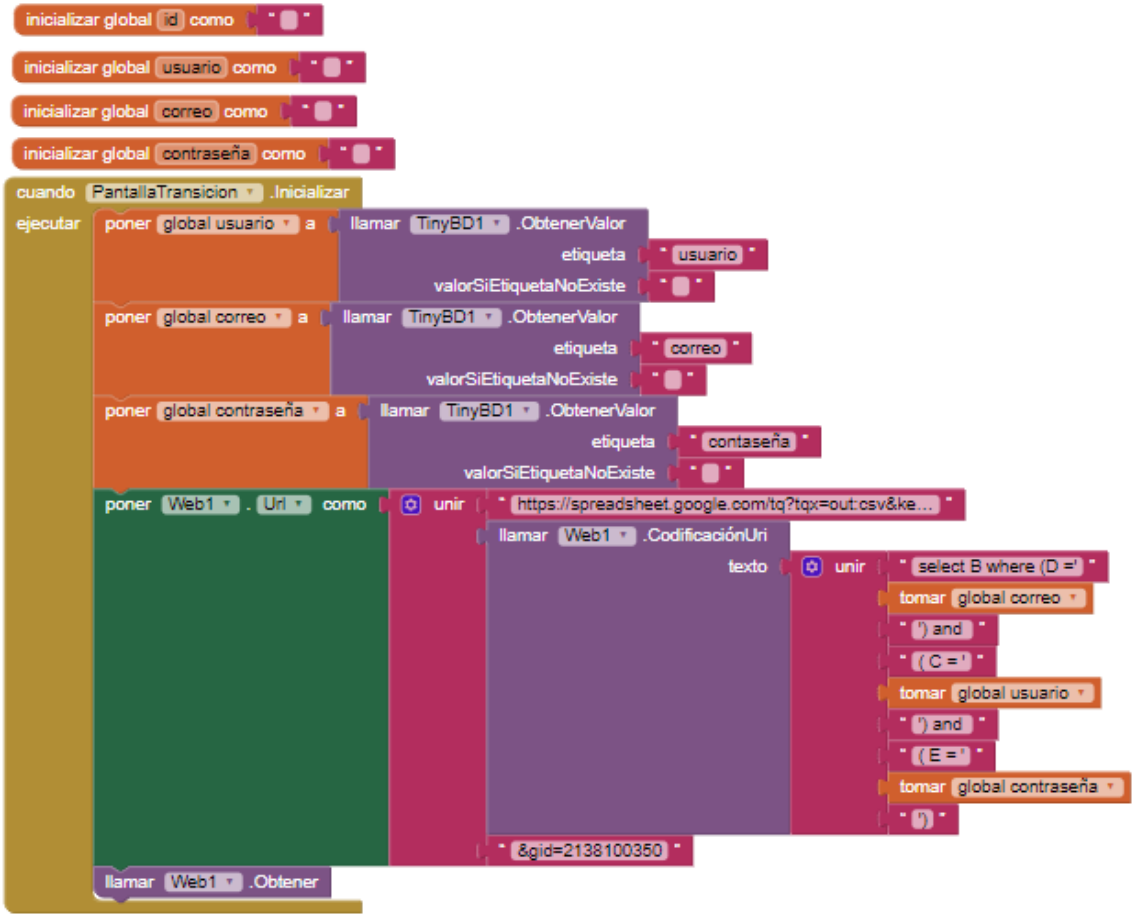
Bloque enviar con máquina, pantalla de incidencias

Como hemos dicho anteriormente crearemos una pantalla en la que hallaremos el id del usuario en los casos en los que el usuario no esté registrado y tengamos que haber mandado los datos.

Tras crear la pantalla *PantallaTransicion*, desde la ventana de diseño, arrastraremos de la paleta de componentes a nuestra pantalla los elementos TinyDB y Web.

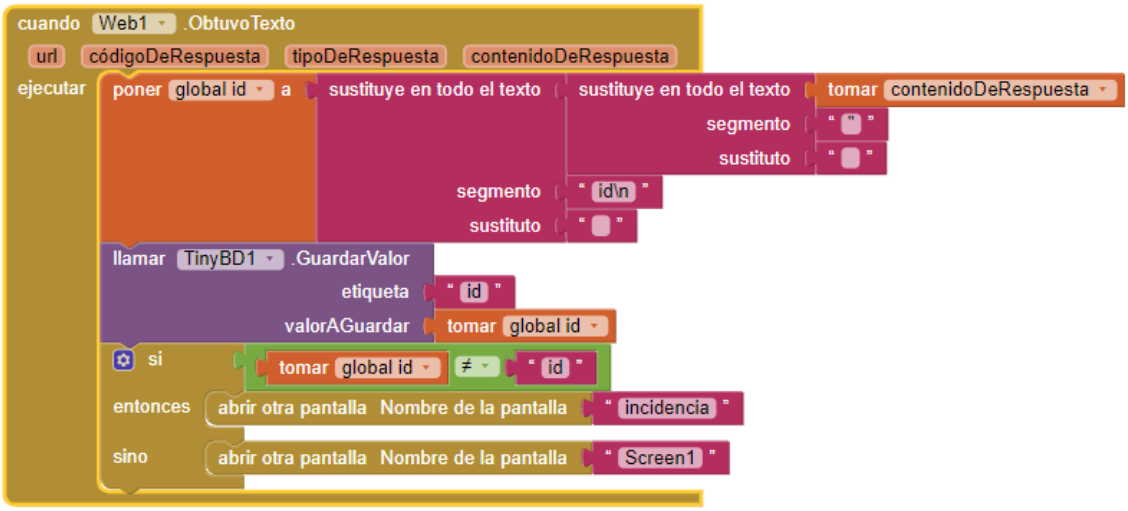
Luego desde la ventana de bloques crearemos las variables id, usuario, contraseña y correo.

Después, crearemos un bloque en el que al inicializarse la pantalla pondremos las variables usuario, contraseña y correo los valores almacenados en TinyDB y realizaremos una consulta mediante una URL de la misma manera que hemos hecho anteriormente.



Bloque iniciar PantallaTransicion, de la PantallaTransicion

Tras hacer la consulta obtenemos el valor de id y pasamos a la pantalla de incidencias.



Bloque respuesta Web, PantallaTransicion

6.8. Ver mis últimas incidencias desde la aplicación

En este punto añadiremos a la aplicación una funcionalidad que será de gran utilidad, incorporaremos una opción para poder ver todas las incidencias que se han introducido a la base de datos con el usuario que este activo en ese momento en la aplicación.

Para crear esta opción añadiremos otra página a la aplicación, la llamaremos *mis_incidencias*. Arrastramos de la paleta de componentes dos botones, TinyDB, el componente web y un VisorWeb.

Renombraremos los botones, uno será el encargado de realizar la función de volver a la página anterior y lo llamaremos *volver* y el otro se encargará de mostrar las ultimas incidencias y lo nombraremos como *histórico*.

VisorWeb es un componente que nos permite ver páginas web, utilizaremos esta funcionalidad para realizar una búsqueda en nuestras tablas de Google Sheets mediante un enlace.

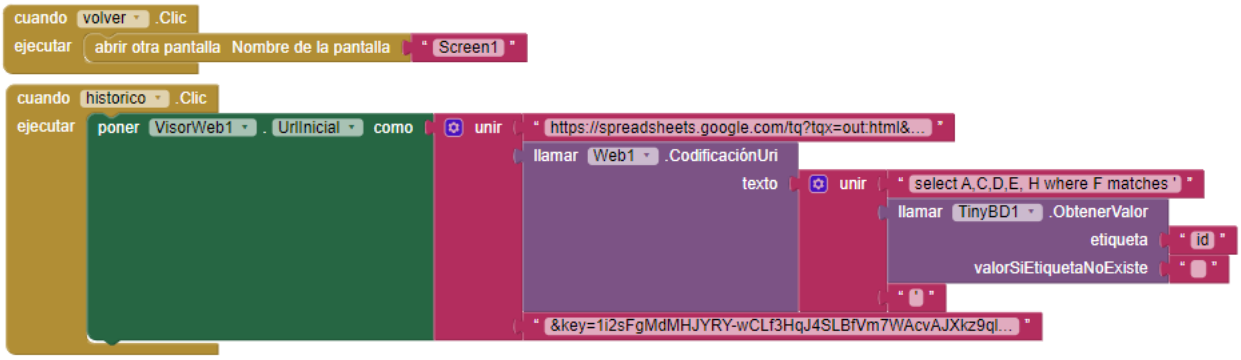
Pantalla mis_incidencidas

La consulta que realizaremos a nuestra base de datos en la hoja de incidencias será:

SELECT A,C,D,E, H where F matches 'id'

Función SELECT

Para realizar esto utilizaremos un enlace para acceder a la tabla y a este le añadiremos la consulta anterior mediante la configuración URI.



