TimeLock: aplicativo autenticador TOTP implementado em Android nativo

Talles Rezende Espíndola

PROJETO DE CONCLUSÃO DE CURSO

Data de Depósito: 04/12/2023

Assinatura: _____

TimeLock: aplicativo autenticador TOTP implementado em Android nativo

Talles Rezende Espíndola

Valter Ribeiro Lima Júnior

Monografia apresentada ao Instituto Superior de Educação do UNIFOR/MG, como requisito parcial para obtenção do título de bacharel em Ciência da Computação, sob a orientação do Profº. Me. Valter Ribeiro Lima Júnior

Unifor-MG - Formiga
2023

À minha família, amigos e todos os que me ajudaram ao longo desta caminhada.

Agradecimentos

Em primeiro lugar, gostaria de expressar minha sincera gratidão à minha família, alicerce inabalável que sempre me apoiou e encorajou ao longo desta jornada acadêmica. A vocês, meu eterno agradecimento.

Agradeço também a todos os professores do Centro Universitário de Formiga (UNIFOR-MG) pelos ensinamentos e por compartilharem seus conhecimentos técnicos e experiências. Em especial ao Prof. Me. Valter Ribeiro, por sua orientação, essencial na conclusão deste projeto.

"Não se compare com ninguém mais nesse mundo. Se você fizer isso, estará se insultando."

Bill Gates

Sumário

Lista de Siglas

HMAC - *Hash-based Message Authentication Code* - Código de Autenticação de Mensagem baseado em Hash
HOTP - *HMAC-based One Time Password* - Senha de Uso Único baseada em HMAC
HTML5 - *Hypertext Markup Language 5* - Linguagem de Marcação de Hipertexto 5
iOS - *iPhone Operating System* - Sistema Operacional do iPhone
JDK - *Java Development Kit* - Kit de Desenvolvimento Java
JRE - *Java Runtime Environment* - Ambiente de Tempo de Execução Java
JVM - *Java Virtual Machine* - Máquina Virtual Java
NoSQL - *Not only SQL* - Não apenas SQL
OTP - *One Time Password* - Senha de Uso Único
PIN - *Personal Identification Number* - Número de Identificação Pessoal
RFC - *Request For Comments* - Requisição Para Comentários
SDK - *Software Development Kit* - Kit de Desenvolvimento de Software
SQL - *Structured Query Language* - Linguagem de Consulta Estruturada
TOTP - *Time-based One Time Password* - Senha de Uso Único baseada no Tempo

Lista de Figuras

Lista de Tabelas

Resumo

ESPINDOLA, T. R. *TimeLock: aplicativo autenticador TOTP implementado em Android nativo.* Monografia (Graduação) — Centro Universitário de Formiga – UNIFOR-MG – Formiga, 2023.

Com o desenvolvimento incessante da tecnologia e a constante digitalização dos mais diversos tipos de serviços, cada vez mais as pessoas armazenam informações importantes em contas *online*. A partir do momento que estas informações envolvem dados pessoais, cartões de crédito ou contas bancárias, uma simples senha se torna uma barreira muito fraca entre a segurança de uma conta e possíveis criminosos cibernéticos. O propósito deste projeto é desenvolver um aplicativo para dispositivos móveis capaz de incrementar a segurança de contas *online* por meio da autenticação em dois fatores. A implementação foi realizada utilizando a linguagem Java para Android e os serviços de autenticação e armazenagem de dados em nuvem do Firebase. O aplicativo apresenta uma interface de usuário simples, pela qual os usuários podem configurar a autenticação em dois fatores em suas contas *online*. Ao realizar o *login* em um serviço com a autenticação em dois fatores configurada, será solicitado, além da senha da conta, um código único. Este código é gerado pelo aplicativo autenticador e é válido por um pequeno período de tempo, o que efetivamente adiciona uma camada extra de segurança ao processo de *login*.

Palavras-chave: Autenticador, Aplicativo, Android, Java, Autenticação em dois fatores.

Abstract

ESPINDOLA, T. R. *TimeLock: TOTP authenticator app implemented in native Android.* Monograph (Graduation) — Centro Universitário de Formiga – UNIFOR-MG – Formiga, 2023.

With the unceasing development of technology and the constant digitalization of various types of services, people are increasingly storing important information in online accounts. As these pieces of information often involve personal data, credit cards, or bank accounts, a simple password becomes a weak barrier between an account's security and potential cybercriminals. The purpose of this project is to develop a mobile application capable of enhancing the security of online accounts through two-factor authentication. The implementation was carried out using the Java language for Android and Firebase for authentication services and cloud data storage. The application features a simple user interface which allows users to configure two-factor authentication for their online accounts. When logging into a service with two-factor authentication, in addition to the account password, a unique code will be requested. This code is generated by the authenticator app and is valid for a short period, effectively adding an extra layer of security to the login process.

Keywords: Authenticator, App, Android, Java, Two-factor authentication.

Capítulo

1

Introdução

1.1 Considerações Iniciais

Desde sua criação, a Internet cresce continuamente, e hoje pode ser considerada uma parte essencial da vida de grande parte das pessoas. “Existem 5.16 bilhões de usuários de internet no mundo hoje, o que significa que 64,4% da população mundial está *online*” (KEMP, 2023).

Esse crescimento é justificado com a variedade de serviços disponíveis na Internet. Além daqueles que não existiam antes dela, como redes sociais e fóruns, há também serviços que foram incrementados ou facilitados graças à ela, como lojas *online* que substituiram a necessidade de ir a uma loja física e serviços de *streaming* que facilitaram o consumo de mídias como filmes, séries e músicas.

A maioria desses serviços faz o uso de contas *online*, que são formas de identificar os usuários e garantir que apenas o dono de uma conta tenha acesso a ela, por meio da combinação de um nome de usuário ou e-mail e uma senha: um valor secreto, definido pelo usuário.

Vários serviços já faziam uso de senhas como uma forma de garantir segurança àqueles que os utilizam, como contas de bancos e cartões de crédito. Porém, a facilidade de acesso à informação que a Internet permite também pode ser prejudicial.

A senha de uma conta *online* deve ser armazenada de alguma forma na Internet, pois para que o processo de *login* funcione, a senha inserida pelo usuário em um dispositivo qualquer deve ser comparada com a senha definida na criação da conta, para garantir ou negar o acesso à conta. Sabendo que há um caminho até informações como essas senhas atráves da Internet, cibercriminosos tentam encontrá-lo para obter acesso à dados sensíveis de outros usuários. Essa é uma ocorrência frequente, como estima um levantamento do SafeLabs em parceria com a ISH, que indica que 30,18 milhões de senhas foram vazadas no Brasil no ano de 2022 (ID, 2023).

Diante destes fatos, muitos provedores de serviços entendem que uma senha pode não oferecer segurança suficiente a uma conta e solicitam outras informações ao usuário no momento do *login*, como uma confirmação por e-mail, uma mensagem de SMS ou um código temporário.

1.2 Objetivos do Trabalho

Para tanto, este trabalho tem como objetivo central o desenvolvimento de um aplicativo autenticador TOTP para dispositivos móveis Android. A abreviação TOTP significa *Time-based One Time Password*, ou Senha de Uso Único Baseada no Tempo.

Este tipo de aplicativo deve ser capaz de gerar e fornecer códigos secretos e temporários para a autenticação de dois fatores em contas de outros aplicativos, sites ou serviços que possuam suporte à este tipo de autenticação.

A aplicação deve possuir duas formas de configurar uma nova conta para a autenticação em dois fatores: inserindo um código manualmente, ou por meio da leitura de um QR Code. Também deve exibir uma lista com todas as contas configuradas e seus códigos de acesso.

