Universidad de Las Palmas de Gran Canaria
Grado en Ingeniería Informática
Trabajo Fin de Título

Implementación de InputStick en la app Android de Bitwarden para escritura automática de contraseñas

Autor
Pablo Ortigosa Quevedo

Tutor
Antonio Ándres Ócon Carreras

Junio, 2023

Resumen

Modificación de la aplicación Android de Bitwarden, un administrador de contraseñas en la nube, con el fin de añadir soporte para InputStick, un receptor *Universal Serial Bus* (USB) Bluetooth que simula un teclado *Human Interface Device* (HID). Con esto se lograría escribir una contraseña en dispositivos compatibles con el estándar HID mediante USB, que no tienen ningún cliente de Bitwarden instalado y sin necesidad de usar el cliente web. Por tanto se lograría evitar escribir la clave maestra en un dispositivo en el que no se confía y así sólo poner en riesgo la clave que se necesite usar en el momento.

Abstract

Modificacion of the Android aplication of Bitwarden, a password manager on the cloud, with the objective of adding support for InputStick, a Bluetooth *Universal Serial Bus* (USB) receiver that simulates a *Human Interface Device* (HID) keyboard. This would make possible to type a password in devices compatible with the HID standard through USB, that do not have a Bitwarden client installed and without the need to use the web-interface client. This way we would accomplish the feat of avoiding writing the master password in an untrusted device so the only password that could be compromised would be the one needed at the moment.

Índice

Índice de figuras

Índice de tablas

Glosario

freemium De *free* y *premium*. Que ofrece un servicio gratuito además de una opción de pago. 17, 18

bóveda *vault*. Donde Bitwarden almacena los credenciales de las distintas cuentas del usuario. Adicionalmente en este informe se usa indistintamente para referirse también a una cuenta de Bitwarden. 9, 12, 15, 16, 26, 27

cuenta Cuando se hable de cuenta se habla de a lo que un *elemento de cuenta* hace referencia, que principalmente suelen ser cuentas de servicios web. Cuando se refiera a una cuenta de Bitwarden se usará el término bóveda. 12, 15, 23

elemento de cuenta *Login item*: Par de credenciales Usuario-Clave[1]. 7, 9, 23–28, 32, 42, 46

Siglas y acrónimos

2FA *Two-factor Authentication.*

AES *Advanced Encryption Standard.*

API *Application Programming Interface.*

CLI *Command Line Interface.*

E2EE *End-to-end encryption.*

GUI *Graphical User Interface.*

HID *Human Interface Device.*

HMAC *Hash-based Message Authenthication Code.*

JBL *Java Bindings Library.*

MAC *Message Authentication Code.*

MVC *Model-View-Controller.*

MVP *Model-View-Presenter.*

MVVM *Model-View-ViewModel.*

PBKDF *Password-Based Key Derivation Function.*

QMK *Quantum Mechanical Keyboard.*

SHA *Secure Hash Algorithm.*

SRP *Secure Remote Password.*

UML *Unified Modeling Language.*

URI *Uniform Resource Identifier.*

URL *Uniform Resource Locator*.

USB *Universal Serial Bus*.

VPN *Virtual Private Network*.

ZMK *Zephyr Mechanical Keyboard*.

1. Conceptos previos

1.1. Bitwarden

Bitwarden es un administrador de contraseñas código abierto en la nube, donde se guardan los credenciales de las distintas cuentas del usuario. Se accede a la base de datos desde los distintos clientes:

- Escritorio
- Web
- Extensión de navegador
- *Command Line Interface* (CLI)
- **App de Android**
- App de iOS

Figura 1: Logo de Bitwarden. Realizado por Bitwarden.

Una vez en un cliente el usuario puede crear una bóveda donde almacenar los credenciales de inicio de sesión en distintos servicios. Los clientes de Bitwarden almacenan una caché de la bóveda para así poder acceder a ella incluso cuando no pueden conectarse al servidor [2]. Todo esto ocurre bajo un método de conocimiento cero *End-to-end encryption* (E2EE), y los credenciales sólo son descifrados en memoria. Existen distintos tipos de servidores Bitwarden:

- Servidor oficial: Servidor de la propia empresa Bitwarden.
- Self-host: Alojar una estancia manualmente en un servidor propio.

- **Vaultwarden**: Implementación de la *Application Programming Interface* (API) de los servidores de Bitwarden en Rust, *self-host*.

Figura 2: Logo de Vaultwarden. Realizado por Vaultwarden.

Para este proyecto se modificará la app de Android y se usará como servidor una estancia en red local de Vaultwarden.

1.2. InputStick

Actualmente existen 2 modelos InputStick: Bluetooth 2.1 y Bluetooth 4.0.[3] :a versión que usamos en este proyecto es la 4.0 ya que es más reciente, por ello siempre que hablemos de InputStick nos estaremos refiriendo a la versión 4.0.

InputStick es un dispositivo *Universal Serial Bus* (USB) con conexión Bluetooth. Cuando está conectado a un dispositivo mediante el puerto USB, InputStick actúa como un periférico *Human Interface Device* (HID), mientras que a través de la conexión Bluetooth se le indica a InputStick qué hacer, en nuestro caso será simular presionar las teclas de un teclado. 4 [4]

Figura 3: Logo de InputStick. Realizado por InputStick

Figura 4: Diagrama de conexión InputStick. Realizado por InputStick

Debido a su capacidad de dispositivo HID, InputStick puede hacerse pasar como:

- **Teclado**
- Ratón
- Controlador de Videojuegos
- Botones multimedia
- Pantalla táctil

Como medida de seguridad InputStick usa conexión Bluetooth cifrada[5], además es posible activar el protocolo de cifrado *Advanced Encryption Standard* (AES)-128 para enviar datos, y se verificarán usando *Hash-based Message Authenthication Code* (HMAC)-*Secure Hash Algorithm* (SHA)256[6]. Esta clave se puede configurar de 2 formas, ambas desde la app InputStickUtility:

- De forma no segura: La clave se envía a InputStick, por lo que para ello no se puede usar una clave.
- De forma segura: Se le solicita a InputStick que genere una clave, InputStick procederá a escribirla en el dispositivo que se encuentre conectado, ahora el usuario la puede copiar a mano, ya que la puede ver en pantalla.

InputStick nombra a la frecuencia de reportes de escritura como velocidad, así que usaremos la misma terminología, aunque técnicamente sea incorrecto, como al hablar de la velocidad de la memoria RAM, que en realidad es frecuencia.

2. Prólogo

2.1. Motivación

Tras ver a un compañero introducir su clave maestra de Bitwarden en un dispositivo en el cuál no confiábamos surgió la duda y el problema de cómo evitar esta situación. Tras ello discutimos situaciones similares con sus inconvenientes e investigamos superficialmente sobre el tema, llegando así al *abstract* de este proyecto como una solución a dicho problema. Es por tanto el objetivo de este proyecto investigar la proposición, desarrollarla y valorar la viabilidad, la seguridad y la conveniencia de esta. En resumen la pregunta a resolver es: ¿Cómo podemos introducir los credenciales de un servicio de forma cómoda y segura en la mayoría de dispositivos?

2.2. Introducción

En este mundo digital en el que vivimos, parte de los elementos más importantes son los credenciales a las diversas cuentas que pueda tener un usuario. Por ello estos credenciales no deberían repetirse ya que constantemente estos credenciales son expuestos al público tras ataques informáticos[7], sin embargo somos seres humanos, así que desafortunadamente nuestra memoria nos puede fallar, por lo que es surrealista esperar que un usuario recuerde todos los credenciales de sus diversas cuentas. Por ello es recomendable usar un administrador de contraseñas.

Con una bóveda de Bitwarden este problema se soluciona fácilmente. Sin embargo acceder a una cuenta no siempre es tan fácil, en concreto cuando se quiere entrar a una cuenta en un dispositivo desconocido. Existen 2 formas de lograrlo:

- Introducir los credenciales de la cuenta, exponiendo entonces a este dispositivo desconocido la clave maestra de la bóveda
- Buscar los credenciales en la bóveda y enviarlos por Bitwarden Send, o escribirlos a mano.

En el caso de acceder directamente a la bóveda desde el dispositivo objetivo, existen situaciones específicas en las que es imposible o bien poco seguro:

- Cuando la clave es el inicio de sesión del propio dispositivo, la forma de desbloquear el mismo, por tanto, aún no se puede tener acceso a la bóveda.
- Cuando el dispositivo no posee conexión a internet, por ejemplo, acceder a una base de datos interna de una empresa *on-site*.
- Cuando se pone en duda la seguridad del dispositivo. Aunque, por supuesto con *Two-factor Authentication* (2FA) activada tendríamos una gran protección ante la posibilidad de que alguien con intenciones maliciosas conozca nuestra clave maestra, pero no por ello deberíamos ir pregonando dicha clave, como hizo el actual multimillonario Gabe Newell en 2011. [8]
- Cuando el dispositivo objetivo se encuentra bastante limitado, sin permisos de administrador, sin acceso a navegador y/o sin aplicación nativa de Bitwarden. Por ejemplo una consola de videojuegos o una Smart TV.