Bloque botón histórico, pantalla mis_incidencias

Como podemos ver, cuando pulsamos el botón que hemos llamado histórico ponemos en el visor web el enlace que formamos al unir el acceso a la tabla y la consulta transformada a codificación URI mediante el componente web.

Automáticamente nos aparecerá en el visor web una vista de los datos que hemos buscado, en este caso las incidencias registradas por el usuario.

También hemos creado un bloque con el cual podremos volver a la pantalla anterior.

6.9. Crear código en la App para anular incidencia

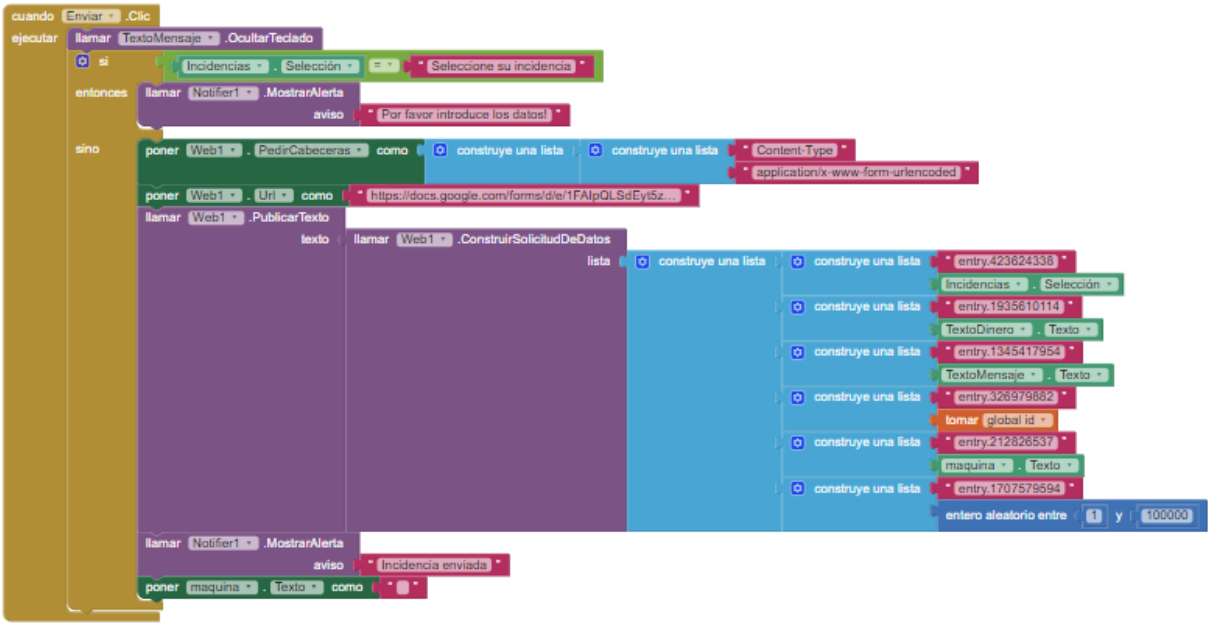
Hasta el momento en la tabla de incidencias estamos enviando datos como el problema sufrido por el cliente, la fecha y la máquina en la que se había ocasionado el problema.

Como lo que pretendemos es solucionar estos problemas, además de los datos descritos anteriormente añadiremos un código que podrá ser utilizado a la hora de solucionar la incidencia. Este código lo generaremos mediante bloques en la aplicación.

Realizaremos los mismos pasos que para enviar el id de la máquina, primero ampliamos el formulario de Google añadiendo una pregunta para el código, luego vamos al formulario, vemos el código fuente y copiamos la acción del formulario del apartado código.

Ahora modificaremos el bloque del botón enviar, ampliaremos la lista de datos que enviaremos añadiendo el código y pegaremos la etiqueta de acción que hemos copiado anteriormente.

Como hemos dicho antes el código lo crearemos desde la aplicación en este caso mediante un bloque que genera un numero aleatorio entre uno y 100000.



Bloque de botón enviar enviando código, pantalla mis_incidencias

Marca temporal	Id_Incidencia	Incidencia	Dinero	Mensaje	Id_Usuario	Id_Maquina	Código
1/06/2019 17:17:05	51	No hay agua			2	2	87533

Tabla de incidencias con columna código

6.10. Crear lista de incidencias

El sistema de automatización de incidencias que crearemos estará compuesto también de una página web en la que se podrán ver los resultados de la aplicación.

Se podrán ver los datos de forma sencilla como incidencias y estadísticas que facilitarán la resolución de los problemas ocasionados por las máquinas de café.

La página web mostrará los datos de las tablas que se encuentran en Google Sheets, dicho de otro modo, la intención es mostrar los datos que tenemos en Google Sheets pero de una forma más simplificada y atractiva.

Crearemos unos archivos para almacenar el código que será necesario para la página web, estos archivos los guardaremos en el repositorio GitHub.

GitHub es una plataforma de desarrollo colaborativo de software para guardar proyectos utilizando el sistema de versiones Git.

```
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Servicio de máquinas de café</title>
<link href="style.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<fbody>
<div id="wrapper">
        <div id="menu">
                <ul>
                        <li class="current_page_item"><a href="
">Inicio</a></li>
                        <li><a href=" ">Incidencias</a></li>
                        <li><a href=" ">Estadísticas</a></li>
                        <li><a href=" ">Mantenimiento</a></li>

                </ul>
        </div>
        <!-- end #menu -->
        <div id="header">
                <div id="logo">
                        <h1><a href="#">Servicio de máquinas de café</a></h1>
                </div>
        </div>
        <!-- end #header -->
        <!-- end #page -->
</div>
        <div id="footer">
                <p>Copyright (c) All rights reserved. Design by Julián Lozano
Moraleda.</p>
        </div>
        <!-- end #footer -->
</body>
</html>
```

Código del archivo index.html

Para crear la página web con estos archivos utilizaremos una aplicación llamada RawGit, esta aplicación actúa como un servidor de almacenamiento en caché para archivos que estén subidos a GitHub. Es básicamente una aplicación de internet que sirve archivos de GitHub a CDN externos. RawGit reenvía todas las solicitudes de usuario a GitHub y luego almacena en cache las respuestas a su navegador.

Para obtener nuestra página web únicamente tenemos que obtener la dirección del archivo en GitHub, ir a la página de RawGit pegar el enlace en el apartado de archivos en bruto y automáticamente se creará una URL que será la dirección de la web.

El aspecto de la página web que hemos creado es el siguiente, en la parte superior hay un menú en el que aparecen las opciones incidencias, estadísticas y mantenimiento para poder movernos fácilmente.

Página web. Inicio

Tal como hemos visto en la página hay una sección para incidencias, en ellas se irán publicando tablas de incidencias con diversos motivos. Una de estas tablas que se publicarán será la de las ultimas incidencias. Pero publicar las tablas tal cual no es estético, ni ofrece una vista cómoda de los datos, por ello hará que solucionar este problema.

La web nos proporciona infinidad de soluciones con las que resolver este problema, entre todas las posibilidades que hay en este caso se ha optado por utilizar Awesome Table.

Awesome table es una aplicación capaz de convertir una información simple y aburrida de una hoja de datos de Google Sheets en una vista atractiva.

Awesome table se integra perfectamente en el entorno de Google, la única condición para utilizar la aplicación web es tener una cuenta de Google.

Con esta aplicación es posible dar un determinado formato a los datos para que se vean de la manera que nosotros queramos, además estos datos se pueden ordenar forma sencilla mediante filtros interactivos.

Aparte de todo esto, esta aplicación web nos permite que incrustemos en cualquier página web el resultado que hemos obtenido.

Accedemos a la página web de Awesome table, ingresamos con nuestro usuario de Google, hacemos clic en crear una nueva vista, elegimos la hoja de Google Sheets de la que queremos extraer los datos y nos aparecerá la tabla en esta web.