Deve contar também com um controle de usuários, para que as contas autenticadas sejam armazenadas e vinculadas a cada usuário. Além disso, é essencial que a interface de usuário seja simples e intuitiva, para que ele consiga se orientar facilmente ao utilizar a aplicação.

A implementação foi realizada com a linguagem Java para Android devido ao fácil acesso de ferramentas a nível de hardware do dispositivo móvel. Produtos do Firebase foram utilizados para prover a autenticação de usuários e o armazenamento de dados de forma segura e integrada à linguagem Java.

1.3 Estrutura da Monografia

Este trabalho foi estruturado em 6 Capítulos, da seguinte maneira: O Capítulo 1 introduz o tema central do projeto e apresenta a proposta principal e seus objetivos. No Capítulo 2 é detalhado o referencial teórico, que contém os principais conceitos, teorias e tecnologias utilizadas no desenvolvimento deste projeto. O Capítulo 3 contém o estado da arte, que apresenta outros projetos e aplicações com propostas semelhantes à do presente trabalho. No Capítulo 4 é descrita a metodologia do projeto, onde é apresentado o planejamento e detalhado como as tecnologias selecionadas foram utilizadas na implementação. Já no Capítulo 5 são expostos os resultados da implementação do projeto, demonstrando as telas da aplicação e seu funcionamento. Por fim, no Capítulo 6 é apresentada a conclusão e as considerações finais sobre a aplicação. Também são citados os trabalhos futuros do projeto.

Capítulo

2

Referencial Teórico

2.1 Considerações Iniciais

Neste capítulo serão apresentadas as ferramentas utilizadas e os principais conceitos e teorias necessários para o desenvolvimento deste projeto.

2.2 Aplicativos Móveis

A popularização dos aplicativos móveis, popularmente conhecidos como *apps*, teve início em 2007, ano do lançamento do iPhone, o primeiro *smartphone* com o sistema operacional iOS. Essas aplicações são *softwares* desenvolvidos exclusivamente para celulares e tablets, generalizados como dispositivos móveis (BOCARD, 2021).

Segundo Holovko (2022), o mercado global de aplicativos está se desenvolvendo de forma acelerada: esse mercado atingiu o valor de $206,73 bilhões em 2022, e é estimado que cresça em 13,4% até 2030. Como comparativo, em 2016, as pessoas de todo o mundo fizeram o *download* de 140,68 milhões de aplicativos, enquanto em 2021, esse número chegou aos 230 bilhões.

Apesar do tamanho do mercado de aplicações móveis, seu desenvolvimento atualmente é focado em apenas dois sistemas operacionais: Android e iOS. Em fevereiro de 2023, a participação de mercado do Android era de 72,27% e do iOS era de 27,1%, com apenas 0,63% representando outros sistemas operacionais (STATS, 2023). Dito isso, percebe-se que a disponibilidade multi-plataformas é um fator a ser considerado no desenvolvimento, e duas opções possíveis neste quesito são aplicações nativas ou híbridas.

2.2.1 Aplicativos Nativos

Aplicativos nativos são desenvolvidos para dispositivos e sistemas operacionais móveis específicos, como iOS ou Android. Aplicativos Android são escritos em Java e Kotlin, enquanto aplicativos para iOS são escritos em Objective-C ou Swift (WELLINGTON, 2021).

Um aplicativo nativo possui um melhor desempenho devido à sua integração direta com o dispositivo e com o sistema operacional, fornece uma aparência mais consistente, uma segurança maior e ainda conta com fácil acesso aos recursos integrados do dispositivo, como câmera, GPS e Bluetooth (IBM, 2023).

No entanto, desenvolver aplicações para apenas uma plataforma pode ocasionar uma perda de oportunidades, caso apenas uma plataforma seja levada em consideração. Contornar essa desvantagem também gera problemas, pois criar um mesmo aplicativo nativo para mais de um sistema operacional pode prolongar significativamente o processo de desenvolvimento, afinal um mesmo código terá de ser convertido e reescrito em outras linguagens (LEDU, 2018).

2.2.2 Aplicativos Híbridos

Aplicações híbridas são *web-apps* desenvolvidos com HTML5 e JavaScript, e então empacotados para se comportarem como um aplicativo nativo. Elas são executadas em mais de um sistema operacional, e os dados do aplicativo são fornecidos por servidores de aplicativos, por meio de chamadas de API de Serviço da Web. Algumas plataformas de desenvolvimento de aplicativos híbridos são IBM Worklight, Cordova, Angular e Ionic (IBM, 2023).

De acordo com Lau (2022), aplicativos híbridos podem ser utilizados em diversas plataformas e dispositivos com um mesmo código de programação, o que leva a um desenvolvimento mais fácil e rápido, atualizações mais fáceis de serem implementadas e uma necessidade de manutenção menos frequente.

No entanto, aplicações híbridas apresentam como principais desvantagens a perda geral de desempenho e otimização. Por serem essencialmente aplicativos *Web*, o desempenho é completamente dependente da velocidade do navegador do usuário. Além disso, o aplicativo não contará com exatamente o mesmo *"look and feel"* (em português, olhar e sentir) em mais de um sistema operacional, pois o sistema iOS tem um estilo de interface e navegação diferente do Android (DACRE, 2019).

2.3 Java

O aplicativo desenvolvido neste projeto foi escrito em Java, uma linguagem de programação multiplataforma orientada a objetos que também funciona como uma plataforma própria. Sua popularidade se deve à ampla disponibilidade de funções, bibliotecas, ferramentas de desenvolvimento, recursos de aprendizado e ao suporte ativo da comunidade (AWS, 2023).

Programas escritos em Java não executam instruções diretamente no computador. Na verdade, eles são executados na *Java Virtual Machine* (JVM), uma máquina virtual composta por *software* que simula uma máquina física. A JVM torna o Java multiplataforma, pois permite que um mesmo código seja executado em diversos sistemas operacionais (FARRELL, 2022).

As ferramentas necessárias para escrever códigos em Java estão contidas no *Java Development Kit* (JDK). O software que executa programas em Java é o *Java Runtime Environment* (JRE), um pacote de formado por classes, bibliotecas e pela JVM. Em dispositivos móveis, no entanto, a capacidade de executar códigos em Java normalmente é integrada pela fabricante do dispositivo e faz parte da base de seu sistema operacional (ORACLE, 2022).

2.4 Banco de Dados

Segundo Alves (2014), um banco de dados é, de forma genérica, um conjunto de dados com um significado implícito. De forma mais específica, é a representação de uma fração do mundo real, com um conjunto lógico ordenado de dados e um objetivo definido. Existem dois principais tipos de bancos de dados: relacionais e não relacionais.

Um banco de dados relacional consiste em uma coleção de tabelas, em que cada linha representa um relacionamento entre um conjunto de valores, e por meio de identificadores, é possível relacionar várias tabelas de modo a unir os dados e obter informações mais completas (SILBERSCHATZ, 2020).

Bancos de dados não relacionais (conhecidos como NoSQL, ou "não apenas SQL"), possuem um esquema mais flexível, alta escalabilidade horizontal e maior capacidade de lidar com um alto tráfego de dados. No entanto, não trabalham com dados normalizados e não possuem a capacidade de realizar junções, subconsultas e o aninhamento de consultas utilizados nas consultas mais complexas de bancos relacionais (ORACLE, 2023).

2.4.1 Firebase

O Firebase, ferramenta usada neste projeto para realizar a autenticação de usuários e o armazenamento dos dados, é uma plataforma digital criada por James Tamplin e Andrew Lee em 2011 e posteriormente adquirida pela Google, em 2014. Sua base é construída na infraestrutura do Google, oferecendo uma plataforma digital que pode ser utilizada no desenvolvimento de aplicativos para Android, iOS e Web (SILVA, 2022).