Por otro lado el servicio Bitwarden Send[9] está planteado para **enviar** datos, información, **claves** y archivos **a otras personas**, pero se sigue pudiendo usar personalmente para enviarse a uno mismo una contraseña, como antaño cuando no era popular el almacenamiento online y uno se enviaba un correo electrónico con un archivo porque se había olvidado el pendrive en casa. Aún así crear un Send es un poco rudimentario, pues requiere muchos pasos y el *Uniform Resource Locator* (URL) es bastante largo como para escribirlo a mano. 5

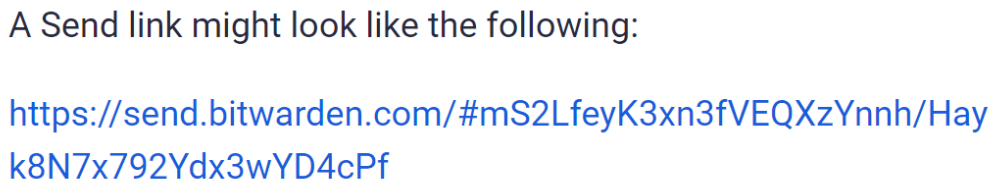


Figura 5: Un URL de Bitwarden Send. Realizado por Bitwarden.

Adicionalmente si la estancia de Bitwarden se encuentra en una red local tras una *Virtual Private Network* (VPN) o similar en vez de en un dominio

público, sólo se podría acceder al enlace si el dispositivo objetivo se encuentra conectado a la misma VPN.

Así llegamos a la solución que en este proyecto hemos implementado. Hoy en día, es muy poco probable no llevar un móvil encima, por ello lo cómodo y simple sería que el móvil escribiese por nosotros los credenciales específicos, evitando así exponer la clave maestra. Para llevar a cabo dicha tarea usaremos InputStick como medio de transmisión.

2.3. Estado del arte

2.3.1. Administradores de contraseñas

- 1Password
  - En la nube
  - Propietario
  - Suscripción de pago
  - Seguridad: [10]
    - *Zero knowledge encryption*
    - *Password-Based Key Derivation Function* (PBKDF)2-HMAC-SHA256
    - *Secure Remote Password* (SRP)
- Bitwarden
  - En la nube
  - Open source
  - Suscripción *freemium* con alternativa *self-host*
  - Seguridad: [11]
    - *Zero knowledge encryption*
    - E2EE AES256
    - *Salted hashing*
    - PBKDF2 SHA256
- KeePass
  - Local

- Open source. KeePassXC es una implementación de KeePass en C++.
- Gratuito
- Seguridad: [12]
  - *Zero knowledge encryption*
  - AES256
  - Chacha20
  - SHA256

- LastPass
  - En la nube
  - Propietario
  - *Freemium*
  - Seguridad: [13]
    - *Zero knowledge encryption*
    - AES256
    - *Salted hashing*
    - PBKDF2 SHA256

- Authorizer [14]
  - Local
  - Open Source
  - Gratuito
  - Seguridad:
  - Autoescritura:
    - Por USB. Requiere una modificación del kernel de Android.
    - Por Bluetooth. El dispositivo objetivo necesita poder establecer una conexión Bluetooth directa.

2.3.2. Dispositivos de escritura automática

- Arduino: Las placas arduino basadas en procesadores 32u4 o SAMD pueden usarse como dispositivo HID [15]. Con esto resuelto el resto podría hacerse fácilmente, con un módulo de pantalla táctil y ser independiente o bien con un módulo Bluetooth y depender de otro dispositivo. Incluso con otro arduino que actúe como *master* y una conexión USB a otro dispositivo este podría dar las órdenes por el puerto serial. Esto de hecho era el primer planteamiento de este proyecto, sin embargo ni hace falta reinventar la rueda, ni parece una alternativa cómoda el llevar una placa encima, comparado con llevar un "pendrive" (InputStick no es un pendrive pero tiene la misma forma y tamaño) en el llavero.

- Firmware: Existen diversos lenguajes de firmware para teclados, con ellos se podría resolver la escritura automática:

  - *Quantum Mechanical Keyboard* (QMK) [16]
  - KMK [17]
  - Vial [18]
  - *Zephyr Mechanical Keyboard* (ZMK) [19]
  - VIA [20]

  La mayoría requieren placas basadas en procesadores 32u4 o SAMD, como las placas arduino.

- InputStick

- Raspberry Pi Zero

- Android:

  - Laa. Requiere Bluetooth o wifi. [21]
  - Android keyboard gadget. Funciona por USB, requiere root ya que modifica el kernel de android. [22]. por usb, requiere root.
  - BLE HID over GATT Profile for Android. Requiere Bluetooth. [23]

3. Proyecto

3.1. Objetivos

Los objetivos de este proyecto son los siguientes:

- Soporte para InputStick en Bitwarden: Se pretende que se puedan escribir los credenciales mediante InputStick a través de Bitwarden de la forma más cómoda posible. Solucionando así el problema de escribir claves complejas en dispositivos que no soportan clientes de administradores de contraseñas. Además debe ser fácil de entender y usar para el usuario menos experimentado y que esta implementación no destaque por encima de ninguna otra funcionalidad de la app, gráfica y programáticamente, que parezca que esta funcionalidad estuvo siempre ahí.
- Escribir código en un programa "grande" y observar los diversos patrones y estilos que se usen en este código profesional. Analizando el mismo y aprendiendo de este.
- Escribir código en una aplicación para dispositivos móviles y observar qué hay que tener en cuenta para ello.

3.2. Competencias

3.2.1. TFG01

«Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.»
La realización de este trabajo cuya idea proviene de un problema que se expone en el día a día, buscando extender las funcionalidades de un programa informático "grande" como es la app de Android de Bitwarden.

3.2.2. CII08

«Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de

programación más adecuados.»
La modificación de código en una aplicación "grande" requiere un estudio previo de dicha aplicación para que la modificación sea coherente y consistente.

3.2.3. CII017

«Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.»
El estudio y modificación de la *Graphical User Interface* (GUI) para añadir funcionalidad nueva.

3.2.4. CII018

«Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.»
La modificación de código abierto implica el estudio de las licencias a las que está sujeto.

3.2.5. IS01

«Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.» El estudio de un código en una aplicación "grande" implica el estudio de patrones de diseño de la ingeniería del software

3.2.6. TI07

«Capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos.»
El estudio de un administrador de contraseñas y la proposición de enviar estas claves de forma segura a InputStick.

3.3. Análisis

Parte del análisis, en concreto los patrones de de diseño, se investigó en paralelo que la implementación, cuando algo resultaba no ser como creíamos había que revisar lo comprendido de analizar el código de la aplicación. Por ello esta sección pretende parte el marco teórico del proyecto aprendido de observar el código de Bitwarden. Mientras que más adelante se discutirá la parte práctica. Por tanto el orden aquí expuesto no representa el orden cronológico del proyecto.

3.3.1. Historias de usuario

- Cernuda es un joven al que le encanta ver a sus creadores de contenido favoritos, como Ibai, AuronPlay o ElRubius. Para verlos usa la plataforma *Twitch*. Cernuda quiere iniciar sesión en su *Smart TV*, pero como usuario de Bitwarden su clave es muy larga y por lo tanto tosca de introducir con el mando a distancia.

- Lorca es un fanático de los videojuegos, recientemente se ha comprado la nueva consola de última generación PlayStation 5. Lorca quiere iniciar sesión en PlayStation Network, pero como usuario de Bitwarden conoce poco más que su clave maestra. Al igual que Cernuda, su clave de un servicio web es demasiado larga y resulta molesto introducirla en el dispositivo.

- Góngora es un estudiante de informática. Entre el estudio para las asignaturas y cursos de la universidad, suele usar ordenadores de la biblioteca y el "Gaming Space ULPGC". Por ello tiene que iniciar sesión con frecuencia en GitHub y otros sitios webs, sin embargo esto requiere muchos pasos, iniciar sesión en Bitwarden, copiar la clave al portapapeles y luego pegarla en el campo para iniciar sesión. Además teme que previamente alguien haya activado el historial del portapapeles de Windows y un día se olvide de comprobarlo.

- Quevedo es un trabajador que frecuenta múltiples oficinas, por lo tanto requiere iniciar sesión con frecuencia en ordenares distintos, al igual que Góngora le resulta que son demasiados pasos los necesarios hasta poder iniciar sesión.

Adicionalmente están estas solicitudes por parte de usuarios de Bitwarden de realizar lo que se plantea en este proyecto:

- https://community.bitwarden.com/t/does-bitwarden-android-support-logging-in-via-bluetooth/9364/3
- https://community.bitwarden.com/t/add-inputstick-api-to-apps-ios-android/3041
- https://community.bitwarden.com/t/inputstick-integration/42596

3.3.2. Modificaciones en la GUI

Para desarrollar una solución es primero necesario hacer un análisis. Como el objetivo es hacer parecer que la funcionalidad siempre estuvo en la aplicación, lo primero será buscar en qué contexto puede ser necesario escribir una clave desde la aplicación mediante InputStick. Empezaremos por lo más trivial, el usuario ya tiene una cuenta creada y quiere escribir la clave de forma inalámbrica, para ello, el usuario navegará hasta el elemento de cuenta [1], donde se encuentra la pantalla de la Figura 6.