1 - 15 / 58

Marca temporal	Id_Incidencia	Incidencia	Dinero	Mensaje	Id_Usuario	Id_Maquina	Código	Estado
27/02/2018 12:14:23	1	No me ha devuelto el cambio	5		10	2	130010	cerrado
27/02/2018 12:19:28	2	No hay agua			10	4	140010	cerrado
27/02/2018 12:31:25	3	No hay agua			10	5	150010	cerrado
28/02/2018 10:26:07	4	No hay café			10	2	160010	cerrado
6/03/2018 12:16:28	5	No hay vaso			12	2	110012	cerrado
6/03/2018 12:16:38	6	No hay vaso			12	3	120012	cerrado
6/03/2018 12:16:49	7	No hay azúcar			12	2	130012	cerrado
6/03/2018 12:24:49	8	El líquido está frío			10	5	140010	cerrado
6/03/2018 12:28:42	9	Cantidad insuficiente			10	5	210010	cerrado
6/03/2018 12:29:46	10	Cantidad insuficiente			10	2	220010	cerrado
6/03/2018 18:49:15	11	No hay agua			10	5	230010	cerrado
6/03/2018 18:49:50	12	No hay agua			10	5	240010	cerrado
6/03/2018 19:04:35	13	No hay agua			9	3	250009	cerrado
6/03/2018 19:05:02	14	No hay agua			9	3	260009	cerrado
6/03/2018 19:05:23	15	No hay agua			9	4	270009	cerrado

1 - 15 / 58

Tabla de incidencias de Awesome table

La tabla que nos aparece a simple vista a cambiado muy poco, en las características podemos elegir el rango de datos de la tabla que queremos mostrar.

Otra de las opciones que nos ofrece esta aplicación web es añadir los filtros que nosotros queramos a nuestra tabla. Esto se hace escribiendo en las columnas de la tabla de Google Sheets justo encima de los datos el tipo de filtro que queremos añadir. Como se muestra en la siguiente imagen.

Marca temporal	Id_Incidencia	Incidencia	Dinero	Mensaje	Id_Usuario	Id_Maquina	Código	Estado
DateFilter	NoFilter	CategoryFilter	NumberRangeFilter	NoFilter	csvFilter	csvFilter	NoFilter	csvFilter

Tabla de Google Sheets con filtros para Awesome table

Ahora volvemos a cargar la tabla en Awesome table y nos aparecerán los filtros que hemos elegido, con los que podremos hacer búsquedas en el tiempo, elegir un usuario o máquina en concreto, el tipo de incidencia o el estado en el que se encuentra, que podrá ser abierto o cerrado.

Marca temporal 27/02/2018 02/06/2019 | Incidencia | Dinero 1 18 | Id_Usuario | Id_Maquina | Estado

1 - 15 / 58

Marca temporal	Id_Incidencia	Incidencia	Dinero	Mensaje	Id_Usuario	Id_Maquina	Código	Estado
27/02/2018 12:14:23	1	No me ha devuelto el cambio	5		10	2	130010	cerrado
27/02/2018 12:19:28	2	No hay agua			10	4	140010	cerrado
27/02/2018 12:31:25	3	No hay agua			10	5	150010	cerrado
28/02/2018 10:26:07	4	No hay café			10	2	160010	cerrado
6/03/2018 12:16:28	5	No hay vaso			12	2	110012	cerrado
6/03/2018 12:16:38	6	No hay vaso			12	3	120012	cerrado
6/03/2018 12:16:49	7	No hay azúcar			12	2	130012	cerrado
6/03/2018 12:24:49	8	El líquido está frío			10	5	140010	cerrado
6/03/2018 12:28:42	9	Cantidad insuficiente			10	5	210010	cerrado
6/03/2018 12:29:46	10	Cantidad insuficiente			10	2	220010	cerrado
6/03/2018 18:49:15	11	No hay agua			10	5	230010	cerrado
6/03/2018 18:49:50	12	No hay agua			10	5	240010	cerrado
6/03/2018 19:04:35	13	No hay agua			9	3	250009	cerrado
6/03/2018 19:05:02	14	No hay agua			9	3	260009	cerrado
6/03/2018 19:05:23	15	No hay agua			9	4	270009	cerrado

1 - 15 / 58

Tabla de incidencias de Awesome table

Una de las ventajas de utilizar Awesome table es que podemos empotrar las tablas que creemos en cualquier página web que diseñemos. Para ello hacemos clic en compartir, después en empotrar y copiamos el código que nos aparece, este lo añadiremos al de nuestra página web.

Para publicar las tablas crearemos otro archivo HTML, este formara parte de la página web. En este archivo pegaremos el código de la tabla que hemos obtenido. En esta tabla se muestran todas las incidencias producidas en las máquinas de café.

```
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Servicio de máquinas de café</title>
<link href="style.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<fbody>
<div id="wrapper">
        <div id="menu">
                <ul>
                        <li><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/index.html#">Inicio</
a></li>
                        <li class="current_page_item"><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/tablas.html#">Inciden
cias</a></li>
                        <li><a href=" ">Estadísticas</a></li>
                        <li><a href=" ">Mantenimiento</a></li>
                </ul>
        </div>
        <!-- end #menu -->
```

```
        <div id="header">
                <div id="logo">
                        <h1><a href="#">Servicio de máquinas de café</a></h1>
                </div>
        </div>
        <!-- end #header -->
        <div id="page">
        <div id="page-bgtop">
        <div id="page-bgbtm">
                <div id="content">
                        <div class="post">
                                <h2 class="title"><a href="#">Todas las
incidencias </a></h2>
                                <div class="entry">
                                        <p>Texto </p>
<div data-type="AwesomeTableView" data-viewID="-LgWBXkcmLbs7FKS90Hx"></div>
<script src="https://awesome-table.com/AwesomeTableInclude.js"></script>
                                </div>
                        </div>
        <!-- end #page -->
</div>
        <div id="footer">
                <p>Copyright (c) All rights reserved. Design by Julián Lozano
Moraleda.</p>
        </div>
        <!-- end #footer -->
</body>
</html>
```

Código del archivo tablas.html

6.11. Enviar correo electrónico al iniciar incidencia.

Uno de los objetivos de este trabajo es mantener conectados al usuario con la plataforma que se encargara de solucionar las incidencias. Por ello el usuario una vez registrado y mandado la incidencia recibirá un correo electrónico notificándole que la incidencia se ha enviado correctamente. Además, en este correo irá el código de la incidencia, el usuario podrá introducir este código en la aplicación para poder dar por solucionada su incidencia.

Para poder realizar la función de mandar el correo automáticamente utilizaremos un archivo de Google Script. Para ello iremos a una hoja de Google Sheets y desde herramientas accedemos al editor de Script, crearemos el siguiente archivo.

```
function onEditEnvioMail(e) {
Logger.log('onEditEnvioMail');
```

```
  Logger.log(e);
  Logger.log(e.range.getA1Notation());

  var values = e.values;
  var user = values[5];

  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var userv = lookUpUserId(ss.getSheets()[0], user, 2, 5);

  var direccion = userv[3];
  var usuario = userv[2];
  var fecha = values[0];
  var incidencia = values[2];
  var cantidad = values[3];
  var comentario = values[4];
  var idmaquina = values[6];
  var codigo = values[7];

  var otro;
  var texto1 = "No me ha devuelto el cambio";

  if (incidencia = otro) {
   var mensaje = 'Hola '+usuario+' hemos recibido su incidencia '+comentario+' le avisaremos
cuando el problema este resuelto, disculpe las molestias.';
} else if (incidencia = texto1) {
   var mensaje = 'Hola '+usuario+' hemos recibido su incidencia le avisaremos cuando
hayamos dejado la cantidad de '+cantidad+' a su disposion.';
} else {
   var mensaje = 'Hola '+usuario+' hemos recibido su incidencia '+incidencia+' le avisaremos
cuando el problema este resuelto, disculpe las molestias.';
}

  var hoy = new Date();
  var mes = Utilities.formatDate(hoy,Session.getTimeZone(), "MM");
  var dia = Utilities.formatDate(hoy,Session.getTimeZone(), "dd");
  var fecha = dia+"/"+mes;
  var asunto ='Incidencia recivida : '+fecha+' ';

   MailApp.sendEmail(direccion, asunto, mensaje);
}

function lookUpUserId(sheet, id, col, numCols){
  var columnValues = sheet.getRange(2, col, sheet.getLastRow()).getValues();
  var found = columnValues.findIndex(id); //Row Index - 2
  if(found== -1) return null;

  return sheet.getSheetValues(found + 2,1,1,numCols)[0];
}

Array.prototype.findIndex = function(search){
  if(search == "") return false;
  for (var i=0; i<this.length; i++)
```