O Firebase é um *back-end as a service*, que oferece serviços de autenticação, armazenamento de dados em nuvem, análise de dados, notificações e muitos outros. A ideia da plataforma é tratar de todo o *back-end*, de forma que o desenvolvedor precise apenas ativar e configurar seus serviços e possa focar no desenvolvimento da aplicação do *front-end* (RIBEIRO, 2023).

Como pode ser visto em Google (2023d), devido à grande variedade de funcionalidades oferecidas, o Firebase divide seus serviços por produtos. A seguir estão alguns produtos que fazem parte do Firebase:

- *Firebase Authentication*: fornece serviços de autenticação de usuários utilizando senhas, números de telefone e provedores de identidade reconhecidos como Google, Facebook e Twitter (GOOGLE, 2023c).

- *Cloud Firestore*: oferece um banco de dados NoSQL flexível e escalonável com atualizações em tempo real, suporte *offline* e consultas expressivas (GOOGLE, 2023a).

- *Realtime Database*: possui um banco de dados hospedado em nuvem que armazena dados como JSON e é capaz de sincronizá-los em tempo real para todos os clientes conectados (GOOGLE, 2023e).

- *Cloud Storage*: serviço de armazenamento de objetos capaz de realizar *upload* e *download* de arquivos nos aplicativos do Firebase com a segurança do Google (GOOGLE, 2023b).

2.5 Segurança da Informação

A segurança da informação é um conceito que abrange diversas medidas que garantem os critérios de integridade, confidencialidade e disponibilidade dos dados sobre uma empresa ou de pessoas físicas. Seu principal objetivo é a proteção dos dados e certificar que só possam ser acessados e modificados por pessoas autorizadas (NEOWAY, 2020).

Para alcançar este objetivo, existem dois tipos de controles de segurança: físicos e lógicos. Os controles físicos envolvem o uso de câmeras de segurança, alarmes, portas, muros e fechaduras. Controles lógicos utilizam softwares como controladores de acesso (senhas, biometria, métodos de autenticação), criptografia, assinaturas digitais e certificações para monitorar e controlar o acesso às informações (TUTIDA, 2021).

2.5.1 Roubo de credenciais

Segundo Strickland (2021), "A complexidade necessária para a prestação de serviços totalmente digitais também representa riscos". Esses riscos envolvem o roubo de credenciais, uma das táticas utilizadas por cibercriminosos.

O Relatório de vazamento de dados no Brasil, da Axur, indica que ocorreram 465,5 milhões de registros vazados no segundo trimestre de 2021. O roubo de credenciais pode causar danos como prejuízos financeiros e a perda de reputação (SANTOS, 2022).

Essa tática consiste no roubo, compra ou vazamento de informações de usuários, que são utilizados em vários outros sites por criminosos, na tentativa de conseguir o acesso a contas de bancos, redes sociais e e-mails, por exemplo (STRICKLAND, 2021).

2.5.2 Autenticação de dois fatores e multifator

Nesse contexto, existem algumas medidas que tem o intuito de adicionar camadas de segurança adicionais no processo de *login*. Uma delas é a autenticação de dois fatores, que exige que o usuário forneça duas formas de autenticação: uma senha e um outro fator, como um OTP (*One time password*), que pode ser uma mensagem de texto ou de voz, um token físico ou um token gerado por software, como códigos TOTP (AUTHY, 2023c).

Uma outra opção é a autenticação multifator. De acordo com CISA (2022), para realizar o *login* em um aplicativo ou site, esse tipo de autenticação geralmente exige que o usuário apresente uma combinação dos elementos a seguir:

- Algo que você sabe: uma senha ou um número de identificação pessoal (PIN);
- Algo que você tem: um cartão ou token;
- Algum fator biométrico: digitais, reconhecimento de voz ou facial.

2.5.3 Autenticação com algoritmos HMAC

De acordo com Ainsworth (2022), existem dois tipos principais de OTPs: HOTP (*HMAC-based OTP*) e TOTP (*Time-based OTP*). Ambos usam uma chave secreta ou *seed* e um fator em movimento, que é o diferencial de cada método: em algoritmos HOTP esse fator é um contador baseado em eventos e em algoritmos TOTP o contador é baseado no tempo.

Estes algoritmos foram definidos em publicações denominadas RFCs (*Request for Comments*), documentos formais que contém especificações sobre tópicos relacionados à Internet e redes de computadores. Os algoritmos HOTP e TOTP são detalhados nos RFCs 4266 (M'RAIHI et al., 2005) e 6238 (M'RAIHI et al., 2011), respectivamente.

Tanto HOTP quanto TOTP utilizam o algoritmo HMAC (*Hash-based message authentication code*), que de forma simplificada utiliza uma função criptográfica de *hash* que recebe como parâmetros uma chave secreta e um contador e gera um *hash* de 160 bits. Esse *hash* é truncado para um número menor de *bits* e convertido para um número inteiro, produzindo um código que pode ser digitado facilmente (RUBLON, 2022).

Segundo Smith (2018), um algoritmo HOTP usa um contador baseado em eventos e uma chave secreta compartilhada entre um token físico e o servidor. O contador do token incrementa quando um botão é pressionado, enquanto o contador do servidor incrementa quando um OTP é validado e o usuário realiza a autenticação com sucesso. É possível que o usuário incremente o contador do token sem realizar uma autenticação, fazendo com que o servidor também precise validar possíveis incrementos de seu próprio contador.

A autenticação por TOTP, utilizada neste projeto, substitui o contador de eventos por um contador de tempo. Esse contador é calculado dividindo o tempo atual do Unix pelo tempo de vida do OTP (normalmente 30 segundos). Tanto o autenticador TOTP quando o servidor calculam um novo OTP sempre que o anterior expira, e o usuário deve inserir o valor exibido no autenticador em um campo de texto na página de *login*. O servidor então compara seu próprio OTP com o valor inserido e se ambos forem iguais, o servidor permite o acesso ao usuário (RUBLON, 2022).

2.6 Considerações Finais

Neste capítulo foram apresentadas e descritas as ferramentas e os conceitos que são as principais referências para o planejamento e a implementação da aplicação.

Capítulo

3

Estado da Arte

Neste capítulo serão apresentados trabalhos que possuem um objetivo semelhante ao do aplicativo desenvolvido neste trabalho. Os aplicativos foram escolhidos com base em sua popularidade e avaliação na Google Play Store.

3.1 Google Authenticator

O Google Authenticator é um aplicativo lançado em 2010 que possui como finalidade adicionar uma camada extra de segurança à contas *online*.

O autenticador é configurado lendo um QR Code (fornecido pelo serviço em que será ativada a autenticação) com a câmera do celular. Após concluir esse processo, o aplicativo passará a gerar códigos que serão requisitados quando o usuário tentar realizar o *login*, como pode ser observado na Figura 3.1(a). O código pode ser gerado mesmo que o telefone não possua conexão com a internet (GOOGLE, 2023f).

O aplicativo possui suporte para a geração de códigos baseados em tempo (TOTP) e em contadores (HOTP) e pode ser utilizado para gerenciar e autenticar múltiplas contas. Não possui muitas funções adicionais, além da configuração e geração dos códigos e de opções para importar e exportar contas configuradas (GOOGLE, 2023f).

Apesar de ser o aplicativo que fornece autenticação por TOTP mais conhecido popularmente, possui muitas reclamações devido à dificuldade de recuperar o acesso às contas autenticadas após perder o celular. Em resposta à esse problema, a Google adicionou ao aplicativo a função de *Cloud syncing* na versão 4.0, que permite a sincronização dos códigos do autenticador a uma conta do Google.