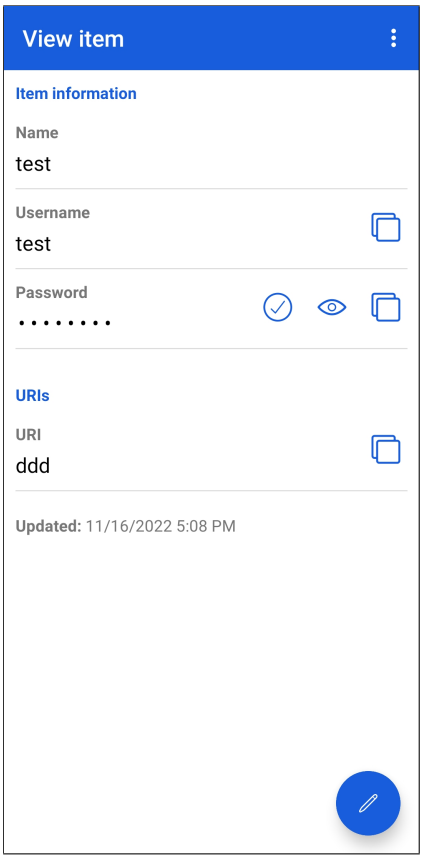


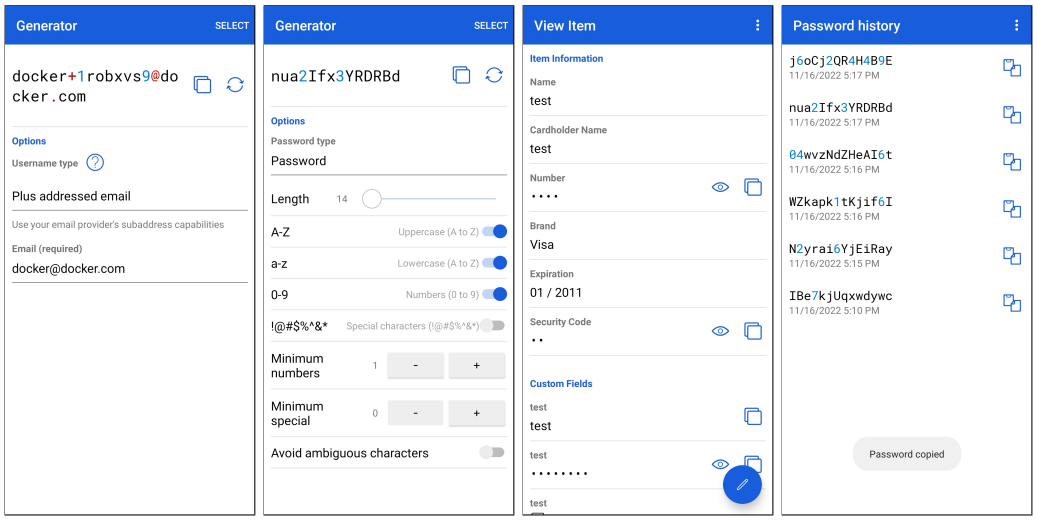
Figura 6: pantalla de un elemento de cuenta. De la app de Bitwarden.

Podemos observar que el campo *Password*, tiene diferentes botones, en orden de izquierda a derecha:

- Comprobar clave: Comprueba si la clave ha sido comprometida mediante la API de *Have I Been Pwned*. [24]
- Alternar visibilidad: Alterna el enmascaramiento de la clave.
- Copiar clave: Copia la clave al portapapeles.

De aquí podemos extraer que cada vez que haya un botón para copiar al portapapeles es un caso de uso donde el usuario podría querer escribir de forma inalámbrica su clave. Como en la imagen no sólo hay un botón para copiar al portapapeles la clave, sino que lo hay también para el usuario y el

Uniform Resource Identifier (URI) podemos entender que sería útil escribir inalámbricamente también estos campos. Por lo que ahora en el proyecto para lograr uno de los objetivos deberemos tener en cuenta dichas situaciones. Pasaremos pues a buscar en todas las pantallas de la aplicación, cuáles tienen botones para copiar al portapapeles. Esto además mejora la comodidad del producto, pues ni siquiera nos habíamos planteado la opción de enviar el URI, lo cual si bien no ahorra mucho tiempo, el poco que ahorra hace que la interacción sea fluida, lo cual, consideramos extremadamente importante.



(a) (b) (c) (d)

Figura 7: Figura múltiple. De la app de Bitwarden.
(a) pantalla de generación de alias.
(b) pantalla de generación de claves.
(c) pantalla de tarjetas, con campos adicionales.
(d) pantalla de historial de claves.

Como podemos ver en la Figura 7, este botón lo podemos encontrar en las pantallas de generación de claves o alias, la vista de tarjeta y el historial de claves, además de la ya mencionada vista de elemento de cuenta. Así que estas serán las partes que modificaremos para ello.

Lo siguiente es buscar dónde añadir la configuración de InputStick, ya que es necesario poder elegir la velocidad de escritura, así como la disposición, pues el protocolo HID no transmite caracteres, si no un código que se corres-

ponde con la ubicación física de la tecla que se ha pulsado, el estándar usa como base una disposición de Estados Unidos, y cualquier otra disposición es en realidad o bien un mapa posición⟷carácter creado por el Sistema Operativo o bien un mapa carácter⟷posición creado por el dispositivo de entrada[25][26]. Como un dispositivo HID no puede conocer este mapa, y el Sistema Operativo tampoco, el estándar es que el Sistema Operativo interpreta esta información, un teclado comunicará los códigos de las teclas, y el usuario debe ajustar en el Sistema Operativo cuál debiese ser esta, por lo general esto se hace automáticamente al configurar el idioma[27]. Por ello es crucial implementar una forma de cambiar este ajuste, ya que si no se hace, habría que cambiar en el Sistema Operativo la disposición cada vez que se quiera introducir una clave con InputStick y muchas contraseñas se escribirían incorrectamente si no se cambia, pues aunque gran parte del abecedario y los números coinciden entre diferentes disposiciones, el resto de caracteres no suelen coincidir. Lo podemos ver en el siguiente ejemplo interactivo:

- Disposición de España.
- Disposición internacional de Estados Unidos.

Si InputStick está configurado como Estados Unidos internacional, y un dispositivo Windows está configurado como España, al enviar "#" se recibirá "·". Aunque si el lector se anima puede probarlo usted mismo al revés, asumiendo que su teclado tiene la disposición de España. Para cambiar la configuración de la disposición en Windows 10 debe dirigirse a la pantalla de "Configuración", en la opción "Hora e idioma",seleccione "Idioma" y ahora seleccione Inglés de Estados Unidos, si sólo puede seleccionar España tendrá que agregar un "Idioma Preferido" más abajo en la misma pantalla. Con esto al presionar "Shift" y "3", que es equivalente a "·", recibirá en su lugar "#". No olvide cambiar de nuevo su disposición a España, y eliminar el idioma que ha añadido, de lo contrario en la barra de herramientas, a la derecha, se le mostrará la disposición actual y podría cambiarlo sin querer.

En la pestaña de ajustes (Figura 8), podemos ver *Auto-fill services* como un grupo independiente. Esta funcionalidad permite auto rellenar campos de inicio de sesión en navegador y en apps del dispositivo mediante un *pop-up*, tras pulsarlo se abre la bóveda y el usuario puede elegir el elemento de cuenta que quiera.

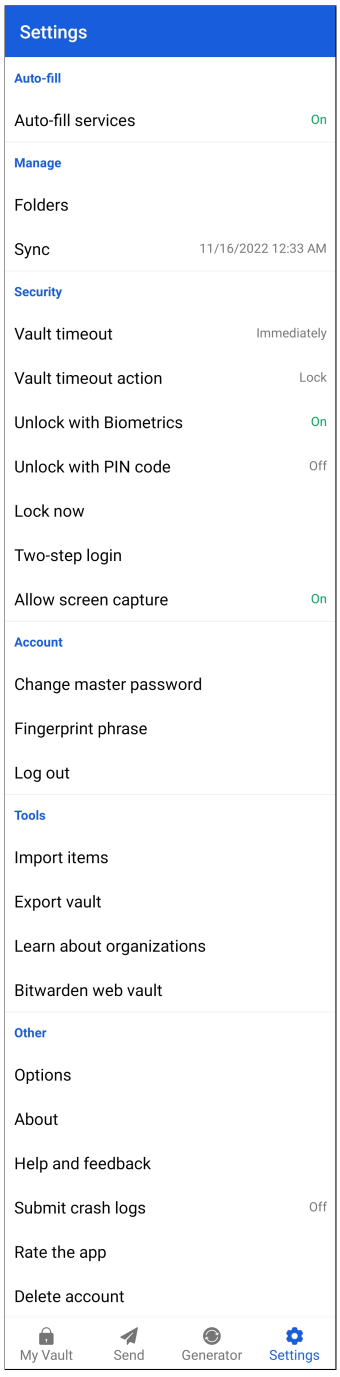


Figura 8: Pantalla de configuración. De la app de Bitwarden.

La funcionalidad que aporta InputStick es también un método de inicio de sesión, por lo que podríamos usar la misma estructura para este proyecto y poner la configuración en la pantalla de ajustes, debajo del servicio de auto rellenado.

También podemos encontrar en la pantalla de Opciones (Figura 9), a la que se accede desde los ajustes, que los servicios de auto rellenado tienen también aquí parte de su configuración. Esta configuración define en qué URIs mostrar el *pop-up* del servicio de auto rellenar, también define si registrar como un elemento de cuenta nuevas entradas. Sin embargo la naturaleza de InputStick es distinta y no es totalmente automático, no existe esta comunicación bidireccional, siempre es unidireccional por el comportamiento del protocolo HID mencionado anteriormente, el proceso de inicio de sesión seguirá siempre este orden:

1. Acceder al sitio donde se pretende introducir los credenciales.
2. Poner el enfoque en el campo de texto que se quiera rellenar-
3. En el móvil abrir la bóveda.
4. En la bóveda buscar el elemento de cuenta.

5. En el elemento de cuenta pulsar el botón de enviar.

Es por ello por lo que no vamos a añadir ningún campo de configuración en esta pantalla, si no que crearemos una pantalla por separado y se accederá directamente desde la pantalla de Ajustes, teniendo así toda la configuración en la misma ubicación.