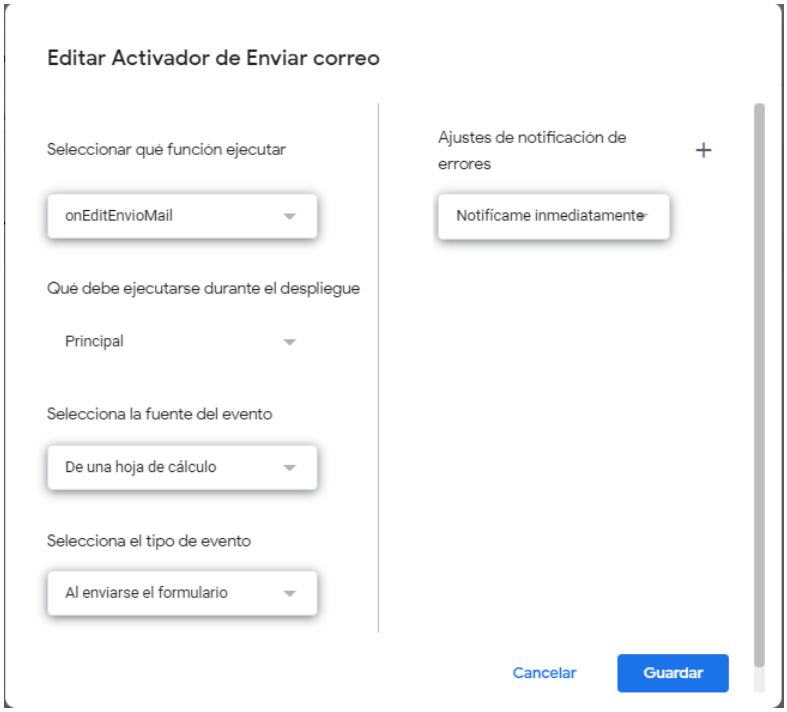
```
    if (this[i] == search) return i;
  return -1;
}
```

Script de Google Enviar correo

Al activarse este script obtiene los datos que el usuario registre en la tabla de incidencias mediante el formulario de la aplicación, de entre estos datos utiliza el *Id_Usuario* para buscar en la tabla de registro de usuario los datos de correo y usuario relacionados con ese id mediante la función *lookUpUserId*. Ya con todos estos datos, envía un correo utilizando la API de Gmail MailApp.sendEmail.

Para que la función de mandar el correo se realice automáticamente crearemos un trigger, desde la opción editar del menú accedemos a nuestros triggers y creamos uno nuevo.

Seleccionamos la función *onEditEnvioMail*, erigiremos una hoja de cálculo como fuente del evento, como evento elegimos al enviarse el formulario y le damos a guardar.



Trigger de función onEditEmvioMail

6.12. Cerrar incidencias desde la web y enviar correo electrónico

Además de proporcionar los datos generados por el sistema también funcionará como vínculo con el personal de mantenimiento proporcionando una pantalla útil para llevar un seguimiento del mantenimiento de las maquinas.

Dentro de la página web crearemos una sección para llevar el mantenimiento de la máquina, para ello crearemos otro archivo HTML, copiaremos el código del archivo index.htm y añadiremos un formulario, quedando de la siguiente forma.

```
    <head>
```

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Servicio de máquinas de café</title>
<link href="style.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<fbody>
<div id="wrapper">
        <div id="menu">
                <ul>
                        <li><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/index.html#">Inicio</
a></li>
                        <li><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/tablas.html#">Inciden
cias</a></li>
                        <li><a href=" ">Estadísticas</a></li>
                        <li class="current_page_item"><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/mantenimiento.html#">
Mantenimiento</a></li>
                </ul>
        </div>
        <!-- end #menu -->
        <!-- formulario -->
<form name="submit-to-google-sheet">
  <input id="ID_Maquina" name="ID_Maquina" type="codigo" placeholder="Nº de
la maquina" required>
  <button type="submit">Send</button>
</form>
<script>
    const scriptURL =
'https://script.google.com/macros/s/AKfycbxpkKbcPXO9WIaAYk5PFoMvbDihZF2m2geDQ
pbRy0K4-W7su68g/exec'
    function setupListener(form) {
        form.addEventListener('submit', e => {
        e.preventDefault()
        fetch(scriptURL, { method: 'POST', body: new FormData(form)})
                .then(function(response) {
                        document.getElementById('ID_Maquina').value=''
                        console.log('Success!', response)
                })
                .catch(error => console.error('Error!', error.message))
        })
    }
    const form = document.forms['submit-to-google-sheet']
    setupListener(form)
</script>
                <!-- end #sidebar -->
```

```
                <div style="clear: both;"> </div>
            </div>
            </div>
            </div>
            <!-- end #page -->
    </div>
            <div id="footer">
                <p>Copyright (c) All rights reserved. Design by Julián Lozano
    Moraleda.</p>
            </div>
            <!-- end #footer -->
    </body>
    </html>
```

Código del archivo mantenimiento.html

Formulario página web, mantenimiento

El formulario que hemos creado dispondrá de un menú en el que el operario de mantenimiento podrá actualizar el estado de las máquinas, una vez que ha realizado su puesta a punto. Así una vez que se ha actualizado el estado de una máquina, todas las incidencias de esta se darán por finalizadas.

La función de este formulario será enviar el numero de la maquina junto con la fecha a una hoja de Google Sheets mediante el método POST.

El formulario de la página estará enlazado con el siguiente archivo de Google Script:

```
var sheetName = 'anular'
/*
Obtiene un almacén de propiedades al que todos los usuarios pueden acceder, pero solo
dentro de este script.
*/
var scriptProp = PropertiesService.getScriptProperties()
/*
Esta es la función de configuración inicial. Obtiene el ID de SpreadsheetApp activo y lo
agrega a nuestro PropertiesService.
*/
function intialSetup () {
  var activeSpreadsheet = SpreadsheetApp.getActiveSpreadsheet()
  scriptProp.setProperty('key', activeSpreadsheet.getId())
}
function doPost (e) {
/*
```

```
Obtiene un bloqueo que impide que cualquier usuario ejecute simultáneamente una sección
de código. Una sección de código
protegido por un bloqueo de script no se puede ejecutar simultáneamente,
independientemente de la identidad del usuario.
*/
  var lock = LockService.getScriptLock()
/*
Intenta adquirir el bloqueo y se agota el tiempo de espera con una excepción después del
número de milisegundos proporcionado.
Este método es el mismo que tryLock (timeoutInMillis), excepto que lanza una excepción
cuando el bloqueo
No se pudo adquirir en lugar de devolver el falso.
*/
  lock.tryLock(10000)
  try {
/*
Abre la hoja de cálculo con la ID dada. Un ID de hoja de cálculo se puede extraer de su URL.
Por ejemplo,
El ID de la hoja de cálculo en la URL
https://docs.google.com/spreadsheets/d/abc1234567/edit#gid=0 es "abc1234567".
*/
    var doc = SpreadsheetApp.openById(scriptProp.getProperty('key'))
/*
Devuelve una hoja con el nombre dado. Si varias hojas tienen el mismo nombre,
Se devuelve el de la izquierda. Devuelve nulo si no hay ninguna hoja con el nombre dado.
*/
    var sheet = doc.getSheetByName(sheetName)
/*
Devuelve el rango con la celda superior izquierda en las coordenadas dadas y con el número
dado de filas.
Luego devuelve la posición de la última columna que tiene contenido.
Luego devuelve la cuadrícula rectangular de valores para este rango (una matriz
bidimensional de valores, indexada por fila y luego por columna).
*/
    var headers = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0]
// Obtiene la última fila y luego agrega una
    var nextRow = sheet.getLastRow() + 1
/*
Asigna la matriz de encabezados a una nueva matriz. Si el valor de un encabezado es
'timestamp', entonces
devuelve un nuevo objeto Date (), de lo contrario, devuelve el valor del parámetro de URL
correspondiente
*/
    var newRow = headers.map(function(header) {
      return header === 'timestamp' ? new Date() : e.parameter[header]
    })
/*
Obtiene un rango desde la fila siguiente hasta la fila final según la cantidad de elementos
que hay en newRow
luego establece los nuevos valores de toda la matriz a la vez.
*/
    sheet.getRange(nextRow, 1, 1, newRow.length).setValues([newRow])
```