3.2 Microsoft Authenticator

O Microsoft Authenticator, lançado em 2015, é uma ferramenta desenvolvida pela Microsoft para oferecer ao usuário uma maior segurança no processo de *login*, com novas formas de se autenticar e etapas adicionais que podem ser habilitadas ao realizar *login*.

O foco do aplicativo é o "ecossistema" da Microsoft, contando com uma autenticação multifator que exige a senha do usuário e um código OTP gerado no aplicativo sempre que for necessário realizar *login* em uma conta Microsoft. Na Figura 3.1(b) podemos ver um exemplo de sua interface.

O autenticador também possui a opção de substituir totalmente a senha de uma conta da Microsoft por uma notificação enviada para o aplicativo que deve ser aprovada para realizar *login*, junto de um PIN, uma digital ou reconhecimento facial (MICROSOFT, 2023a).

No entanto, o Microsoft Authenticator também pode ser utilizado como fornecedor de códigos TOTP para contas de diversos outros serviços, como Facebook, Amazon, LinkedIn, GitHub e até mesmo da Google. Outro diferencial do aplicativo é uma função de armazenamento de senhas, que após salvas são preenchidas automaticamente ao realizar *login* (MICROSOFT, 2023b).

O autenticador da Microsoft possui avaliações positivas em geral, tanto na Apple App Store quanto na Google Play Store. As poucas avaliações negativas, no entanto, se devem ao mesmo problema presente no Google Authenticator, em que usuários perderam o acesso ao telefone e não conseguem desativar a autenticação multifator.

3.3 Twilio Authy Authenticator

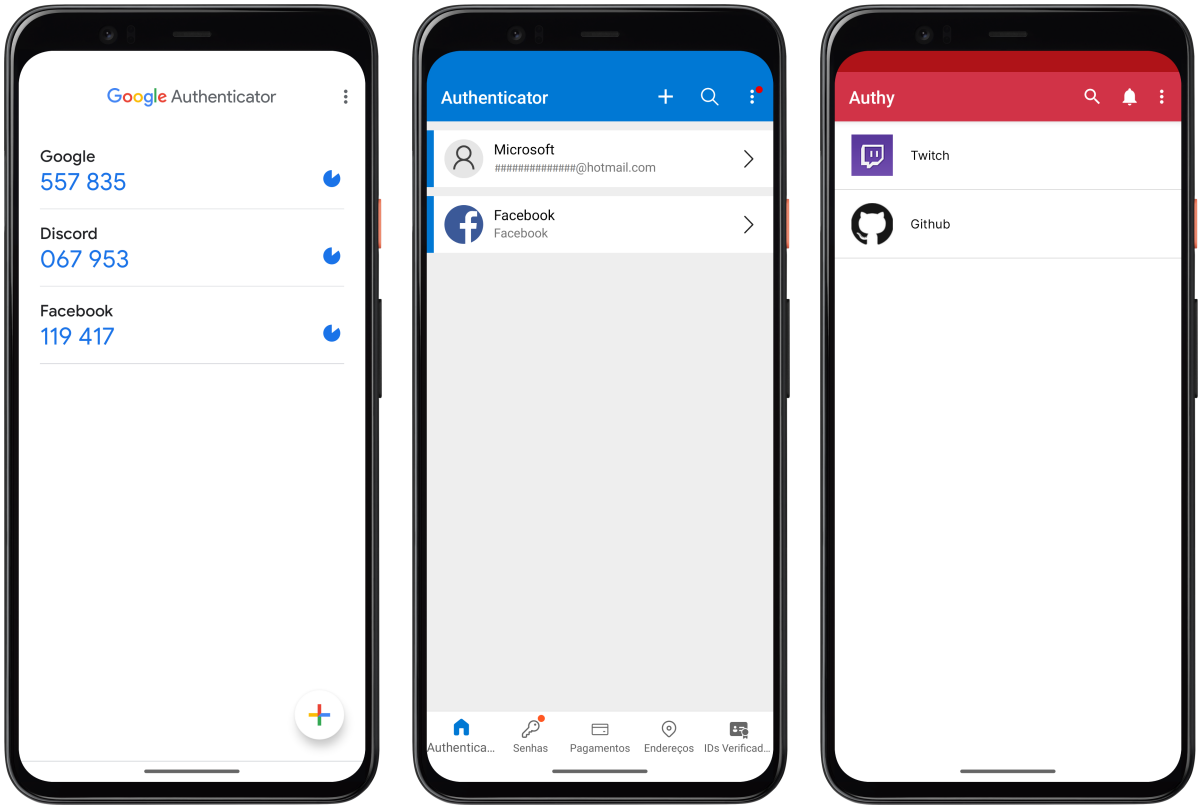
O Twilio Authy é um aplicativo que fornece serviços de autenticação de dois fatores por TOTP, SMS e telefone lançado em 2012. O aplicativo pode ser utilizado tanto em dispositivos móveis com sistemas Android e iOS quanto em computadores com os sistemas operacionais Windows, Linux e macOS (TWILIO, 2015).

O aplicativo oferece suporte geral à autenticação multifator em contas *online*, permite ao usuário realizar backups das contas configuradas na nuvem e sincronizar as contas autenticadas em vários dispositivos. Também é capaz de gerar códigos mesmo sem acesso à internet e pode ser utilizado para proteger carteiras digitais (AUTHY, 2023b).

Além disso, o acesso ao aplicativo pode ser protegido com um código PIN ou por biometria, os códigos armazenados em nuvem são criptografados e é possível gerenciar os dispositivos que tem acesso aos códigos da conta conectada (AUTHY, 2023a).

O Authy possui avaliações positivas na Google Play Store, na Apple App Store e no próprio site (onde a versão para computadores está disponível para download). Sua proposta é ser uma solução para autenticação multifator semelhante ao Google Authenticator, porém com uma maior disponibilidade de acesso aos códigos OTP entre dispositivos (AUTHY, 2017). A Figura 3.1(c) exibe a interface principal do aplicativo.

Figura 3.1: Autenticadores.



(a) Google Authenticator (b) Microsoft Authenticator (c) Twilio Authy Authenticator

Fonte: Elaborado pelo autor.

Capítulo

4

Metodologia

Neste capítulo serão descritos os estudos realizados durante a idealização do projeto e como foi planejado o desenvolvimento do aplicativo.

4.1 Projeto da aplicação

Inicialmente, foi realizado um estudo sobre o desenvolvimento de aplicações para aplicativos móveis em geral, e então uma pesquisa a respeito de aplicativos autenticadores, para estudar o que os usuários pensam a respeito de aplicativos semelhantes já disponíveis no mercado.

Também foi necessário um estudo para o desenvolvimento de aplicações nativas para Android em específico e da linguagem Java para Android, escolhida devido ao fácil acesso às ferramentas do hardware como a câmera do *smartphone* e o leitor de QR Code.

Para a criação da aplicação foi utilizado o ambiente de desenvolvimento integrado Android Studio, devido à sua vasta gama de ferramentas que auxiliam na criação de aplicações nativas para Android. Entre essas ferramentas está a possibilidade de executar e depurar as aplicações desenvolvidas em um emulador integrado ou em um *smartphone* Android.