Adicionalmente, deberemos también añadir en esa pantalla una forma de activar o desactivar el servicio, lo cual activará o desactivará los botones para enviar a InputStick.

En resumen, los objetivos que se plantean son:

- Múltiples botones para enviar a InputStick.
- pantalla de ajustes con:
  - Interruptor para activar o desactivar la funcionalidad.
  - Selector de disposición.
  - Selector de velocidad de escritura automática.

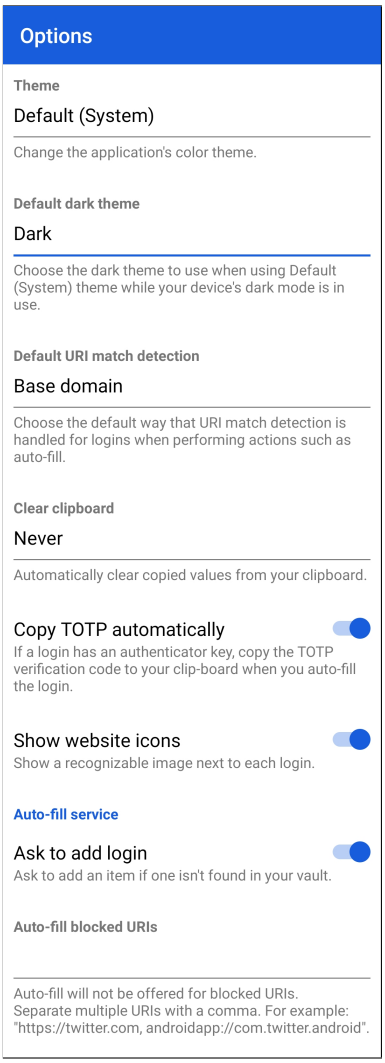


Figura 9: pantalla de opciones. De la app de Bitwarden.

3.3.3. Casos de uso

Antes de comenzar con el código, es necesario analizar los casos de uso, en la figura 10 podemos ver un esquema *Unified Modeling Language* (UML) explicándolos. De forma general casos son enviar campos, elementos del generador y configurar los ajustes.

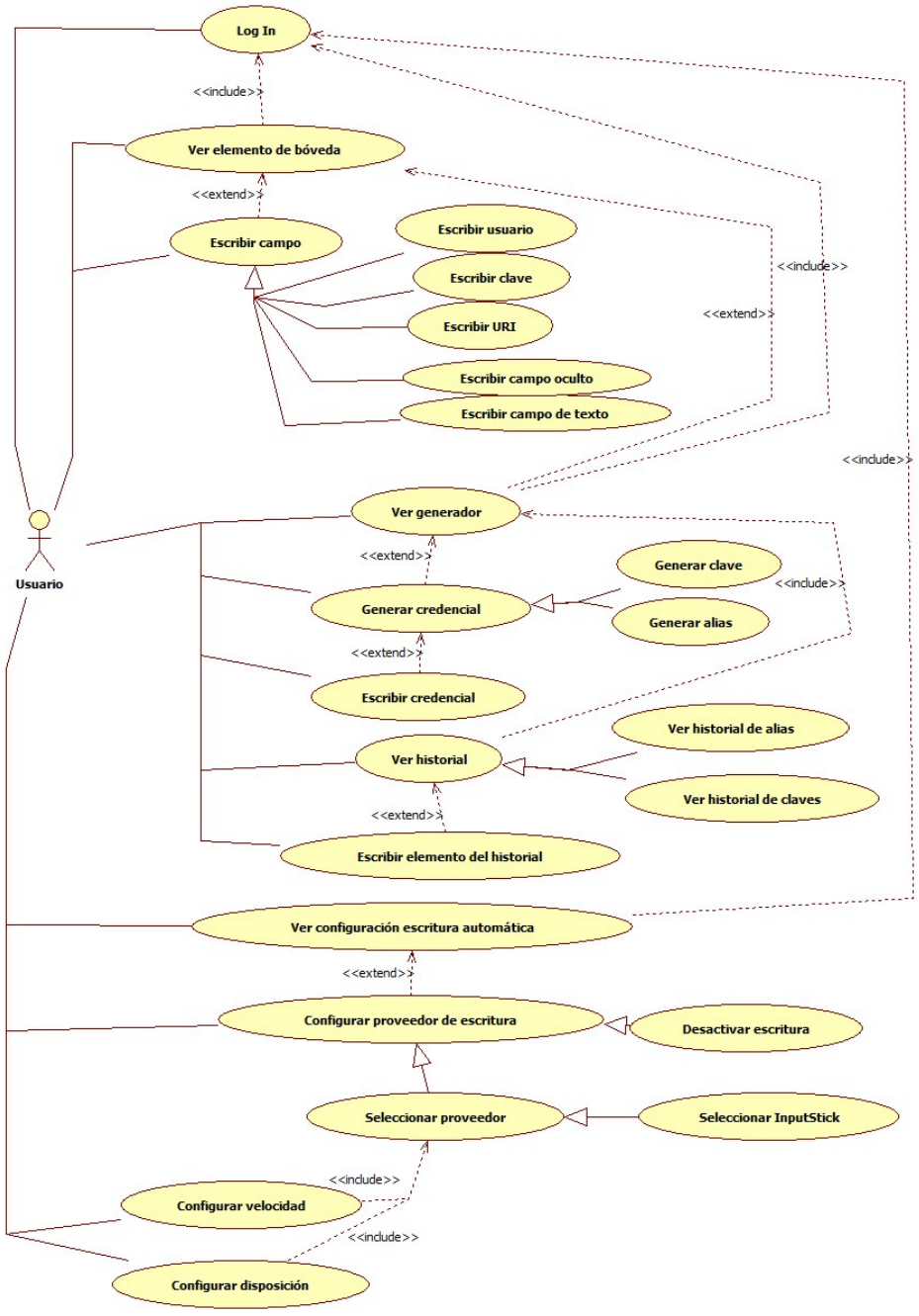


Figura 10: Casos de uso. Realización propia

3.3.4. Esquema

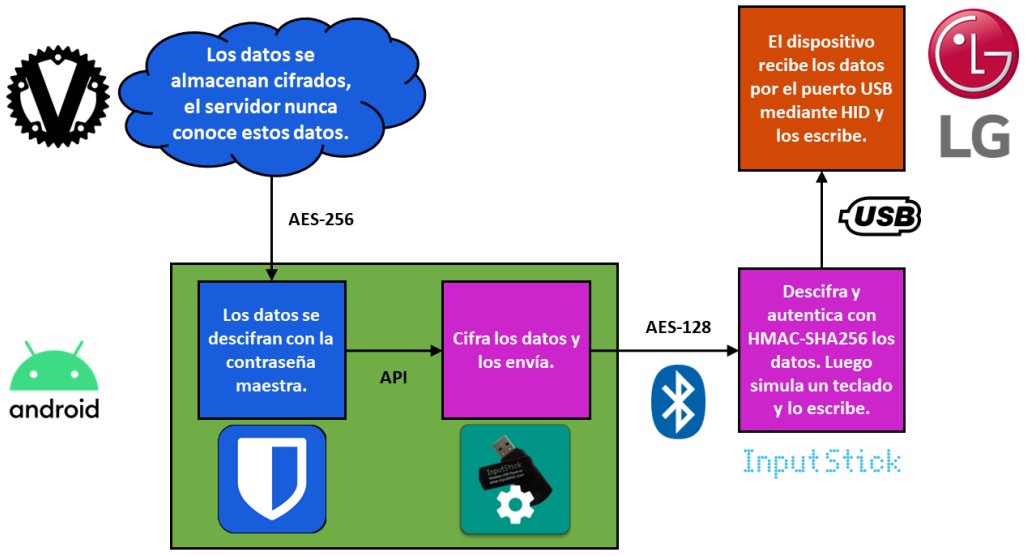


Figura 11: Diagrama de conexiones teórico. Realización propia.
El logo de LG representa una *Smart TV*, a modo de ejemplo, pero es sustituible por cualquier dispositivo compatible con USB HID.

La figura 11 ilustra el mapa de conexiones. El servidor de Bitwarden (en nuestro caso Vaultwarden) almacena los datos cifrados, el servidor desconoce por completo la clave para descifrarlos. Estos datos sólo son descifrados en los clientes los cuales le solicitan los datos a Bitwarden. Luego, cuando el usuario decide enviar una clave a InputStick esta se volverá a cifrar para su transmisión por Bluetooth, este proceso lo lleva la app InputStickUtility, a la cual se le transmite la información mediante la API InputStickBroadcast[1]. Cuando InputStick recibe los datos a escribir los descifra y los envía al dispositivo al que está conectado, usando el protocolo HID por USB. Finalmente el dispositivo lo escribe en el campo que tenga el enfoque. Se puede observar claramente como los **nunca** entran o abandonan Android **sin cifrar**, única-

[1]El nombre pueda dar a entender que los datos se envían a todas las apps, no es así. La llamada a type llama a su vez a send, aquí podemos ver que esto se envía específicamente a la app InputStickUtility.

mente cuando se envía la información a InputStickUtility. En la figura 12 se muestra como quedaría el resultado final de una forma más práctica y menos esquemática.

Figura 12: Diagrama de conexiones práctico. Realización propia.
El ordenador sobremesa es a modo de ejemplo, pero es sustituible por cualquier dispositivo compatible con USB HID.
Marco de móvil. Imagen por brgfx en Freepik.com.
Servidor. Imagen por macrovector en Freepik.com.
Marco de monitor. Imagen por d3images en Freepik.com.

