Universidad de Cantabria

Facultad
de
Ciencias

HAPI SECURITY: DESARROLLO DE UNA APP MÓVIL PARA COMPARAR LA SEGURIDAD EN DISPOSITIVOS IoT

(Hapi Security: Development of a mobile app to compare the security of IoT devices)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Mario Ingelmo Diana

Director: Carlos Blanco Bueno

Co-Director: Juan Maria Rivas Concepcion

Julio – 2023

Índice

Resumen

La idea de desarrollar esta aplicación nace hablando con el tutor del proyecto, Carlos Blanco, sobre un proyecto otorgado a la universidad **(indicar nombre proyecto)**. Carlos busca crear, en consonancia con el proyecto y unos estándares de la ENISA (Agencia de la Unión Europea para la Ciberseguridad), una aplicación donde poder comparar la seguridad y sostenibilidad de distintos dispositivos IoT que podamos tener en nuestros hogares, de manera que el usuario pueda tomar decisiones basándose en datos que comúnmente son de difícil acceso y así poder saber que tan seguro y sostenible es el producto que ha adquirido o quiere adquirir.

A la vista de la idea inicial y el encanto que tiene para mí el desarrollar una aplicación de este estilo, decido seguir para adelante con la idea, creando así Hapi Security. Hapi Security es una aplicación para dispositivos móviles, desarrollada en Java, que cuenta con diferentes funcionalidades, dispone de filtros y un buscador, de un escáner de códigos de barras, de favoritos para guardar tus dispositivos favoritos y de una sección donde compartir la aplicación con la gente que te rodea. Además de obviamente, tener los datos de los dispositivos de IoT más comunes en diferentes secciones y sus respectivas puntuaciones en seguridad y sostenibilidad.

Para el desarrollo de Hapi Security se ha tenido que dividir el proyecto en dos partes: La primera, donde se ha desarrollado un servicio REST con Spring Boot de donde poder tomar los datos de los dispositivos IoT desde la aplicación. Y la segunda, la propia aplicación desarrollada en Android Studio usando como lenguaje Java y usando el servicio desarrollado anteriormente para obtener los datos de los dispositivos.

Palabras clave:

Aplicación móvil, Dispositivos IoT, Seguridad, Sostenibilidad, Servicio REST, Java, Android Studio.

Abstract

The idea of developing this application arises from a conversation with the project supervisor, Carlos Blanco, regarding a project assigned to the university **(specify project name)**. Carlos aims to create, in line with the project and ENISA standards (European Union Agency for Cybersecurity), an application that allows users to compare the security and sustainability of different IoT devices found in their homes. This way, users can make informed decisions based on data that is often difficult to access, enabling them to determine the level of security and sustainability of a product they have acquired or wish to acquire.

Considering the initial idea and my personal enthusiasm for developing an application of this nature, I decide to proceed with the concept, thus creating Hapi Security. Hapi Security is a Java-based mobile application that offers various functionalities. It includes filters and a search function, a barcode scanner, a favorites feature to save preferred devices, and a section for sharing the application with people in your surroundings. Additionally, it provides comprehensive data on the most common IoT devices in different sections, along with their corresponding security and sustainability ratings.

The development of Hapi Security required splitting the project into two parts: First, a REST service was created using Spring Boot to retrieve IoT device data for the application. Second, the application itself was developed using Android Studio, using Java as the programming language, and utilizing the previously developed service to obtain device data.

Key Words:

Mobile application, IoT devices, Security, Sustainability, REST service, Java, Android Studio.

1. Introducción

Hoy en día vivimos rodeados de dispositivos IoT, desde los asistentes virtuales, pasando por la iluminación y terminando en los electrodomésticos inteligentes entre muchos otros campos, muchas de las cosas que nos rodean disponen de una conexión a Internet y eso supone un riesgo en la seguridad de estos y en tu seguridad.