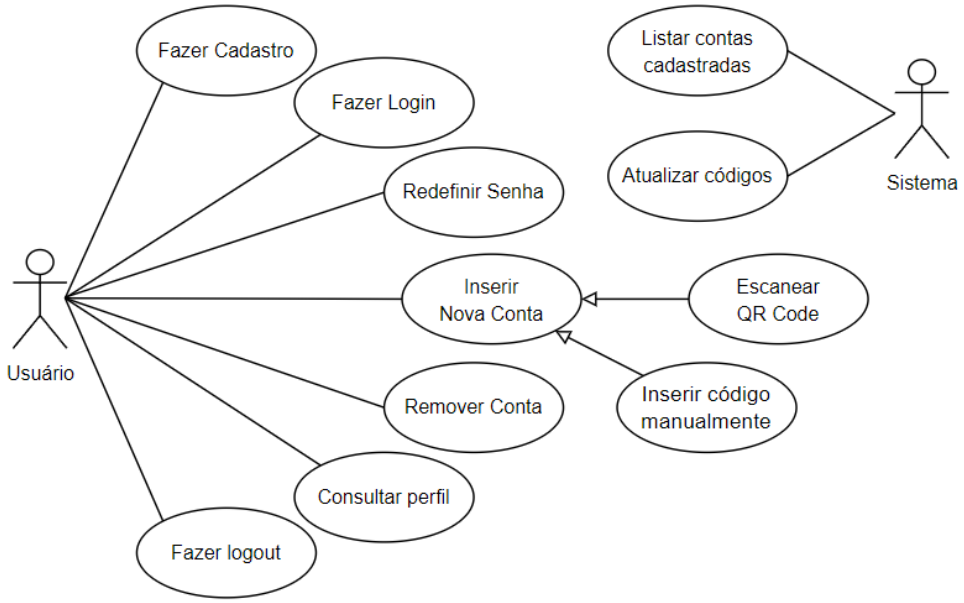
Para o gerenciamento de usuários e a realização do cadastro e *login* foi usado o serviço de Autenticação do Firebase, e para o armazenamento das contas autenticadas também foi utilizado o serviço de banco de dados em tempo real Firestore do Firebase.

4.2 Casos de uso

Um diagrama de caso de uso é uma forma de visualizar as funções que um sistema possui e o fluxo de ações que os atores (usuários, sistemas ou hardwares externos) devem percorrer para executá-las.

O diagrama de casos de uso do aplicativo TimeLock foi construído utilizando a ferramenta *online* gratuita Draw.IO e pode ser observado na Figura 4.1.

Figura 4.1: Diagrama de Casos de Uso.



Fonte: Elaborado pelo autor.

Os casos da aplicação representados são cadastro, *login*, redefinição de senha, inserir uma nova conta (por meio do scan de um QR Code ou inserindo manualmente), remover uma conta, consultar o perfil, fazer logout, listar as contas cadastradas e atualizar os códigos.

Na Tabela 4.1 está especificado o cadastro de usuário. Para se cadastrar é necessário que o usuário possua um e-mail previamente, e ao finalizar o processo, é liberado o acesso à aplicação.

Tabela 4.1: UC001: Fazer Cadastro

Ator	Usuário
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de cadastro; 3) Preencher os campos de e-mail e senha e pressionar o botão Cadastrar.
Fluxo alternativo	1) Não preencher algum dos campos. 2) Uma mensagem de erro é exibida para o usuário.
Pré-condições	1) Possuir um endereço de e-mail.
Pós-condições	1) O usuário é direcionado à tela de Início.

A Tabela 4.2 detalha o processo de *login*. Para realizar o *login*, o usuário precisa antes ter realizado o cadastro (descrito na Tabela 4.1) e ao finalizar o processo pode acessar a aplicação.

Tabela 4.2: UC002: Fazer Login

Ator	Usuário
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de *login*; 3) Preencher os campos de e-mail e senha e pressionar o botão Entrar.
Fluxo alternativo	1) Não preencher algum dos campos ou inserir e-mail ou senha incorretos. 2) Uma mensagem de erro é exibida para o usuário.
Pré-condições	1) Possuir um cadastro na aplicação.
Pós-condições	1) O usuário estará logado no aplicativo e será direcionado à tela de Início.

Na Tabela 4.3 está descrito o processo de redefinição de senha. Para alterar sua senha, o usuário deve selecionar uma opção na tela de *login* e então informar o e-mail da conta. Após isso, deve acessar o *link* enviado ao seu e-mail para informar a nova senha.

Tabela 4.3: UC003: Redefinir senha

Ator	Usuário
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de *login*; 3) Selecionar a opção ”Esqueceu sua senha?” e informar o e-mail; 4) Acessar o e-mail e clicar no *link* enviado para redefinição; 5) Inserir a nova senha.
Fluxo alternativo	1) Inserir um e-mail inválido. 2) Uma mensagem de erro é exibida para o usuário.
Pré-condições	1) Possuir um cadastro na aplicação.
Pós-condições	1) A senha do usuário com o e-mail informado é alterada.

Nas Tabelas 4.4 e 4.5 estão especificadas as duas formas de configurar uma nova conta para autenticação, sendo elas por meio da leitura de um QR Code com a câmera do celular, ou pelo preenchimento manual com os dados da conta.

Para utilizar o método de leitura do QR Code, o usuário precisa permitir ao aplicativo o acesso à câmera do aparelho, e o código deve ter o formato utilizado pelo Google Authenticator. Após inserir uma nova conta, o usuário é redirecionado à tela de Início.

Tabela 4.4: UC004: Inserir Nova Conta - Leitura de QR Code

Ator	Usuário
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de nova conta; 3) Pressionar o botão ”Ler QR Code” e apontar a câmera para o código.
Fluxo alternativo	1) Ler um QR Code inválido. 2) Uma mensagem de erro é exibida e a conta não é adicionada.
Pré-condições	1) Estar logado na aplicação; 2) Permitir o acesso à câmera do aparelho.
Pós-condições	1) A nova conta será inserida.

Tabela 4.5: UC005: Inserir Nova Conta - Manualmente

Ator	Usuário
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de nova conta; 3) Preencher os campos de nome da conta, e-mail e código; 4) Pressionar o botão "Adicionar Conta."
Fluxo alternativo	1) Preencher o campo do código com um valor inválido. 2) Uma mensagem de erro é exibida e a conta não é adicionada.
Pré-condições	1) Estar logado na aplicação.
Pós-condições	1) A nova conta será inserida.

Na Tabela 4.6 está especificado o processo de remoção de uma conta. Após acessar a tela de início, o usuário deve pressionar e segurar alguma das contas listadas, e surgirá na tela uma caixa de texto para confirmar a exclusão da conta. Após confirmar a exclusão, a tela de Início atualiza e a conta excluída não é mais exibida na lista de contas.

Tabela 4.6: UC006: Remover Conta

Ator	Usuário
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de Início; 3) Clicar e segurar em alguma conta, e então confirmar a exclusão.
Fluxo alternativo	1) Clicar e segurar em alguma conta, e então cancelar a exclusão. 2) A exclusão é a cancelada e o usuário retorna à tela de Início.
Pré-condições	1) Estar logado na aplicação.
Pós-condições	1) A conta selecionada será removida.

Na Tabela 4.7 está especificado o processo para visualizar o perfil. Na tela de perfil, é possível ver o e-mail do usuário logado, a quantidade de contas configuradas e uma breve seção informando o funcionamento do aplicativo.

Tabela 4.7: UC007: Visualizar Perfil

Ator	Usuário
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de Perfil.
Fluxo alternativo	
Pré-condições	1) Estar logado na aplicação.
Pós-condições	1) O usuário poderá visualizar a tela de perfil.

Na Tabela 4.8 está especificado o processo de Logout. Após acessar a tela de perfil, o usuário pode desconectar a conta logada no aplicativo para retornar à tela de *login*, onde pode realizar um novo Cadastro ou *login* com outra conta.