3.3.5. Model-View-ViewModel

La app está diseñada con el patrón arquitectónico *Model-View-ViewModel* (MVVM). Como los patrones *Model-View-Controller* (MVC) y *Model-View-Presenter* (MVP), este patrón tiene como objetivo abstraer la vista de la lógica. Este patrón pretende solventar problemas como el grado de acoplamiento que existe en los otros patrones, en MVC existe poca independencia entre sus componentes, mientras que en MVP ocurre con la vista y el presentador. Por ello la vista modelo desconoce la existencia de la vista y el modelo desconoce la existencia de la vista modelo. En este caso, la vista y la vista modelo están enlazadas mediante los campos de datos que son estados que indican a la vista cómo debe presentarse al usuario. Por otro lado la vista modelo, como intermediaria, mantiene estos campos actualizados recogiendo

la información necesaria del modelo, el cual se ocupa de acceder a la base de datos.[28][29]

En esta app, el uso de MVVM hace que sea bastante fácil y cómodo desarrollarla para distintos Sistemas Operativos como es el caso de Android e iOS, y con la ayuda de Xamarin, esto resulta más claro, ya que ambas versiones comparten el código de la vista, y la vista modelo. Aunque en este proyecto nos centraremos sólo en Android.

3.3.6. Command

Para que la vista pueda enviar información a la vista modelo se usa el patrón de diseño command, de esta forma separan el comportamiento de un botón de la vista. Esto se logra convirtiendo solicitudes en objetos, cada objeto es una subclase de la interfaz command[30][31]. Todos los botones tienen una referencia en la vista al comando, el cual se encuentra declarado en la vista modelo. Curiosamente el ejemplo que se usa en libro que se cita está relacionado con el portapapeles (pegar desde el portapapeles), mientras que descubrimos que se usa el patrón command al analizar los botones anteriormente mencionados de copiar al portapapeles.

3.3.7. Factory method

Se usa el patrón de diseño factory method para crear diferentes versiones de campos personalizados de un elemento de cuenta, estos pueden ser booleanos, ocultos, de texto o relaciones. Asumimos que el objetivo es centralizar la creación de estos objetos, pues no se aprovecha el principal beneficio de este patrón, que es evitar especificar la subclase que usar, ya que se le pasa por parámetro un enumerado especificando que subclase se quiere[30][31].

3.3.8. Servicios

La aplicación tiene un sistema de registro de servicios para fácil acceso de distintas funcionalidades en el resto de la aplicación. Se trata de un contenedor de servicios, este es una clase estática, por lo que se puede solicitar acceso a estos servicios desde cualquier parte de la aplicación. No hemos podido encontrar que patrón podría ser este, pero sospechamos que está relacionado con evitar la creación y borrado continuo de objetos.

3.3.9. *Secure Hash Algorithm* (SHA)

En concreto SHA-256, del grupo SHA-2 es una función criptográfica de hash, que se usa para verificar la integridad de un mensaje o archivo y por tanto asegurar que no ha sido alterado. Es el estándar actual debido a su eficiencia y baja posibilidad de colisiones, que se refiere a cuando 2 mensajes distintos devuelven el mismo resultado tras aplicar la función.

3.3.10. *Hash-based Message Authenthication Code* (HMAC)

HMAC usa como base una función criptográfica de hash, por ejemplo SHA y lo combina con una clave simétrica. Se usa para verificar la integridad de un mensaje y por tanto asegurar que no ha sido alterado.

3.3.11. *Advanced Encryption Standard* (AES)

Es un algoritmo simétrico de cifrado en bloque que tras diversas operaciones como sustitución y permutación cifran el mensaje o archivo. Es el estándar actual debido a su eficiencia y resistencia.

3.4. Desarrollo

3.4.1. Metodología

Este proyecto se ha llevado a cabo mediante la metodología SCRUM. Por lo que semanalmente se ha procurado tener un producto entregable, iterando en este y mejorándolo. Esto se verá reflejado en el orden de realización de tareas. Para organizar dichas tareas se ha usado Obsidian con el plugin Kanban, así se mantiene el orden de las mismas, qué hay que hacer y qué queda por hacer.

3.4.2. Librería de InputStick

Para la comunicación con InputStick es necesario importar la librería de InputStick[32] al proyecto. Como está escrita en Java y la aplicación de Bitwarden está escrita en Xamarin (C#) habrá que usar un intermediaro para realizar la comunicación, para esto usaremos *Java Bindings Library*

(JBL)[33] en VisualStudio, que nos permitirá importar el .jar que es la librería de InputStick.[2]

3.4.3. Código

En esta sección hablaremos del código más relevante, hemos dejado fuera algunas partes de las implementaciones de las interfaces de las que se habla, así como la parte Vista y Vista-Modelo, sólo se discutirá el Modelo, por supuesto el resto del código se encuentra en el repositorio. 3.4.5

El código se ha escrito desde la base de que InputStick es un proveedor de escritura automática, pero pueden haber varios. Por ello, aunque actualmente sólo haya uno, en los ajustes hay un selector del proveedor, en vez de un botón para alternar activado y desactivado. Al proveedor se le debe proporcionar toda la información necesaria cada vez que se quiere escribir, este no almacena nada, excepto lo necesario para realizar conexiones, que es específico al proveedor. Podemos verlo en su interfaz:

```
6  namespace Bit.Core.Abstractions
7  {
8      public interface IAutoTyperProvider
9      {
10         void Connect();
11         void Disconnect();
12         void Type(string text, LayoutType layout, SpeedType
       speed);
13     }
14 }
```

src/Core/Abstractions/IAutoTyperProvider.cs

Así que hemos creado un servicio AutoTyper, este servicio se ocupa de guardar y cargar datos persistentes, aunque realmente es un intermediario, así puede interceptar la información, y validarla. Esto lo hacemos por si un proveedor ya no es compatible con una disposición o se elimina el soporte para un proveedor. Si esta información es errónea o nunca ha sido configurada por el usuario se le otorga un valor por defecto.

```
7  namespace Bit.Core.Abstractions
8  {
9      public interface IAutoTyperService
10     {
```

[2] Bindings library 1, Bindings library 2

```
11          IAutoTyperWrapper GetTyperWrapper();
12
13          Task InitAsync();
14
15          Task<AutoTyperProviderType> GetProviderTypeAsync();
16          Task SetProviderAsync(AutoTyperProviderType type);
17          Task<LayoutType> GetLayoutAsync(AutoTyperProviderType
     type);
18          Task SetLayoutAsync(LayoutType layout,
    AutoTyperProviderType type);
19          Task<SpeedType> GetSpeedAsync(AutoTyperProviderType
    type);
20          Task SetSpeedAsync(SpeedType speed,
    AutoTyperProviderType type);
21
22          List<AutoTyperProviderType> GetCompatibleProviders();
23          List<LayoutType> GetCompatibleLayouts(
    AutoTyperProviderType type);
24          List<SpeedType> GetCompatibleSpeeds(
    AutoTyperProviderType type);
25
26          IAutoTyperProvider CreateTyper(AutoTyperProviderType?
     type);
27          AutoTyperProviderType ProviderType(IAutoTyperProvider
    ? provider);
28      }
29  }
```

src/Core/Abstractions/IAutoTyperService.cs

Aquí el proveedor de auto escritura se asegura de devolver datos válidos:

```
32  ...
33      public async Task<AutoTyperProviderType>
    GetProviderTypeAsync()
34      {
35          var provider = (AutoTyperProviderType?) await
    _stateService.GetAutoTyperProviderAsync();
36          return provider ?? AutoTyperProviderType.None;
37      }
38  ...
```

src/Android/Services/AutoTypers/AutoTyperService.cs

El servicio de estados es el que realmente se ocupa de guardar los datos, los cuales son locales, pero individuales a cada cuenta en el dispositivo, este al recoger la información se asegura que esté definida, si no lo está devuelve

nulo y se le relega solucionar el problema a quien realiza la llamada, que siempre es nuestro servicio intermediaro AutoTyper.

```
1318 ...
1319     public async Task<int?> GetAutoTyperProviderAsync(string
     userId = null)
1320     {
1321         var reconciledOptions = ReconcileOptions(new
     StorageOptions { UserId = userId },
1322             await GetDefaultStorageOptionsAsync());
1323         var key = Constants.AutoTyperProviderKey(
     reconciledOptions.UserId);
1324         int? provider = await GetValueAsync<int?>(key,
     reconciledOptions);
1325         // Check if provider is valid
1326         if (provider != null && !Enum.IsDefined(typeof(
     AutoTyperProviderType), (byte)provider))
1327         {
1328             provider = null;
1329         }
1330         return provider;
1331     }
1332 ...
```

src/Core/Services/StateService.cs

Adicionalmente al recoger las disposiciones y velocidades siempre se debe devolver un valor concreto, no existe “Ninguna disposición” ni “Ninguna velocidad”. Por ello el servicio de auto escritura se solicita a sí mismo las opciones compatibles y contrasta el valor por defecto con esta lista, ya que un proveedor puede no ser compatible con ese valor, si no lo es elegimos un valor cualquiera, en este caso el primer elemento de la lista.

```
107 ...
108         // Helpers
109 
110         /**
111          * Checks if the element is in the list
112          * If not, it defaults to def
113          * If def is not in the list, it defaults to the
     first element of the list
114          */
115         private static T Validate<T>(T element, List<T> list,
      T def)
116         {
```

```
117             return FindOrDefault(FindOrDefault(element, list,
    def), list, list[0]);
118         }
119
120         /**
121          * Checks if the element is in the list
122          * If not, it defaults to def
123          */
124         private static T FindOrDefault<T>(T element, List<T>
    list, T def)
125         {
126             return list.Contains(element) ? element : def;
127         }
128     }
129 }
```

src/Android/Services/AutoTypers/AutoTyperService.cs

Al servicio de auto escritura se le puede solicitar un contenedor de proveedor, esta es la única forma de contactar con el proveedor, así forzamos a que cuando se vaya a enviar algo para escribir al proveedor los datos ya hayan sido inicializados (proveedor, distribución y velocidad). Y aunque pueden haber varias instancias de la implementación del contenedor, el servicio devolverá siempre la misma instancia cuando se le solicite, de forma similar al patrón singleton, ya que el contenedor actua como una extensión del servicio.

```
3  namespace Bit.Core.Abstractions
4  {
5      public interface IAutoTyperWrapper
6      {
7          Task LoadAsync();
8          void Connect();
9          void Disconnect();
10         void Type(string text);
11         bool IsEnabled();
12     }
13 }
```

src/Core/Abstractions/IAutoTyperWrapper.cs

En nuestro caso hemos usado la librería de InputStick mediante llamadas a la app de InputStickUtility, que se ocupa de mandar los datos a InputStick. Lamentablemente no es lo que se pretendía, pero la falta de tiempo nos ha impedido usar la API de InputStick en mayor profundidad. Dicha API se puede usar de dos formas:

- InputStickBroadcast: Al usar esta parte de la API es mucho más simple contactar con InputStick, ya que únicamente es necesario mandar a InputStick qué tiene que escribir, del resto se ocupa InputStickUtility.
- Completa: Al usar la API en su totalidad se puede evitar la instalación de InputStickUtility, pero es requiere una implementación más compleja ya que las facilidades que otorgaba InputStickUtility ya no están presentes, por ejemplo es necesario saber la dirección *Media Access Control*(MAC)[3] de InputStick.

```
9  namespace Bit.Droid.Services.AutoTypers
10 {
11     public class InputStickBroadcastAndroid :
       IAutoTyperProvider
12     {
13         public void Connect()
14         {
15             InputStickBroadcast.RequestConnection(Application
       .Context);
16         }
17
18         public void Disconnect()
19         {
20             InputStickBroadcast.ReleaseConnection(Application
       .Context);
21         }
22
23         public void Type(String text, LayoutType layout,
       SpeedType speed)
24         {
25             InputStickBroadcast.Type(Application.Context,
26                 text,
27                 _layouts.ContainsKey(layout) ? _layouts[
       layout] : _layouts[0], // Should not happen, this is a
       fallback
28                 SpeedToValue(speed));
29         }
30 ...
```

src/Android/Services/AutoTypers/InputStickBroadcastAndroid.cs

[3]Esta es la única vez en este documento que usaremos el acrónimo MAC con un significado distinto a *Message Authentication Code*

Para administrar la selección de distribución se usa un enum, este enum contiene todas las disposiciones listadas por InputStick: https://github.com/inputstick/InputStickAPI-Android#keyboard-layouts

```
namespace Bit.Core.Enums
{
    public enum LayoutType
    {
        [LocalizableEnum("AutoTyperLayoutCSCZ")]
        cs_CZ = 0,
        [LocalizableEnum("AutoTyperLayoutDADK")]
        da_DK = 1,
...
```

src/Core/Enums/LayoutType.cs

Igualmente para administrar la velocidad de escritura se usa otro enum, este enum es una ampliación de las posibilidades que ofrece InputStick, ya que las apps oficiales de InputStick usan todas las velocidades listadas excepto "Fast" y "Faster". De cara al usuario se usan estos términos y no las velocidades reales dos motivos:

- Para no abrumar al usuario: Es innecesario que el usuario sepa las velocidades concretas, mostrar tantos números, en una escala que realmente a las velocidades más altas una persona no es capaz de cuantizar la diferencia, solo lograría sobreestimularlo.
- Porque realmente no es consistente: Estas velocidades se han extraído de las apps oficiales de InputStick creando un pequeño script de python en el que medir las velocidades, pues el creador no dice en ningún lugar las velocidades ya que dependen de la carga de trabajo del dispositivo en el que esté conectado. Si el dispositivo está muy sobrecargado la máxima velocidad no funcionará correctamente y el dispositivo no leerá todas las teclas y por tanto se saltará alguna. Por tanto las velocidades descritas son aproximaciones bajo poca carga.

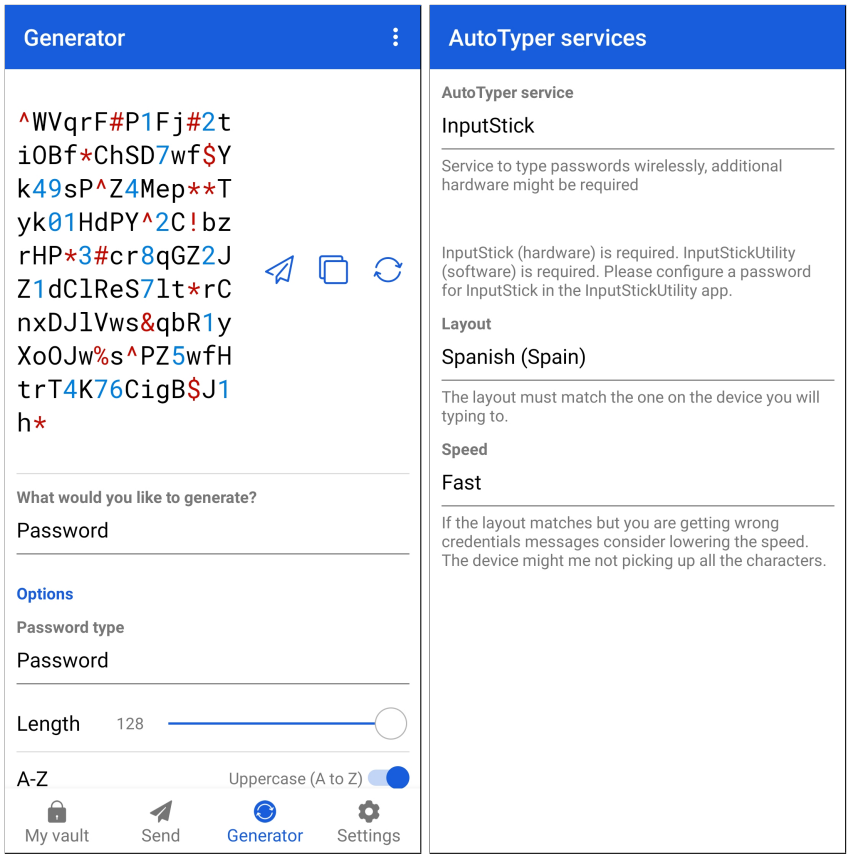
```
namespace Bit.Core.Enums
{
    public enum SpeedType : int          // keys per second (
   aproximate) - ratio
    {
        [LocalizableEnum("Slowest")]     // 8 - 0.1
        Slowest = 0,
```

```
9           [LocalizableEnum("Slower")]     // 16 - 0.2
10          Slower = 2,
11          [LocalizableEnum("Slow")]       // 40 - 0.5
12          Slow = 3,
13          [LocalizableEnum("Normal")]     // 80 - 1
14          Normal = 5,
15          [LocalizableEnum("Fast")]       // 120 - 1.5
16          Fast = 7,
17          [LocalizableEnum("Faster")]     // 180 - 2.25
18          Faster = 6,
19          [LocalizableEnum("Fastest")]    // 240 - 3
20          Fastest = 10,
21      }
22  }
```

src/Core/Enums/SpeedType.cs

3.4.4. Cambios visuales

En la figura 13 podemos ver los cambios hechos en la GUI. En cada campo que hubiese un botón de copiar al portapapeles se ha añadido el botón de enviar al proveedor de escritura automática, este campo sólo es visible cuando se ha seleccionado un proveedor. También se ha añadido una pantalla de configuración de esta funcionalidad, con selectores de proveedor, idioma y velocidad. Se puede acceder a la pantalla de los ajustes del proveedor de escritura automática desde la pantalla de ajustes.



(a) (b)

Figura 13: Figura múltiple. Modificación de la app de Bitwarden.
(a) Pantalla de generación con el botón de enviar.
(b) Pantalla de ajustes del proveedor de escritura automática.

En la figura 14 podemos ver todos los cambios realizados en la pantalla de un elemento de cuenta. El botón de enviar se encuentra en varios campos, hay que destacar que si un campo se encuentra vacío no hay nada que enviar, por lo que no se mostraría. Tampoco se mostraría si el proveedor de escritura automática se encuentra desactivado.