Prueba de esto es la cantidad de ataques que se detectan a dispositivos de este tipo. *Kaspersky*, conocida compañía internacional en el sector de la ciberseguridad hizo públicos los siguientes datos sobre ataques a sus honeypots (Software que imita un dispositivo IoT vulnerable) en 2021: “En el primer semestre de 2021, el número de intentos de infección totales alcanzó los 1.515.714.259, mientras que durante los seis meses anteriores fueron 639.155.942” [1]. A la vista de estos datos observamos como el aumento de los ataques, solamente en “señuelos” de la empresa *Kaspersky* casi se triplican, lo que nos da una idea general de lo que puede suponer a nivel global donde actualmente hay alrededor de 7 mil millones de dispositivos IoT conectados a la red y se estima un crecimiento hasta los 27 mil millones en 2025 [2].

A la vista de estos datos, podemos observar la gran importancia que tiene la seguridad en los dispositivos IoT, pero lamentablemente, es un aspecto al que poca gente presta atención y cuyos datos son de difícil acceso.

Por todo ello y gracias al otorgue del proyecto ***INSERTAR NOMBRE*** a la universidad, decidí desarrollar Hapi Security, una aplicación móvil donde poder consultar la seguridad de los diferentes dispositivos IoT del mercado, además de la sostenibilidad y las listas con los aspectos tanto positivos como negativos de seguridad y sostenibilidad, de manera que el usuario tenga fácil acceso a los mismos y pueda valorar diferentes opciones a la hora de comprar dispositivos IoT en materia de seguridad y sostenibilidad.

1.1. Objetivo

El objetivo principal es darle al usuario una aplicación móvil donde poder comparar la seguridad y sostenibilidad de diferentes dispositivos IoT ayudándole a la hora de decidir que dispositivo comprar. También que pueda buscar los dispositivos de los que ya dispone, mediante un buscador o escaneando el código de barras del dispositivo y tener una sección donde guardar sus dispositivos favoritos.

Para conseguirlo, se debe desarrollar el proyecto en dos partes: Una parte donde se crea y despliega un servicio donde almacenar y obtener los datos de los dispositivos y otra parte donde desarrollar la aplicación móvil que recoja los datos y se los muestre al usuario. En este documento se recogen estas dos partes, así como los requisitos, el diseño e implementación de estas.

2. Materiales y metodología utilizada

Este apartado recoge tanto la metodología y la planificación del trabajo seguida, como las tecnologías y herramientas utilizadas.

2.1. Metodología

La metodología seguida ha sido la iterativa incremental. Esta metodología consiste en dividir el proyecto en diferentes iteraciones o ciclos. En cada iteración el producto se va actualizando de manera que se desarrolla hasta llegar a un producto final que cumpla con los requisitos y objetivos marcados [3]. Se ha elegido esta metodología porque así las funcionalidades se desarrollan de una en una, dado que en mi opinión, esto beneficia el correcto desarrollo del producto total al pulir cada una de las funcionalidades en la iteración correspondiente y poder ir usando esas implementaciones en el desarrollo de las siguientes a esta.

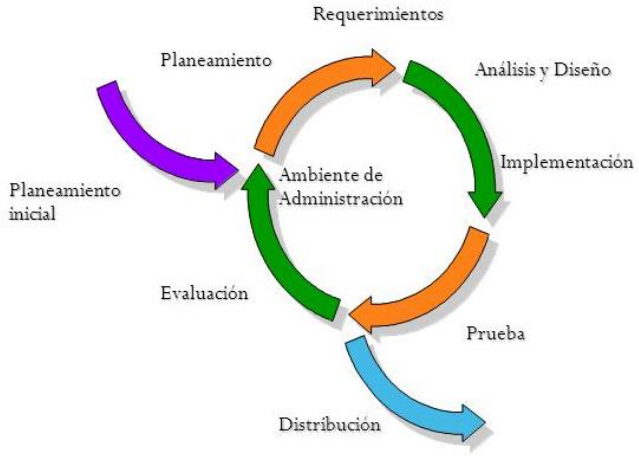

Figura 1. Representación de la metodología iterativa incremental

En este proyecto cada iteración se ha dividido en 4 partes:

- Requisitos.
- Análisis y Diseño.
- Implementación.
- Pruebas.