Na Tabela 4.9 está especificada a listagem das contas configuradas. O usuário é direcionado à essa tela após realizar o processo de cadastro ou *login* e pode acessá-la também pela opção Início no menu. Cada conta possui um nome e o código de autenticação gerado pelo sistema.

Tabela 4.8: UC008: Fazer Logout

Ator	Usuário
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de Perfil; 3) Clicar no botão Sair.
Fluxo alternativo	
Pré-condições	1) Estar logado na aplicação.
Pós-condições	1) O usuário retornará à tela de *login*.

Tabela 4.9: UC009: Listar Contas Cadastradas

Ator	Sistema
Fluxo principal	1) Abrir o aplicativo; 2) Acessar a tela de Início.
Fluxo alternativo	
Pré-condições	1) Deve haver um usuário logado na aplicação.
Pós-condições	1) O sistema listará as contas configuradas e códigos do usuário logado.

Na Tabela 4.10 está especificado o processo realizado pelo sistema para atualizar os códigos únicos de autenticação, executado a cada 30 segundos. O processo é realizado apenas quando há um usuário logado.

Tabela 4.10: UC010: Atualizar códigos

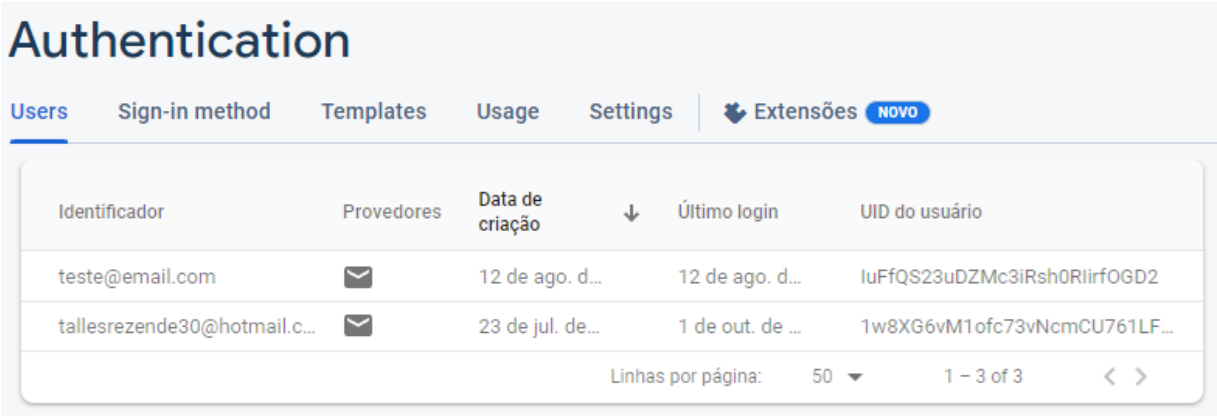
Ator	Sistema
Fluxo principal	1) Abrir o aplicativo; 2) Atualizar os códigos de autenticação.
Fluxo alternativo	
Pré-condições	1) Deve haver um usuário logado na aplicação.
Pós-condições	1) Novos códigos substituirão os anteriores para todas as contas.

4.3 Banco de Dados

O processo de cadastro e *login* de usuários é realizado por meio do Authentication, uma solução para serviços de autenticação oferecida pelo Firebase. O método utilizado no aplicativo desenvolvido neste trabalho é o *login* por e-mail e senha.

Na Figura 4.2 é possível observar a listagem de usuários ao acessar o console do Firebase.

Figura 4.2: Authentication - Listagem de Usuários.



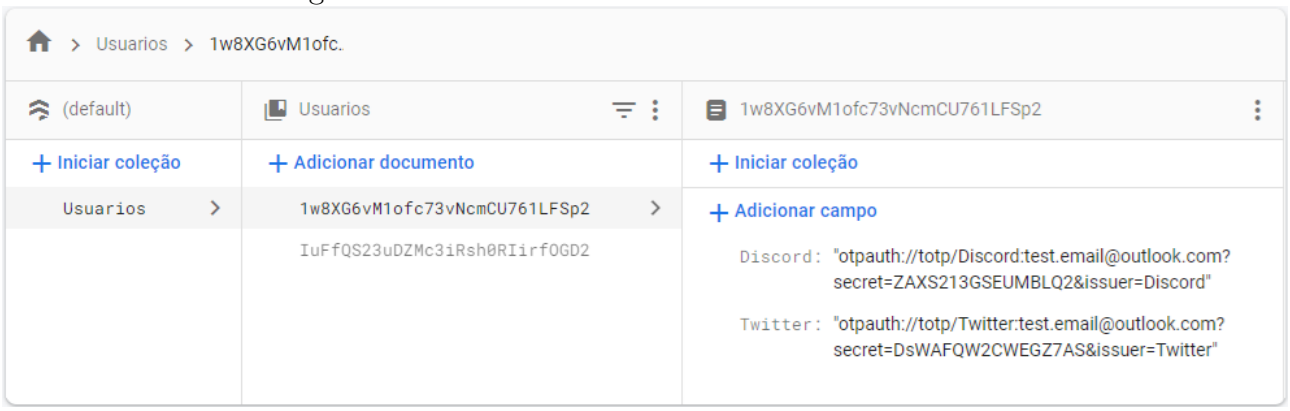
Fonte: Elaborado pelo autor.

Os dados exibidos são o identificador (no método utilizado, o e-mail do usuário), o provedor, as datas de criação e do o último *login*, e por fim um UID de usuário. Este UID é um identificador único gerado pelo próprio Firebase no momento da criação da conta.

Para a armazenagem das contas configuradas de cada usuário, foi utilizado o Cloud Firestore, um banco de dados NoSQL do Firebase que pode ser acessado diretamente por SDKs nativos e armazena os dados na nuvem.

Na Figura 4.3 é possível visualizar a estrutura de um banco de dados do Firestore, que conta com coleções, documentos e campos.

Figura 4.3: Firestore - Estrutura do Banco de Dados.



Fonte: Elaborado pelo autor.

O banco do aplicativo possui apenas uma coleção nomeada Usuarios. Assim que um usuário cadastrado finaliza a adição de uma conta para autenticação no aplicativo, um documento com seu UID (gerado pelo Authentication) é criado.

Em seguida, um novo campo é inserido neste documento, contendo o nome da conta adicionada (o nome do serviço em que foi ativada a autenticação por dois fatores) e seu código TOTP completo, no formato utilizado pelo Google Authenticator.

Caso o mesmo usuário adicione outra conta, um novo campo é adicionado ao documento com seu UID, preservando as contas configuradas anteriormente no mesmo documento. Se uma conta for removida pelo aplicativo, seu campo também é excluído do documento.

4.4 Geração de códigos TOTP

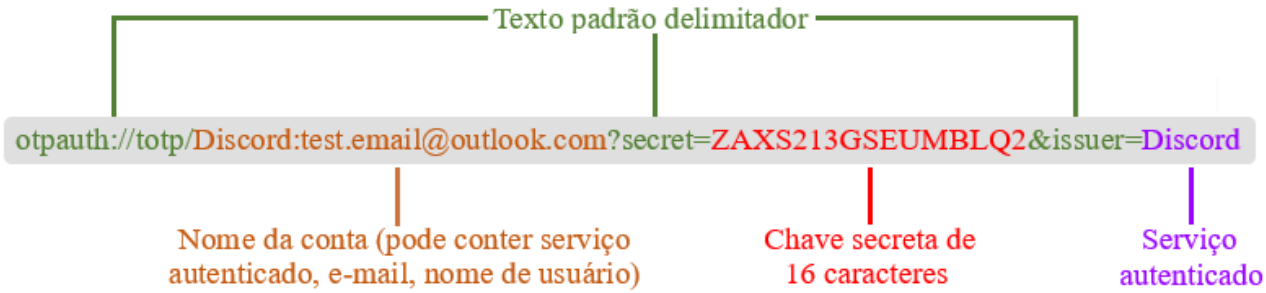
Para configurar uma nova conta para autenticação, o usuário deve primeiro obter uma chave secreta, fornecida pelo serviço autenticado. Essa chave é uma combinação de 16 letras maiúsculas e números, e apesar de alguns serviços exibirem o código para que o próprio usuário copie e insira no aplicativo autenticador, normalmente ele é transformado em um QR Code.