```
/*
Devolver resultados exitosos como JSON
*/
   return ContentService
     .createTextOutput(JSON.stringify({ 'result': 'success', 'row': nextRow }))
     .setMimeType(ContentService.MimeType.JSON)
  }
  catch (e) {
   return ContentService
     .createTextOutput(JSON.stringify({ 'result': 'error', 'error': e }))
     .setMimeType(ContentService.MimeType.JSON)
  }
  finally {
/*
Libera el bloqueo, permitiendo que otros procesos que esperan en el bloqueo continúen.
*/
   lock.releaseLock()
  }
}
```

Script, enviar id de máquinas

Publicaremos el script como web app, se nos generara el siguiente enlace: 'https://script.google.com/macros/s/AKfycbxpkKbcPXO9WIaAYk5PFoMvbDihZF2m2geDQpbRy0K4-W7su68g/exec', este lo pegaremos en el archivo mantenimiento.html a continuación de *const scriptURL =.*

De esta forma escribiremos una fila con la maquina actualizada y la fecha en la que se ha realizado el mantenimiento en la hoja de Google Sheets que hemos llamado *anular* quedando de la siguiente forma:

	A	B
1	timestamp	ID_Maquina
2	22/01/2019	5
3	22/01/2019	3
4	22/01/2019	3
5	22/01/2019	3
6	22/01/2019	6

Tabla de máquinas actualizadas

6.12.1 Enviar correo al anular incidencia

Una de las tareas que queremos que realice la plataforma que estamos creando es la de avisar al usuario afectado a la hora de abrir una incidencia como ya hemos hecho, pero también cuando se cierre.

El cliente tendrá la posibilidad de cerrar las incidencias que el haya registrado, pero habrá muchos tipos de incidencias, y algunas no se cerraran por medio del consumidor. Ahora ya tenemos registrado en la base cuando se le ha hecho mantenimiento a cada máquina y este es un dato que podemos utilizar para cerrar las incidencias, ya que una vez que se le ha hecho la puesta a punto, la maquina debería funcionar correctamente.

Para utilizar este dato crearemos el siguiente código en un script de Google.

```
function onEditEnvioMail(e) {

    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getSheets()[2];

    // obtenemos el ultimo valor de esa tabla
    var lastRow = sheet.getLastRow();
    var lastColumn = sheet.getLastColumn();
    var lastCell = sheet.getRange(lastRow, lastColumn);
    var ultima = lastCell.getValue();

    if (ultima == 'ID_Maquina') {
     var mensaje = "la tabla anular está vacía";
    } else {
     var mensaje = "la tabla anular tiene valores";

      //buscamos los datos de incidencia de esa máquina
      var user = ultima;
     var ss = SpreadsheetApp.getActiveSpreadsheet();
     var userv = lookUpUserId(ss.getSheets()[1], user, 7, 6);
     var fila = devuelvefila(ss.getSheets()[1], user, 7, 6)+2;
     var columna_Estado ='I';
     var rango = (columna_Estado + fila);

    cerrar(rango)

     var fecha = userv[0];
     var incidencia = userv[2];
     var id_usuario = userv[5];

     //busco los datos de usuario
     var ss1 = SpreadsheetApp.getActiveSpreadsheet();
     var datos_usuario = lookUpUserId(ss1.getSheets()[0], id_usuario, 2, 5);

     var usuario = datos_usuario[2];
     var correo = datos_usuario[3];

    var mensaje ='hola '+usuario+' su incidencia '+incidencia+' con fecha '+fecha+' ya está
solucionada';
    var asunto ='Incidencia solucionada ';
MailApp.sendEmail(correo, asunto, mensaje);

    //funcion borrar filas por valor en columna
   readRows(ultima)

 }

function lookUpUserId(sheet, maquina, col, numCols){
  var columnValues = sheet.getRange(2, col, sheet.getLastRow()).getValues();
  var found = columnValues.findIndex(maquina); //Row Index - 2
  if(found== -1) return null;
  return sheet.getSheetValues(found + 2,1,1,numCols)[0];
```

```
}

function devuelvefila(sheet, maquina, col, numCols){
  var columnValues = sheet.getRange(2, col, sheet.getLastRow()).getValues();
  var found = columnValues.findIndex(maquina); //Row Index - 2
  if(found== -1) return null;
  return found
}

     function readRows(ultimo) {
      var ss = SpreadsheetApp.getActiveSpreadsheet();
      var sheet = ss.getSheets()[2];
      var rows = sheet.getDataRange();
      var numRows = rows.getNumRows();
      var values = rows.getValues();
      var rowsDeleted = 0;
      for (var i = 0; i <= numRows - 1; i++) {
      var row = values[i];
      if (row[1] == ultimo) {
      sheet.deleteRow((parseInt(i)+1) - rowsDeleted);
      rowsDeleted++;
     }
      }
     }

    function cerrar(rango){
     var ss = SpreadsheetApp.getActiveSpreadsheet().getSheets()[1];
     var cell = ss.getRange(rango);
     var bold = SpreadsheetApp.newTextStyle().setBold(true).build();
     var value = SpreadsheetApp.newRichTextValue()
     .setText("cerrado")
     .setTextStyle(0, 5, bold)
     .build();
    cell.setRichTextValue(value);
}

}

Array.prototype.findIndex = function(search){
  if(search == "") return false;
  for (var i=0; i<this.length; i++)
    if (this[i] == search) return i;
  return -1;
}
```

Script de Google Enviar_correo_resuelto

El script obtiene el ultimo valor escrito en la tabla de máquinas actualizadas y busca ese valor en la columna de *ID_Maquina* de la tabla de registro de incidencias, obtiene los datos de las incidencias que coinciden con ese id de máquina y además se escribe en la columna de estado el valor *cerrado*. A continuación, hace una búsqueda con el dato de *Id_Usuario* en la tabla de registro de usuarios, de aquí obtiene el nombre y el correo del usuario y con estos datos ya

puede enviar un correo al cliente notificándole que la incidencia que registró ya se ha solucionado.

Para que este script se active solo crearemos un trigger de la misma forma que en el programa *Enviar correo.* Creamos uno nuevo, erigiremos una hoja de cálculo como fuente del evento y escogemos que se active al editarse el documento.

6.13. Cerrar incidencias desde la aplicación

Para que el usuario de la maquina pueda cerrar la incidencia cuando este vea que se ha resuelto su problema habrá que poner a su disposición un mecanismo con el cual pueda realizar esta acción.

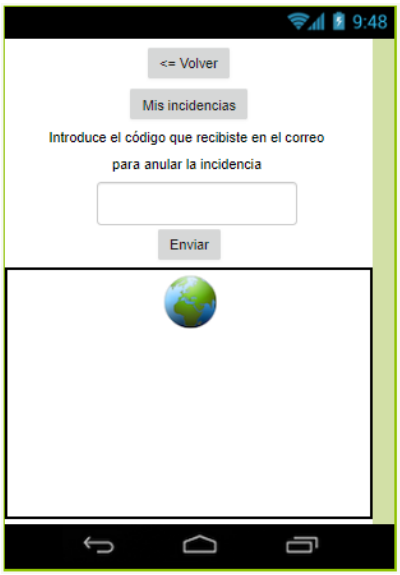
Iremos a Google Sheets desde donde crearemos una nueva hoja y desde esta un formulario con el campo código. Una vez hecho, iremos al código fuente de este formulario y copiaremos la etiqueta de acción de la pregunta código.

Ahora volveremos App Inventor y añadiremos los siguientes componentes a la pantalla de diseño *mis_incidencias*:

-Una caja de texto que renombraremos *código*, este será el espacio en el que el usuario podrá introducir el código.

-Un botón, al que nombraremos *BotonAnular* que será el encargado de enviar el código.

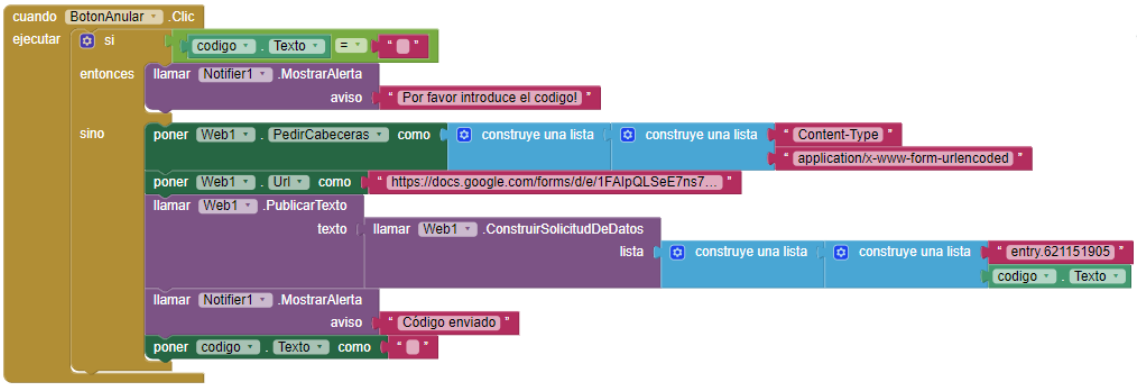
-El elemento de notificación, que nos ayudara a notificar cualquier fallo que se produzca.



Pantalla mis_incidencias con campo de texto y botón para anular incidencia

Una vez añadidos los componentes pasaremos a editar los bloques para poder realizar el trabajo descrito.

Crearemos un bloque que al hacer clic sobre el *BotonAnular* enviará mediante un post el código a la base de datos. Lo haremos de la misma forma que para registrarnos o enviar la incidencia, la diferencia es que para este bloque utilizaremos la etiqueta de código que hemos copiado anteriormente.



Bloque BotonAnular, pantalla mis_incidencias

Después de realizar estos cambios, al escribir el código en la caja de texto y pulsar el *BotonAnular* se enviará junto con la fecha a la tabla elegida de nuestra base de datos.

Fecha_resolucion	Codigo
9/01/2019 22:43:55	340016
9/01/2019 22:44:15	350016
9/01/2019 23:18:11	360016

Tabla de código recibido

Una vez que somos capaces de enviar el código a la base de datos podemos utilizarlo para hacer efectiva la anulación o cierre de la incidencia.

Para ello crearemos el siguiente script:

```
function cerrardesdeApp(e) {

    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getSheets()[3];
    var lastRow = sheet.getLastRow();
    var lastColumn = sheet.getLastColumn();
    var lastCell = sheet.getRange(lastRow, lastColumn);
    var ultima = lastCell.getValue();

    if (ultima == 'Codigo') {
    var mensaje = "la tabla anular está vacía";
    } else {
    var mensaje = "la tabla anular tiene valores";
```

```
 //busco los datos de incidencia de ese código
 var user = ultima;

 var ss = SpreadsheetApp.getActiveSpreadsheet();
 var userv = lookUpUserId(ss.getSheets()[1], user, 8, 6);
 var fila = devuelvefila(ss.getSheets()[1], user, 8, 6)+2;
 var columna_estado ='I';
 var rango = (columna_estado + fila);
 cerrar(rango)

 var fecha = userv[0];
 var incidencia = userv[2];
 var id_usuario = userv[5];

 var ss1 = SpreadsheetApp.getActiveSpreadsheet();
 var datos_usuario = lookUpUserId(ss1.getSheets()[0], id_usuario, 2, 5);
 var usuario = datos_usuario[2];
 var correo = datos_usuario[3];

 var mensaje ='hola '+usuario+' su incidencia '+incidencia+' con fecha '+fecha+' se ha
cerrado';
 var asunto ='Incidencia cerrada ';
 MailApp.sendEmail(correo, asunto, mensaje);
 }

function lookUpUserId(sheet, maquina, col, numCols){
   var columnValues = sheet.getRange(2, col, sheet.getLastRow()).getValues();
 Logger.log(columnValues);
 var found = columnValues.findIndex(maquina); //Row Index - 2
 if(found== -1) return null;
 return sheet.getSheetValues(found + 2,1,1,numCols)[0];
}

function devuelvefila(sheet, maquina, col, numCols){
 var columnValues = sheet.getRange(2, col, sheet.getLastRow()).getValues();
 var found = columnValues.findIndex(maquina); //Row Index - 2
 if(found== -1) return null;
 return found
}

    function cerrar(rango){
     var ss = SpreadsheetApp.getActiveSpreadsheet().getSheets()[1];
     var cell = ss.getRange(rango);
     var bold = SpreadsheetApp.newTextStyle().setBold(true).build();
     var value = SpreadsheetApp.newRichTextValue()
     .setText("cerrado")
     .setTextStyle(0, 5, bold)
     .build();
  cell.setRichTextValue(value);
}
}
```

```
Array.prototype.findIndex = function(search){
  if(search == "") return false;
  for (var i=0; i<this.length; i++)
    if (this[i] == search) return i;
  return -1;
}
```

Script anular_desde_aplicacion

Este script obtiene los datos de código y fecha que enviamos desde la aplicación, con el dato del código buscaremos en la tabla de registro de incidencias los datos de la incidencia relacionada con él mediante la función *lookUpUserId* y cuando los encontramos utilizamos la función *cerrar* para escribir *cerrado* en la columna de estado.

Para que se active automáticamente tendremos que crear un nuevo trigger que haga que se active al enviarse un formulario.

6.14. Ver estadísticas desde la página web

Para llevar un control de lo que ocurre con las máquinas y llevar un seguimiento de los problemas más habituales, o conocer en que máquinas se ocasionan el mayor número de incidencias, crearemos una sección en la página web a la que llamaremos estadísticas.

En esta sección publicaremos gráficos actualizados con las estadísticas de las máquinas e incidencias.

Para crear un gráfico en Google Sheets simplemente tenemos que acceder a la opción insertar del menú, elegimos gráfico y nos aparecerá en la pantalla el editor de gráficos. Elegimos la columna de la que queremos exportar los datos, elegimos el tipo de gráfico que queremos y nos aparecerá en la pantalla.

En este caso hemos creado gráficos relacionados con el tipo de incidencia y las maquinas que presentan más fallos, clasificados en un nivel general y también en la última semana.

Incidencias más habituales:

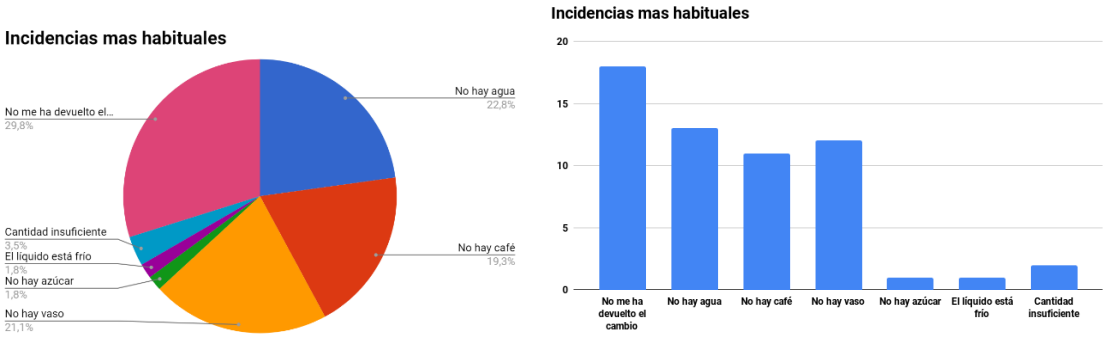


Gráfico de incidencias más habituales

Máquinas con mayor número de incidencias:

Gráfico de máquinas con más incidencias

Incidencias más comunes en la última semana:

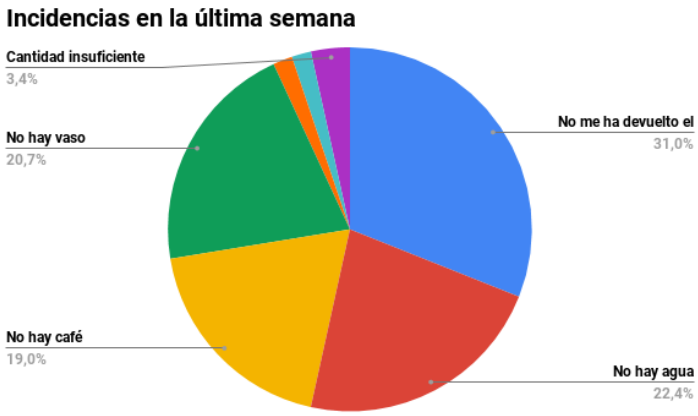


Gráfico de incidencias en la última semana

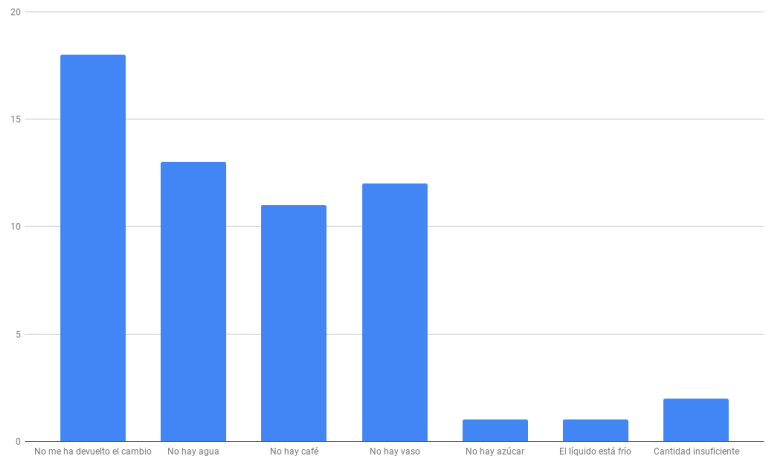


Gráfico de incidencias en la última semana II

La idea es subir estos archivos a la página web. Para ello crearemos otro archivo HTML, copiamos el código del archivo index.html y lo pegaremos en el que llamaremos estadísticas.html.

Para subir los gráficos que hemos creado, únicamente tendremos que pinchar en cada gráfico y elegir la opción publicar el grafico, copiaremos el código que nos aparece y lo pegaremos en nuestro archivo HTML.