2.2. Planificación del trabajo

La planificación de este proyecto puede dividirse en tres apartados principales: Preparación previa al desarrollo, desarrollo y despliegue del software y desarrollo de la memoria del trabajo.

2.2.1. Preparación previa al desarrollo

Este apartado es clave puesto que es el inicio de todo. Que este apartado nazca con buen pie es de suma importancia, ya que se determinan los objetivos principales del

proyecto y qué herramientas se van a utilizar para logarlos. Por lo que una buena elección de objetivos y herramientas ayuda en las diferentes etapas del proyecto.

Tras una pequeña reunión con Carlos donde intercambiamos las ideas que teníamos cada uno, pusimos en consonancia los objetivos a desarrollar para este proyecto, llegando rápidamente a un acuerdo y rellenando un documento con estos, para ir revisándolos y ver que todo se completaba adecuadamente.

También fue sencillo el tema de las herramientas, en la misma reunión mencionada anteriormente establecimos que tecnologías utilizar para el desarrollo de las diferentes partes del proyecto. Estas se explicarán más adelante.

2.2.2. Desarrollo y despliegue del software

Este apartado corresponde con lo hablado anteriormente en la metodología. Se ha dividido en iteraciones y cada una de sus partes mencionadas anteriormente, para una vez finalizado todo el trabajo, realizar el despliegue final del servicio y de la aplicación móvil.

Ahora explicaré un poco más en detalle las partes en las que se divide cada iteración:

- **Requisitos:** Se establecen los objetivos a desarrollar en cada iteración del software (requisitos), estos pueden dividirse en dos:
    - **Requisitos funcionales:** Son requisitos sobre las funcionalidades que nuestro sistema deberá implementar y ofrecer a los usuarios, tales como poder buscar, filtrar, escanear, etc…
    - **Requisitos no funcionales:** Son requisitos sobre las diferentes propiedades del sistema, tales como la seguridad, el rendimiento, la mantenibilidad, etc…

- **Análisis y Diseño:** Se analizan los requisitos definidos anteriormente y se plantea una solución sobre como poder implementarlos en consonancia con los ya implementados, esto implica generar o ampliar el diseño para el software (Arquitectura), así como también tomar decisiones en cuanto al tema gráfico en la aplicación (logo, relación de colores, etc).

- **Implementación:** Basándose en el diseño creado en el apartado anterior, este es implementado en el software de manera que añada toda la funcionalidad nueva establecida en los objetivos, actualice funcionalidad ya implementada o realice cambios gráficos en la aplicación.

- **Pruebas:** Tras realizar la implementación del software en el apartado anterior se pasa al testeo de los cambios realizados, ya sea funcionalidad nueva o actualizada, tanto individualmente, como en conjunto. Por ello, se realizan cuatro tipos de pruebas diferentes que explicaremos más adelante: unitarias, integración, interfaz y aceptación.

2.2.3. Desarrollo de la memoria.

La memoria se ha realizado una vez todo el software ha sido realizado, testeado y desplegado. Esta decisión podría haber sido completamente diferente y haberlo hecho en paralelo con el desarrollo, pero en mi caso me decanté por un desarrollo de esta posterior, para así centrar todos mis esfuerzos en el correcto desarrollo del servicio y la aplicación.

2.3. Tecnologías y Herramientas

Las tecnologías y herramientas que se han utilizado son las siguientes:

2.3.1. Git y GitHub

Git es un sistema avanzado de control de versiones (como el “control de cambios” de Microsoft Word) distribuido (Ram 2013; Blischak et al. 2016). Git permite “rastrear” el progreso de un proyecto a lo largo del tiempo ya que hace “capturas” del mismo a medida que evoluciona y los cambios se van registrando. Esto permite ver qué cambios se hicieron, quién los hizo y por qué, e incluso volver a versiones anteriores [4].