A captura do QR Code requer a instância de um objeto responsável por inicializar a câmera traseira do celular e exibir em tempo real na tela o que está sendo capturado. Assim que o usuário selecionar a opção de adicionar a conta pelo QR Code, esse objeto é inicializado e ativa a câmera. No momento em que um QR Code é detectado pela câmera, o objeto retorna em formato de texto simples o conteúdo do código capturado, encerrando a captura de vídeo.

O QR Code, assim como um código de barras, é apenas uma representação de um texto. No entanto, os códigos gerados para esse propósito normalmente contém, além da chave, informações adicionais para automatizar a etapa de configuração da conta para o usuário.

Os códigos seguem o formato observado anteriormente na Figura 4.3. Este formato, popularizado pelo Google Authenticator, contém um nome para a conta autenticada, a chave secreta e o provedor do serviço a ser autenticado. Entre esses elementos, existem caracteres delimitadores que permitem sua separação, como pode ser visto na Figura 4.4.

Figura 4.4: Estrutura de um QR Code TOTP.



Fonte: Elaborado pelo autor.

Após a captura do conteúdo do QR Code ou da inserção manual das informações, o código é armazenado na base de dados e vinculado ao usuário. A partir de então, o aplicativo passa a processar aquele código, de forma a transformá-lo em um número inteiro legível pelo usuário, como descrito anteriormente na Seção 2.5.3.

Esse processamento é realizado por meio de um algoritmo TOTP, que requer dois parâmetros: um fator em movimento e uma chave. A chave é a cadeia de 16 caracteres que pode ser extraída do código armazenado na base dados por RegEx (expressão regular), pois é delimitada pelos termos "?secret=" à esquerda e "&issuer=" à direita, como demonstrado na Figura 4.4.

Já o fator em movimento é o tempo do Unix (representa o número de segundos decorridos desde 01/01/1970 às 00:00 UTC) dividido pelo tempo em que cada código temporário deve ser válido (normalmente 30 segundos), e então arredondado para um valor inteiro.

O algoritmo TOTP alimenta uma função criptográfica conhecida como HMAC-SHA-1, que recebe uma mensagem e uma chave, e retorna um *hash* de 160 *bits* (ou 20 *bytes*). Para reduzir seu tamanho, o *hash* é truncado, e finalmente convertido em um número inteiro de seis dígitos.

Na Figura 4.5 a seguir é possível observar um pseudocódigo comentado que representa o algoritmo TOTP e detalha seu funcionamento.

Figura 4.5: Pseudocódigo: Geração de Códigos TOTP.

```
Funcao TOTP(String chave, int tempoUnix) {

      //Decodifica a chave em Base32, transformando-a em um array de bytes
      Byte[] chaveBase32 = base32decode(chave);

      //Divide o tempo do Unix por 30 e o arredonda para baixo
      int tempo = arredonda(tempoUnix / 30);

      //Executa a função HMAC SHA1 que gera o hash, um array de bytes
      byte[] hash = hmacSha1(chaveBase32, tempo);

      //Obtém o valor do byte na última posição do hash
      byte ultimoByte = hash[19];

      //Define o offset (converte o valor dos últimos 4 bits para inteiro
      int offset = ultimoByte[5..8];

      //Captura 4 dos 20 bytes do hash gerado, iniciando pela posição do offset
      byte[] hashTruncado = hash[offset..offset+3]

      //Altera o primeiro bit do primeiro byte para zero, garantindo que após
      //converter para inteiro, o número será sempre positivo
      hashTruncado[0][0] = 0;

      //Divide o valor como inteiro do hash por 10⁶ e obtém o resto,
      //garantindo que o resultado tenha no máximo 6 dígitos
      int codigoInt = hashTruncado mod 1000000;

      //Converte para string e adiciona zeros até atingir 6 caracteres
      String codigoOtp = PadLeft(String(codigoInt),"0",6);

      //Retorna o código final que é exibido ao usuário
      return codigoOtp;
}
```

Fonte: Elaborado pelo autor.

Este algoritmo é executado a cada 30 segundos, o intervalo necessário para que o algoritmo produza um novo código para uma mesma chave. Para isso, deve ser implementada uma *thread*: uma sequência de instruções executada em paralelo com a execução do código principal.

A execução em paralelo do algoritmo TOTP permite que a atualização do código ocorra precisamente a cada 30 segundos sem que seja necessário adicionar esse intervalo de espera na linha de execução principal, o que travaria outros elementos da aplicação.

Além do algoritmo TOTP, é possível implementar da mesma maneira uma referência visual que indique ao usuário em quanto tempo ocorrerá a atualização dos códigos, como uma barra de progresso ou um temporizador.

4.5 Considerações Finais

Este capítulo detalhou o planejamento do projeto e a metodologia utilizada em sua implementação, utilizando casos de uso para descrever as funções do sistema e entender o processo de cada uma. Tanto o planejamento da estrutura do banco de dados quanto o entendimento do algoritmo de geração de códigos TOTP foram essenciais para o desenvolvimento da aplicação.

Capítulo

5

Resultados

Este capítulo tem como objetivo mostrar a aplicação após a implementação, detalhando suas funcionalidades, exibindo suas telas e relacionando o planejamento com os resultados obtidos.

5.1 Apresentação do sistema

A aplicação possui 5 telas: *login*, cadastro, início, nova conta e informações. Além disso, as funções para redefinir a senha de usuário e remover uma conta escurecem a tela de fundo e abrem uma janela *pop-up* em destaque.

5.1.1 Login

Ao abrir o aplicativo pela primeira vez, o usuário visualiza a tela de *login*, que pode ser observada na Figura 5.1(a). Por meio dela, o usuário pode realizar o processo de *login* descrito no Caso de Uso UC002, mostrado na Tabela 4.2.