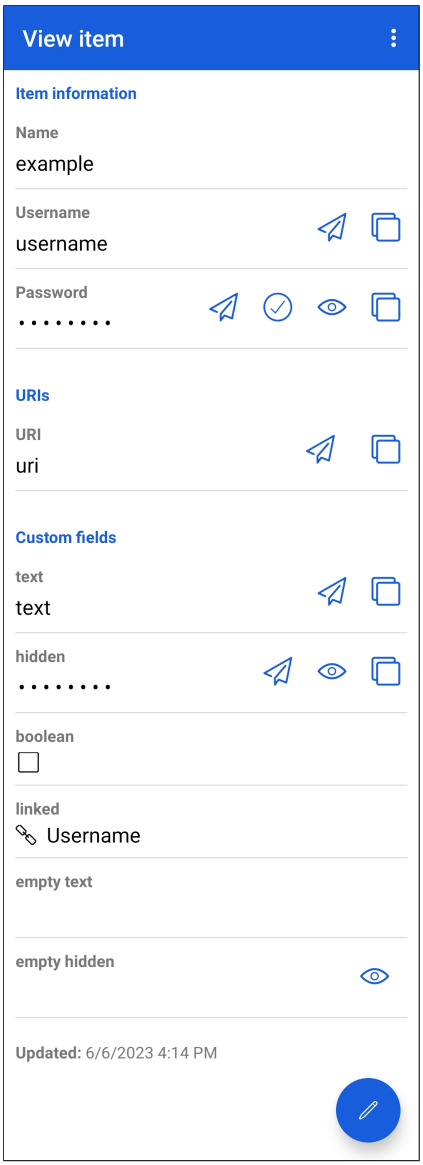


Figura 14: Pantalla de elemento. Modificación de la app de Bitwarden

3.4.5. Repositorios

Los cambios realizados se encuentran en el fork de GitHub de PabloOQ. Debido a un problema de compatibilidad durante el desarrollo hubo que hacer un merge de la rama master de Bitwarden, todo el código se encuentra en:
https://github.com/PabloOQ/mobile/tree/stick-after-merge
Sin embargo, para visualizar rápidamente el código modificado antes de hacer el merge, se podría ver en esta rama (aunque este está también en la rama mencionada anteriormente):
https://github.com/PabloOQ/mobile/tree/stick
Finalmente en este enlace se muestra el código añadido, al ser el total no incluye el código previo a las refactorizaciones.
https://github.com/PabloOQ/mobile/compare/tfg-diff...PabloOQ:mobile:stick-after-merge
El informe escrito en LaTeX se encuentra en:
https://github.com/PabloOQ/informeTFG

3.4.6. Plan de trabajo

En la tabla 1 se muestra el plan de trabajo original.

Fases	Duración Estimada	Tareas
Estudio previo / Análisis	85	Tarea 1.1: Estudio de la aplicación de Bitwarden Tarea 1.2: Estudio de la API de InputStick
Diseño / Desarrollo / Implementación	85	Tarea 2.1: Diseño y creación de la interfaz de usuario Tarea 2.2: Adaptar la API de InputStick a C# Tarea 2.3: Implementación de InputStick a Bitwarden
Evaluación / Validación / Prueba	30	Tarea 3.1: Verificación de la GUI Tarea 3.2: Verificación de la funcionalidad nueva
Documentación / Presentación	100	Tarea 4.1: Documentación de la GUI Tarea 4.2: Documentación de la funcionalidad nueva Tarea 4.3: Documentación de temas relacionados

Tabla 1: Plan de trabajo original. Realización propia.

Sin embargo realmente las horas dedicadas son las que se muestran en la tabla 2. Esto cambio se debe a que el estudio de la aplicación fue mucho más complejo de lo esperado y por tanto requirió mucho más tiempo.

Fases	Duración Estimada	Tareas
Estudio previo / Análisis	115	Tarea 1.1: Estudio de la aplicación de Bitwarden Tarea 1.2: Estudio de la API de InputStick
Diseño / Desarrollo / Implementación	85	Tarea 2.1: Diseño y creación de la interfaz de usuario Tarea 2.2: Adaptar la API de InputStick a C# Tarea 2.3: Implementación de InputStick a Bitwarden
Evaluación / Validación / Prueba	15	Tarea 3.1: Verificación de la GUI Tarea 3.2: Verificación de la funcionalidad nueva
Documentación / Presentación	85	Tarea 4.1: Documentación de la GUI Tarea 4.2: Documentación de la funcionalidad nueva Tarea 4.3: Documentación de temas relacionados

Tabla 2: Plan de trabajo real. Realización propia.

3.5. Pruebas con usuarios

3.5.1. Usuario de Bitwarden

Se ha usado como usuario de prueba al mismo que originó la idea del proyecto. La opinión de dicho usuario es que es un poco complicado configurarlo, ya que no hay ningún aviso cuando la app InputStickUtility no está instalada, esto se suponía que debía ocurrir, si al intentar comunicarse con InputStickUtility esta no se encuentra instalada, la propia librería genera una ventana emergente, sin embargo cuando dicho usuario lo probó, el diálogo no apareció, nuestra teoría es que está relacionado con JBL, por lo que es algo a solucionar. El usuario plantea un tutorial de ventanas emergentes, lamentablemente esto es algo que estaría fuera de lugar en la app de Bitwarden, pues sería la única instancia donde encontrar dicho tutorial. El otro inconveniente que encontró era el hecho de tener en cuenta la distribución del teclado,

pues es algo que realmente el usuario promedio no sabe que funciona así, y tampoco es fácil de comprender, lamentablemente es inevitable el tener que configurar la opción del teclado, y recae en el usuario, la única forma de solucionar esto sería explicándolo mejor. Creemos que actualmente este sistema innecesariamente complejo, y que su uso a día de hoy se debe simplemente a que es algo que se lleva usando mucho tiempo y cambiar a algo mejor sería una tarea en la que las compañías y grupos más grandes se tendrían que poner de acuerdo. Además de esto el usuario indicó que no se fijó en los nuevos botones porque asumía que la opción estaría en el menú desplegable, esto tiene fácil solución, indicándolo en la descripción de la configuración.

3.5.2. Usuario de InputStick

Lamentablemente, no es fácil encontrar a un usuario que conozca InputStick de antemano, así que se ha seleccionado un usuario y se le ha explicado el funcionamiento de InputStick. Tras ello y simplemente diciéndole el objetivo de la app de Bitwarden se le ha dejado navegar por la app modificada a ver si es capaz de desenvolverse. El único problema que encontró era activar la nueva funcionalidad, pues no sabía dónde descartó la configuración como pantalla para activar la funcionalidad, así que hemos intervenido, pero de resto no ha tenido mayor problema, con los textos en la pantalla de configuración lo ha comprendido fácilmente.

3.6. Manual de usuario

Para usar InputStick en Bitwarden el usuario deberá instalar la versión modificada de Bitwarden realizada en este proyecto, también deberá realizar la instalación de InputStickUtility. Una vez hecho esto, deberá iniciar sesión o registrarse en Bitwarden o Vaultwarden, así como configurar la clave de InputStick en InputStickUtility. Tras ello en los ajustes puede activar la funcionalidad y la configuración de la misma. Finalmente cuando quiera introducir los credenciales en otro dispositivo simplemente hay que conectar InputStick al dispositivo y acceder al elemento de cuenta ahí deberá pulsar en el icono de enviar, se le solicitará activar Bluetooth y el campo se escribirá automáticamente en donde se encuentre el foco en el dispositivo. En la figura 15 vemos lo que ocurre a ojos del usuario.

Figura 15: Diagrama de usuario. Realización propia.
El ordenador sobremesa es a modo de ejemplo, pero es sustituible por cualquier dispositivo compatible con USB HID.
Marco de móvil. Imagen por brgfx en Freepik.com.
Marco de monitor. Imagen por d3images en Freepik.com.

3.7. Herramientas

- GitHub: Servidor git donde subir las versiones del proyecto.
- Sublime Merge: Cliente git para controlado de versiones.
- Visual Studio: Entorno de programación usado para el desarrollo de la aplicación, con extensión para desarrollo móvil en Xamarin.
- Obsidian: Editor de markdown para apuntar notas, con la extensión Kanban para la organización del proyecto, de esta forma es similar a otras herramientas como Trello, pero simplificado
- Overleaf: Editor de LaTeX online.

- Docker: Para hospedar la instancia de Vaultwarden se ha usado Docker, esta herramienta crea contenedores donde lanzar programas de forma compartimentada
- StarUML: Para el diagrama de casos de uso.
- Microsoft PowerPoint: Para el diagrama de conexiones.

4. Conclusión

4.1. Dificultades

El mayor problema ha sido sin duda, entender el código de Bitwarden, sobre todo el patrón MVVM, de hecho es algo normal en desarrolladores con no mucha experiencia[31], esto ha llevado mucho más tiempo del esperado, así como un tira y afloja constante con el código pues se ha tenido que refactorizar más veces de las necesarias por falta de comprensión en su funcionamiento. Antes de comenzar con el proyecto nuestras expectativas eran que la dificultad estuviese en usar la librería de InputStick, escrita en Java, en C# y el uso de la librería de InputStick para la comunicación con el mismo. Sin embargo, Visual Studio se ocupa de añadir una capa intermedia para la compatibilidad con la capa, y el uso de InputStickBroadcast y InputStickUtility facilita mucho el uso de la librería. La API de InputStick se puede usar de forma más profunda pero perdiendo las comodidades aportadas por InputStickUtility, esto era el objetivo del proyecto, sin embargo se usó la parte "simple" de la librería a modo de plantilla y a medida que el proyecto avanzaba quedaba cada vez más claro que no iba a dar tiempo de cambiarlo.

4.2. Reflexión

Este proyecto me ha permitido entender mucho mejor el uso de ciertos patrones arquitectónicos, además de indagar en el ámbito de la criptografía que no se entraba en tanta profundidad en la rama que he cursado. Así como aprender a usar LaTeX. También me he familiarizado con C#, lenguaje en el que no había escrito nunca. También he aprendido a estimar mejor el tiempo de trabajo, como se muestra en las tablas 1 y 2, el tiempo dedicado no fue el esperado.

4.3. Futuro del proyecto

Debido a que la estimación del tiempo del proyecto no fue correcta, algunas ideas se han quedado en el tintero, estas son mejoras que se podrían realizar

- Eliminar InputStickUtility como intermediario:

  Al usar InputStickBroadcast, es necesario tener instalada la app InputStickUtility. Con un uso más profundo de la librería de InputStick se puede eliminar esta limitación.