Por otra parte GitHub es un servidor de alojamiento en línea o repositorio remoto para albergar proyectos basados en Git que permite la colaboración entre diferentes usuarios o con uno mismo (Perez-Riverol et al. 2016; Galeano 2018). Un repositorio es un directorio donde desarrollar un proyecto que contiene todos los archivos necesarios para el mismo [4].

Figura 2. Logos de Git y GitHub

Se usarán tanto Git con su bash, para ir almacenando los cambios y poder llevar así un control de versiones del proyecto, como GitHub para almacenar el repositorio en la nube. Esto se realiza así por si en un futuro se añadieran más personas al proyecto, facilitar la trazabilidad y el manejo del código y los documentos. Ya que estas tecnologías son mundialmente conocidas y utilizadas.

2.3.2. Spring Boot, Maven y Java

Para el desarrollo del servicio implementado se ha tomado la decisión de utilizar Spring Boot software desarrollado por la empresa Spring y que está disponible para usarse en

Java, Kotlin o Groovy. En mi caso, al ser el lenguaje más dominado Java, se ha utilizado este, además de utilizar Maven a la hora de manejar el empaquetamiento del servicio.

Java Spring Boot (Spring Boot) es una herramienta que acelera y simplifica el desarrollo de microservicios y aplicaciones web con Spring Framework gracias a tres funciones principales:

- Configuración automática
- Un enfoque de configuración obstinado
- La capacidad de crear aplicaciones autónomas

Estas características, combinadas, conforman una herramienta que le permite configurar una aplicación basada en Spring con el mínimo de instalación y configuración [5].

Figura 3. Logo de Spring Boot

Hoy en día, la mayoría de las empresas piden conocimientos sobre cómo implementar microservicios con Spring, por lo que, además de parecerme la opción más cómoda después de valorar varias aprendidas en la asignatura de *Servicios Software*, también me pareció la que más variabilidad podía otorgarme y más proyección a futuro podía tener.

2.3.3. Eclipse

Para el desarrollo del servicio mencionado anteriormente, gracias a la gran cohesión que tiene tanto con Java (es uno de los entornos más utilizados a nivel mundial para el desarrollo de software Java) como con Spring Boot y Maven, se ha decidido utilizar como entorno de desarrollo Eclipse IDE for Enterprise Java and Web Developers.

GeneralController.java

```
1  package es.unican.hapisecurity.REST_TFGMarioIngelmoDiana.controllerLayer;
2
3  import java.net.URI;
32
33 @RestController
34 @RequestMapping("REST_TFGMarioIngelmoDiana")
35 public class GeneralController {
36
37     @Autowired
38     private AuthenticationManager authenticationManager;
39
40     @Autowired
41     private UserDetailsServiceImpl usuarioDetailsService;
42
43     @Autowired
44     private GestionTokens gestion;
45
46     @Autowired
47     private GeneralService servicio;
48
```

Figura 4. Ejemplo de uso de Eclipse en el servicio implementado

2.3.4. Azure

A

2.3.5. Android Studio y Java

A

BIBLIOGRAFIA

[1] https://www.kaspersky.es/about/press-releases/2021_el-numero-de-ataques-a-dispositivos-iot-se-duplica-en-un-ano

[2] https://dplnews.com/numero-de-dispositivos-iot-conectados-alcanzara-22-mil-millones-para-2025/#:~:text=El%20experto%20particip%C3%B3%20en%20el,millones%20de%20dispositivos%20IoT%20conectados

[3] https://proyectosagiles.org/desarrollo-iterativo-incremental/

[4] Astigarraga, J., & Cruz-Alonso, V. (2022). ¡ Se puede entender cómo funcionan Git y GitHub!. Ecosistemas, 31(1), 2332-2332.

[5] https://www.ibm.com/es-es/topics/java-spring-boot#%C2%BFQu%C3%A9+es+Java+Spring+Boot%3F