```
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Servicio de máquinas de café</title>
<link href="style.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<fbody>
<div id="wrapper">
        <div id="menu">
                <ul>
                        <li><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/index.html#">Inicio<
/a></li>
                        <li><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/tablas.html#">Incide
ncias</a></li>
                        <li class="current_page_item"><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/estadisticas.html#">
Estadísticas</a></li>
                        <li><a
href="https://rawgit.com/Julian92UCLM/cosmos/master/web/mantenimiento.html#"
>Mantenimiento</a></li>
                </ul>
        </div>
        <!-- end #menu -->
```

```
        <div id="header">
                <div id="logo">
                        <h1><a href="#">Servicio de máquinas de café</a></h1>
                </div>
        </div>
        <!-- end #header -->
        <div id="page">
        <div id="page-bgtop">
        <div id="page-bgbtm">
                <div id="content">
                        <div class="post">
                                <h2 class="title"><a href="#">Incidencias más
comunes  </a></h2>
                                <div class="entry">
                                        <p>En estos gráficos se muestran las
incidencias clasificadas por porcentajes y por cantidad.</p>
                                        <iframe width="600" height="371" seamless
frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/e/2PACX-
1vRJEwR9Gmj9lX9yAydEgjGyegMspLNPvsJuiKC6EBst1EIMIpKqam68HOn8zKTCzgZv7qxIeD0L
pHzH/pubchart?oid=1308194102&amp;format=interactive"></iframe>
                                        <iframe width="718" height="444" seamless
frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/e/2PACX-
1vRJEwR9Gmj9lX9yAydEgjGyegMspLNPvsJuiKC6EBst1EIMIpKqam68HOn8zKTCzgZv7qxIeD0L
pHzH/pubchart?oid=1479895854&amp;format=interactive"></iframe>
                                </div>
                        </div>
                        <div class="post">
                                <h2 class="title"><a href="#">Máquinas con mayor
número de incidencias. </a></h2>
                                <div style="clear: both;"> </div>
                                <div class="entry">
                                        <p>En el siguiente gráfico se muestran
las máquinas con mayor número de incidencias</p>
                                        <iframe width="600" height="371" seamless
frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/e/2PACX-
1vRJEwR9Gmj9lX9yAydEgjGyegMspLNPvsJuiKC6EBst1EIMIpKqam68HOn8zKTCzgZv7qxIeD0L
pHzH/pubchart?oid=1204930832&amp;format=interactive"></iframe>
                                </div>
                        </div>
                        <div class="post">
                                <h2 class="title"><a href="#">Incidencias en la
última semana. </a></h2>
                                <div class="entry">
```

```
                                        <p>En los siguientes gráficos se muestran
las incidencias de la última semana clasificadas por porcentajes y por
cantidad. </p>
                                        <iframe width="600" height="371" seamless
frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/e/2PACX-
1vRJEwR9Gmj9lX9yAydEgjGyegMspLNPvsJuiKC6EBst1EIMIpKqam68HOn8zKTCzgZv7qxIeD0L
pHzH/pubchart?oid=567844230&amp;format=interactive"></iframe>
                                        <iframe width="631.5099125390424"
height="390.61018629993794" seamless frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/e/2PACX-
1vRJEwR9Gmj9lX9yAydEgjGyegMspLNPvsJuiKC6EBst1EIMIpKqam68HOn8zKTCzgZv7qxIeD0L
pHzH/pubchart?oid=719280731&amp;format=interactive"></iframe>
                                </div>
                        </div>

                <div style="clear: both;"> </div>
                </div>
                <!-- end #content -->
                <!-- end #sidebar -->
                <div style="clear: both;"> </div>
        </div>
        </div>
        </div>
        <!-- end #page -->
</div>
        <div id="footer">
                <p>Copyright (c) All rights reserved. Design by Julián Lozano
Moraleda.</p>
        </div>
        <!-- end #footer -->
</body>
</html>
```

Código del archivo estadisticas.html

6.15. Crear etiquetas para devolver el dinero

Cuando la incidencia ocurrida en una máquina sea de naturaleza económica debido a una devolución errónea en el cambio, la incidencia se dará por terminada una vez que la cantidad de dinero sea puesto en un sobre con el nombre del usuario y la razón de la incidencia.

La plataforma creará las etiquetas con los datos del usuario, estas irán en los sobres donde ira el dinero. Diseñaremos las etiquetas mediante una Awesome table, así mostraremos los datos de las tablas dándoles el aspecto que queramos con una plantilla.

Lo primero que haremos será filtrar los datos que queremos en otra tabla de Google Sheets filtraremos los datos de las incidencias con la siguiente consulta, pasando únicamente las incidencias que se encuentren abiertas y que se deban a una devolución errónea del dinero.

```
=QUERY'Formulario incidencias'!A4:J ; "select A, B, C, D, E, F, G, H
where (I != 'cerrado' and C = 'No me ha devuelto el cambio')")
```

Consulta para filtrar los datos de la tabla de formulario de incidencias

Una vez que tenemos los datos de las incidencias utilizaremos la columna *Id_Usuario* para agregar los datos de usuario y el correo de la tabla de registro de usuarios con la función *VLOOKUP.*

```
=VLOOKUP(F:F;'Formulario registro'!B:D;2;0)
```

Función VLOOKUP para filtrar añadir los datos de usuario

Ahora que ya tenemos una tabla con los datos que necesitamos iremos a Awesome table y cargaremos esta tabla. Una vez cargada y elegido el rango añadiremos un formato para nuestra tabla en parámetros avanzados, esta apariencia estará dada por un programa que hayamos escrito en una hoja de Google Sheets, este se aplicará agregando la dirección de la celda en el que lo hayamos guardado.

Existen muchas posibilidades para crear una apariencia más atractiva para la tabla, en este caso el aspecto de los datos lo dará el siguiente texto en lenguaje html.

```
<p>-----------------------------------------------------------------------------------------------------</p>
<span style="font-size:20px">Fecha:    ${"Timestamp"}</span><br/>
<span style="font-size:20px">Usuario:    ${"Usuario"}</span><br>
<span style="font-size:20px"><p>Incidencia:    ${"Incidencia"}</span> <br>
<span style="font-size:20px">Cantidad:    ${"Dinero"}</span>
```

Plantilla para tabla de incidencias

Al aplicar esta configuración veremos la tabla con el siguiente aspecto.

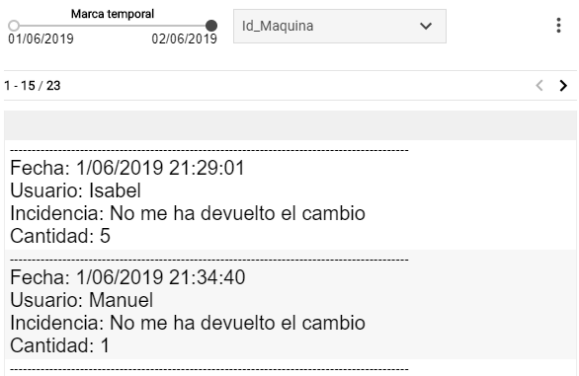


Tabla de etiquetas, Awesome table

Una vez creada la tabla copiaremos el código para pegarlo en el archivo tablas.html

6.16. Obtener el tiempo medio de resolución de incidencias

Otro de los datos que pueden tener gran interés para el estudio de las incidencias es el tiempo que tardan en resolverse las incidencias. Para obtener este dato crearemos la tabla *tiempo_medio*.

El tiempo que tardan en resolverse las incidencias lo hallaremos calculando el tiempo que pasa entre la fecha de inicio y la de final de la incidencia. Para ello haremos dos consultas una para pasar los datos las incidencias que contienen la fecha de inicio desde la tabla de incidencias, y otra consulta para pasar los datos que contienen el código y la fecha en la que se ha cerrado la incidencia de código recibido.

Una vez tenemos los datos que necesitamos en esta tabla enlazaremos los datos de fecha inicial y final con el dato *Código* mediante la función *VLOOKUP.* Una vez lo hemos hecho tendremos los datos de la siguiente forma:

E	F	G
Fecha_resolucion	Codigo	Fecha de inicio
9/05/2019 22:43:55	34905	1/06/2019 22:05:56
9/05/2019 22:44:15	21398	1/06/2019 22:09:19
9/05/2019 23:18:11	54144	1/06/2019 22:09:57
20/05/2019 18:00:03	52992	1/06/2019 22:37:13

Fecha inicial y final, tabla tiempo_medio

Ahora que tenemos los datos que necesitamos, utilizaremos la siguiente función para calcular los días que han transcurrido.