- Conexión directa por Bluetooth, sin InputStick como intermediario:

  Esto eliminaría InputStick en casos donde tenga sentido. Por ejemplo, este proyecto se planteó al querer iniciar sesión en un dispositivo en el que no se confiaba, pero no como única opción. Si alguien quisiese hacer uso de esta funcionalidad con frecuencia en su hogar, por ejemplo en una consola de videojuegos y en una *Smart TV*, que no son capaces de instalar un cliente de Bitwarden. En esta situación resultaría molesto tener que cambiar InputStick con frecuencia entre estos dispositivos, que suelen estar en un mueble o pegados a la pared, y por tanto con acceso incómodo a sus puertos. Sin embargo este tipo de dispositivos con mucha frecuencia tienen la posibilidad de conectar mandos de control mediante Bluetooth. Habría además que seguir teniendo en cuenta el protocolo HID pues Bluetooth usa exactamente la misma definición que en USB[34]. La base hecha en este proyecto facilitaría esto, y esta nueva funcionalidad complementaría la anterior, en el hogar con dispositivos conocidos y compatibles se podría usar Bluetooth, en el exterior y en dispositivos no compatibles se podría usar InputStick.

- Facilitar la identificación de la disposición del tecla:

  Algunos dispositivos no permiten configurar la disposición, lo que resulta en un desconocimiento total de cuál es la disposición que está usando el dispositivo. Un sistema de ayuda para encontrar la disposición podría solventar este problema. Se podría enviar un texto de prueba al dispositivo, en el móvil se muestra y el usuario introduce en el móvil los caracteres que faltan o distintos, entonces el móvil muestra diferentes opciones de lo que puede haber aparecido en su lugar en el

dispositivo, luego el usuario selecciona entre las distintas opciones y el móvil informa al usuario de la disposición que está usando el dispositivo. En la figura 16 se muestra una maqueta. Si un carácter fuese incorrecto el usuario lo podría introducir en el cuadrado situado debajo de dicho carácter. Tampoco es necesario hacer una muestra tan grande, realizando un estudio se podría comprobar qué posiciones tienen tendencia a cambiar entre disposiciones y así minimizar el texto escrito, para no abrumar al usuario.

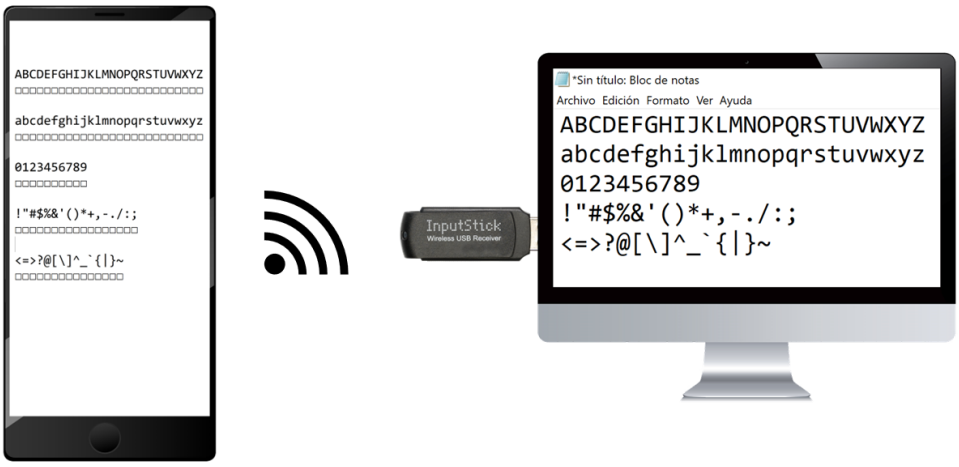


Figura 16: Maqueta de identificación de disposición. Realización propia.
El ordenador sobremesa es a modo de ejemplo, pero es sustituible por cualquier dispositivo compatible con USB HID.
Marco de móvil. Imagen por brgfx en Freepik.com.
Marco de monitor. Imagen por d3images en Freepik.com.

5. Licencias

Este TFT usa códigos Open Source de diversas fuentes, así pues en esta sección se cumple el contrato puesto por estas licencias.

5.1. Bitwarden mobile

Bitwarden mobile usa la licencia GPL 3.0. Como el objetivo principal de este proyecto es la modificación de este código, los cambios realizados se encuentran en el fork de GitHub de PabloOQ. Debido a un problema de compatibilidad durante el desarrollo hubo que hacer un merge de la rama master de Bitwarden, todo el código se encuentra en:
https://github.com/PabloOQ/mobile/tree/stick-after-merge
Sin embargo, para visualizar rápidamente el código modificado antes de hacer el merge, se podría ver en esta rama (aunque este está también en la rama mencionada anteriormente):
https://github.com/PabloOQ/mobile/tree/stick

5.2. Vaultwarden

Vaultwarden usa la licencia GPL 3.0. Este código no ha sido modificado, simplemente se ha usado para crear una estancia vaultwarden en la que hacer pruebas de la aplicación de Bitwarden para Android.

5.3. InputStick

En GitHub la API de InputStick no tiene la licencia establecida, por ello se le preguntó personalmente al creador y él dió el visto bueno, sin poner ningún tipo de condición.

5.4. GNU GPL 3

https://www.gnu.org/licenses/gpl-3.0.en.html

Referencias

[1] "Bitwarden vault item," Bitwarden. [Online]. Available: https://bitwarden.com/help/managing-items/

[2] "Bitwarden cache," Bitwarden. [Online]. Available: https://bitwarden.com/help/security-faqs/#q-how-long-does-bitwarden-cache-session-information

[3] "Inputstick - compatibility," InputStick. [Online]. Available: http://inputstick.com/compatibility/

[4] "Inputstick - how it works," InputStick. [Online]. Available: http://inputstick.com/how-it-works/

[5] "Inputstick faq," InputStick. [Online]. Available: http://inputstick.com/faq/

[6] "Inputstick security," InputStick. [Online]. Available: http://inputstick.com/security/

[7] "Pwned websites," haveibeenpwned. [Online]. Available: https://haveibeenpwned.com/PwnedWebsites

[8] "Gabe newell gives out his password while presenting a safety feature of steam." [Online]. Available: https://www.youtube.com/watch?v=gYs9nS8LlZ8

[9] "Bitwarden send," Bitwarden. [Online]. Available: https://bitwarden.com/blog/bitwarden-send-how-it-works/

[10] "1password security design," 1Password. [Online]. Available: https://1passwordstatic.com/files/security/1password-white-paper.pdf

[11] "How bitwarden works," Bitwarden. [Online]. Available: https://bitwarden.com/products/#how-bitwarden-works

[12] "Keepass scurity," KeePass. [Online]. Available: https://keepass.info/help/base/security.html

[13] "Lastpass - how it works," LastPass. [Online]. Available: https://www.lastpass.com/how-lastpass-works

[14] "Authorizer," tejado. [Online]. Available: https://github.com/tejado/Authorizer

[15] "Arduino keyboard library," Arduino. [Online]. Available: http://reference.arduino.cc/reference/en/libraries/keyboard/

[16] "Quantum mechanical keyboard," QMK. [Online]. Available: https://qmk.fm/

[17] "Kmk firmware," KMKfw. [Online]. Available: http://kmkfw.io/

[18] "Vial," Vial. [Online]. Available: https://get.vial.today/

[19] "Zmk," ZMK. [Online]. Available: https://zmk.dev/

[20] "Via," VIA. [Online]. Available: https://www.caniusevia.com/

[21] "Laa," jerry08. [Online]. Available: https://github.com/jerry08/Laa

[22] "Android keyboard gadget," pelya. [Online]. Available: https://github.com/pelya/android-keyboard-gadget

[23] "Ble hid over gatt profile for android," kshoji. [Online]. Available: https://github.com/kshoji/BLE-HID-Peripheral-for-Android

[24] "Pwned passwords," haveibeenpwned. [Online]. Available: https://haveibeenpwned.com/Passwords

[25] Keyboard and mouse hid client drivers. Microsoft. [Online]. Available: https://learn.microsoft.com/en-us/windows-hardware/drivers/hid/keyboard-and-mouse-hid-client-drivers

[26] USB. [Online]. Available: https://www.usb.org/document-library/device-class-definition-hid-111

[27] Change your keyboard layout. Microsoft. [Online]. Available: https://support.microsoft.com/en-us/windows/change-your-keyboard-layout-245c49b8-f856-7fd7-2cf5-41e54c66f5b3

[28] R. F. García, *MVVM: Model–View–ViewModel*. Berkeley, CA: Apress, 2023, pp. 145–224. [Online]. Available: https://doi.org/10.1007/978-1-4842-9069-9_4

[29] J. Kouraklis, *MVVM as Design Pattern*. Berkeley, CA: Apress, 2016, pp. 1–12. [Online]. Available: https://doi.org/10.1007/978-1-4842-2214-0_1

[30] E. Gamma, *Design patterns : elements of reusable object-oriented software / Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.*, Jan 1994. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=cat07429a&AN=ulpgc.183475&site=eds-live

[31] R. F. García, *Introduction*. Berkeley, CA: Apress, 2023, pp. 1–43. [Online]. Available: https://doi.org/10.1007/978-1-4842-9069-9_1

[32] "inputstickapi for android os," InputStick. [Online]. Available: https://github.com/inputstick/InputStickAPI-Android

[33] "Binding a jar," Microsoft. [Online]. Available: https://learn.microsoft.com/en-us/xamarin/android/platform/binding-java-library/binding-a-jar

[34] Bluetooth hid. Bluetooth. [Online]. Available: https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=7108