```
=MINUS(G;E)
```

Función MINUS

Tras calcular el tiempo pasado entre cada par de fechas seleccionaremos todos los resultados y calcularemos la media de estos con la siguiente función.

```
=AVERAGE(H)
```

Función AVERAGE

El resultado obtenido será el tiempo medio en resolverse las incidencias este dato lo añadiremos también a la página web, en el archivo *estadísticas.html*, utilizaremos también Awesome table, copiaremos el código y lo pegaremos en el documento *estadísticas.html.*

6.17. Crear código QR para descargar aplicación

Para facilitar a los usuarios la descarga de la aplicación crearemos un código QR con la intención de que este código redireccione al cliente mediante un enlace al repositorio, donde se encuentra la aplicación. Para ello podemos hacerlo en cualquier web que haga códigos QR, en este caso será en *https://es.qr-code-generator.com/*, y el enlace que queremos vincular será https://github.com/Julian92UCLM/cosmos/blob/master/ayuda_maquina_de_cafe.apk

Una vez creado, lo colocaremos en todas las máquinas de modo que este accesible a los clientes.

De este modo se podrá descargar de una forma sencilla desde cualquier smartphone.

Código QR de acceso a la aplicación

Capítulo 7

Discusión de resultados y trabajo futuro

Una vez terminado el trabajo tenemos una solución al problema. Al principio de este proyecto nos habíamos marcado dos objetivos, simplificar el proceso de reclamación ante fallos y monitorización del sistema.

Hemos creado una aplicación con la que se podrán resolver los problemas que se ocasionan en una máquina de café, el cliente podrá instalarla en un smartphone o tablet y comunicarse con el servicio de mantenimiento. Esta aplicación la hemos creado con App Inventor ya que nos permitía programar la aplicación mediante bloques.

La aplicación que hemos creado contiene formularios con los que el cliente se podrá registrar y detallar su problema de manera precisa. Para registrar los datos de estos formularios hemos utilizado como base de datos Google Sheets. Los datos de los formularios se pasan a la base de datos mediante formularios de Google y archivos de Google Script, en los que están los programas que hemos creado para la gestión de la base de datos. La aplicación también enviará un código que creará ella misma para cada incidencia, este código servirá para cerrar la incidencia una vez que se haya solucionado.

El usuario puede ver el estado de sus incidencias desde una pantalla de la aplicación, además recibirá un correo una vez se inicie la tramitación de la incidencia que contendrá el código para cerrar la incidencia y otro a la finalización de esta, el envío de estos correos se ha hecho posible gracias a unos programas que se encuentran en archivos de Google Script.

Las incidencias las puede anular el usuario desde su aplicación ya que en la pantalla que hemos creado para ver el estado de las incidencias también hay un espacio para introducir el código para anular la incidencia.

La plataforma contiene una página web con la que se puede realizar un seguimiento del servicio de las máquinas de café. En esta página web hay tres secciones.

Una de las secciones de la página web contiene una tabla con las incidencias con la que se puede interactuar filtrando los datos en función de la fecha, tipo de incidencia, estado, id de usuario o máquina. En esta sección también nos aparecen las etiquetas que irán en los sobres con el dinero devuelto al usuario una vez solucionado el problema, en ellas nos aparecen el dato de fecha usuario y cantidad de dinero. La tabla y las etiquetas las hemos creado con Google Sheets y Awesome table, hemos filtrado los datos en Google Sheets creando tablas con los datos que necesitábamos y posteriormente las hemos subido a Awesome table que nos permite darles un formato a nuestras tablas y crear filtros que nos ayudan a ver los datos que contienen.

También hemos creado una sección en la página web en la que mostraremos estadísticas que nos interesan para controlar la calidad del servicio que realizan las máquinas de café. Estas estadísticas se mostrarán con gráficos sobre las incidencias más comunes o máquinas más averiadas. En esta sección también mostramos el tiempo medio de resolución de las

incidencias. Estos datos los hemos filtrado en tablas de Google Sheets y posteriormente los hemos subido a la web a la sección de estadísticas.

En esta página web también aparece la sección para anular las incidencias desde la web, esto se ha hecho creando un formulario en uno de los archivos .html que forman la web, este archivo va enlazado con un Script de Google que nos permite enviar los datos del formulario a una tabla de Google Sheets.

Para finalizar hemos subido la aplicación a un repositorio para que se pueda descargar mediante un código QR, este nos redireccionará mediante un enlace al sitio donde se encuentra.

Como vemos se ha conseguido dar la solución que buscábamos al problema que queríamos resolver. Como todo, este trabajo tiene puntos en los que se puede hacer mejoras, por ello para trabajos futuros este trabajo se propone:

- Que la aplicación sea capaz de localizar las máquinas más cercanas, ya que sería más cómodo para el usuario que se localizaran las máquinas que se encuentran más cerca.
- Que la cantidad de dinero que no devuelva la máquina se pudiera recibir mediante la aplicación enlazando una cuenta bancaria.

Capítulo 8

Bibliografía

PASCUAL DEL RIQUELME BENAVENT DE BARBERÁ, Antonio. Bluetooth low energy. 2013. Tesis de Licenciatura. Universitat Oberta de Catalunya.

DIMOULIS, Georgios. Bluetooth vs. WLAN. 2013.

BREY, Antoni; ELIAS, Antoni. El fenómeno Wi-Fi. Infonomia, 2005.

HUIDOBRO, José. Código QR. Revista Bit Digital, 2009, no 172, p. 47-49.

CONCARI, Sonia Beatriz. Tecnologías emergentes ¿cuáles usamos. Lat. Am. J. Phys. Educ. Vol, 2014, vol. 8, no 3, p. 494.

PELÁEZ, David; TIPANTUÑA, Christian. Servidor de comunicaciones unificadas con Raspberry Pi y Micro-Elastix. Maskana, 2016, vol. 65, no Supl.

RAMOS CASTRO, Francisco, et al. Sistema de localización de taxi, basado en Android, PHP y MYSQL. 2012.

ROMANO, José Mariano González. Desarrollo de sitios web con PHP y MySQL. 2011.

GILBERT, Lanny. Cookie management systems and methods. U.S. Patent No 7,379,980, 27 Mayo 2008.

MELCHOR, Eduardo; VELÁSQUEZ, Sergio. SISTEMA DOMÉSTICO CON INTERFAZ ANDROID USANDO RASPBERRY PI. Ciencia e Ingeniería, 2017, vol. 2, no 1, p. 12-12.

GOMES, Tancicleide CS; DE MELO, Jeane CB. App inventor for android: Uma nova possibilidade para o ensino de lógica de programaçao. En Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2013.

KLOSS, Jörg H. Aplicaciones de Android con el inventor de la aplicación: la manera rápida y fácil de crear aplicaciones de Android . Addison-Wesley, 2012.

S. Chacon and B. Straub. Pro Git v. 2.1.78. APress, second edition, jul 2018. Disponible en línea en https://git-scm.com/book/en/v2.

K. Schwaber and J. Sutherland. The Scrum Guide™ – The Definitive Guide to Scrum: The Rules of the Game. Disponible online en http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf, sep 2016. (MarcadorDePosición1)

A. Littlefield. The Beginner's Guide To Scrum And Agile Project Management. Disponible online en https://blog.trello.com/beginners-guide-scrum-and-agile-project-management, sep 2016.

L. Sánchez. Directrices de organización del documento final del trabajo fin de grado tareas científico-técnicas. Disponible en línea en https://campusvirtual.uclm.es/mod/resource/view.php?id=1082293, apr 2013.