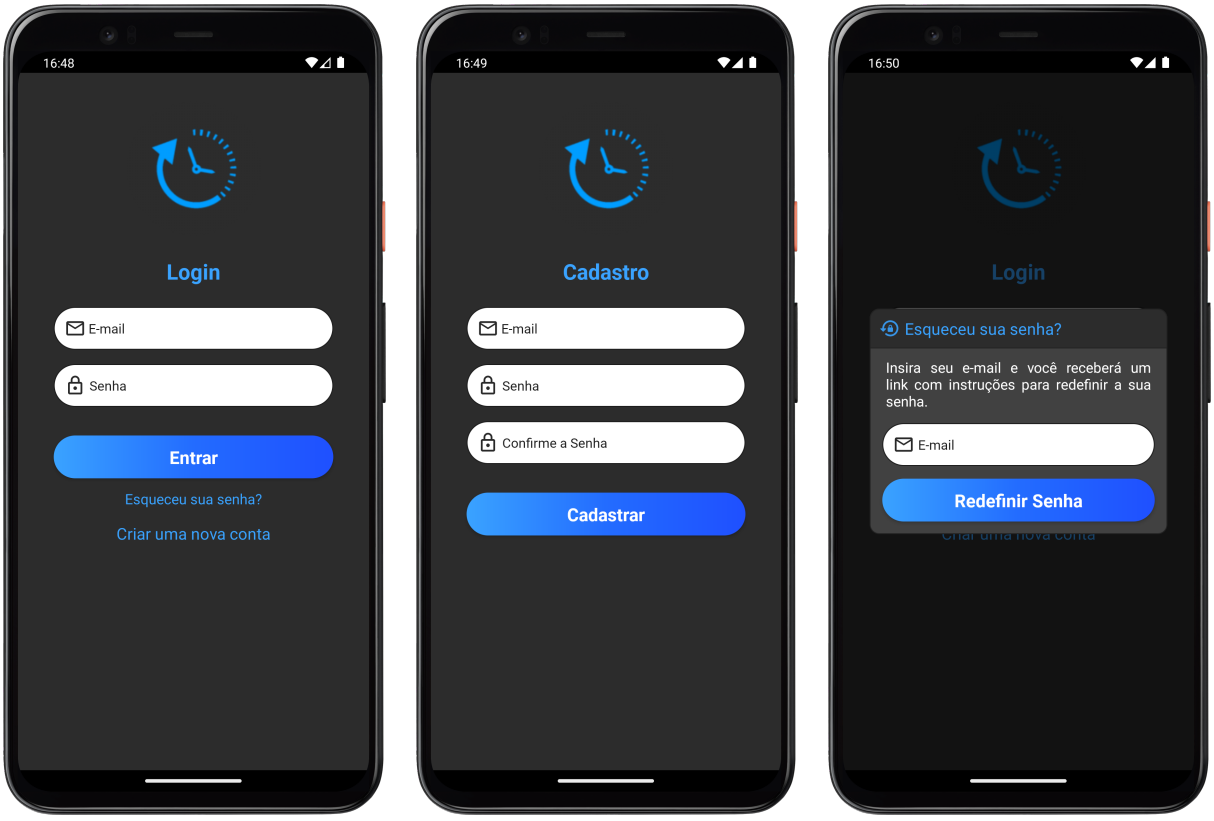
5.1.2 Cadastro

Para se cadastrar, o usuário deve clicar em ”Criar uma nova conta” na tela de *login*. Na tela de cadastro, exibida na Figura 5.1(b), é possível criar uma nova conta de usuário, processo detalhado no Caso de Uso UC001 mostrado na Tabela 4.1. Após inserir seus dados e finalizar o cadastro, o usuário se mantém logado e é direcionado à tela de Início.

5.1.3 Redefinição de Senha

Ainda na tela de *login*, o usuário pode redefinir a sua senha, como descrito no Caso de Uso UC003, mostrado na Tabela 4.3. Ao clicar em ”Esqueceu sua senha?”, um *pop-up* é aberto para que o usuário insira seu e-mail, em que receberá um *link* para redefinir a senha. A janela de redefinição de senha pode ser visualizada na Figura 5.1(c).

Figura 5.1: Telas de *login*, cadastro e janela de redefinição de senha.



(a) Login (b) Cadastro (c) Redefinição de Senha

Fonte: Elaborado pelo autor.

5.1.4 Listagem de contas

Após realizar o *login*, o aplicativo exibe a tela de Início, que lista todas as contas configuradas do usuário. Cada conta é composta por um nome e um código para autenticação. A listagem de contas foi descrita no Caso de Uso UC009 e detalhada na tabela 4.9.

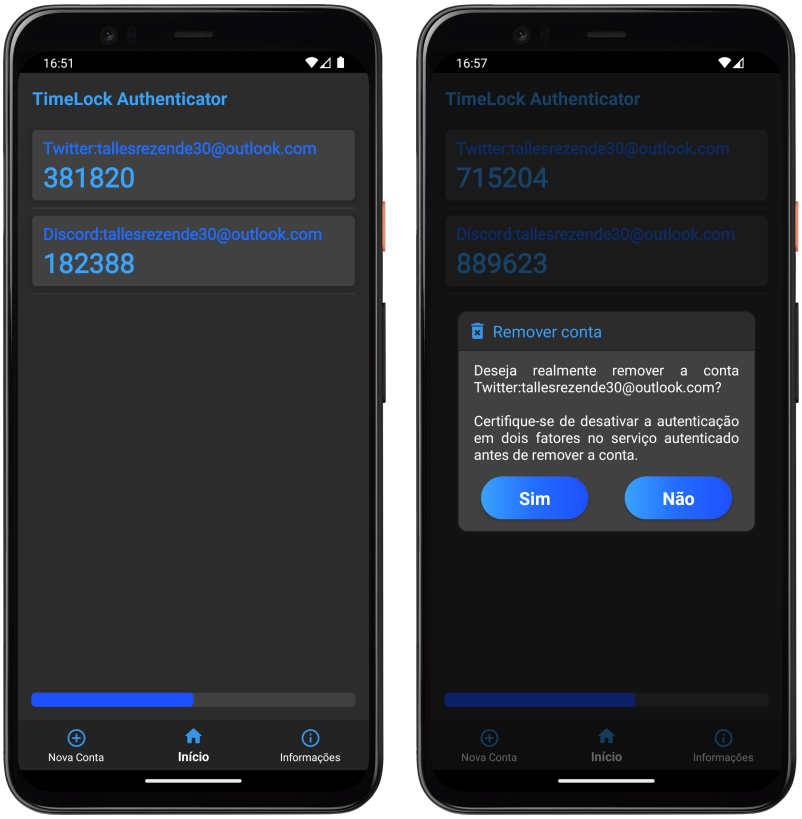
A tela também possui um menu inferior e pode ser observada na Figura 5.2(a). Acima do menu há uma barra de progresso, uma referência visual do processo de atualização dos códigos, descrito no Caso de Uso UC010 e mostrado na Tabela 4.10. A barra é preenchida a cada segundo, e após 30 segundos retorna ao estado inicial, momento em que os códigos são atualizados.

5.1.5 Removendo uma Conta

Ainda na tela de Início, o usuário pode remover uma conta mantendo o dedo pressionado em alguma das contas listadas. Uma janela se abrirá para que o usuário confirme se deseja ou não excluir a conta selecionada, e ao clicar em Sim, a conta é removida permanentemente.

É importante que haja uma etapa de confirmação neste processo, pois o serviço autenticado normalmente solicita um último código de autenticação gerado pelo aplicativo para desativar a autenticação em dois fatores. Este processo foi descrito no Caso de Uso UC006, mostrado na Tabela 4.6, e pode ser observado na Figura 5.2(b).

Figura 5.2: Tela de início e janela de remoção de contas.



(a) Início (b) Remover Conta

Fonte: Elaborado pelo autor.

5.1.6 Adicionando uma Nova Conta

Ao clicar em ”Nova Conta” no menu inferior na tela de Início, o usuário é direcionado à tela exibida na Figura 5.3(a). Por meio dela, é possível adicionar uma nova conta para autenticação de duas formas. A primeira delas, descrita no Caso de Uso UC004 e na Tabela 4.4, é por meio do botão ”Ler QR Code”, que utiliza a câmera do celular para capturar um QR Code.

Assim que o botão para ler o código é pressionado, a tela exibe a imagem capturada pela câmera do celular em tempo real, até que identifique um QR Code. Quando isto acontece, o aplicativo valida o conteúdo do QR Code escaneado, que deve conter um texto no formato descrito na Figura 4.4. Se o QR Code for validado, a adição da conta é finalizada e o usuário é redirecionado à tela de Início, onde pode visualizar a conta que adicionou.

A segunda maneira de configurar uma conta foi detalhada no Caso de Uso UC005, mostrado na Tabela 4.5. Este método consiste na inserção manual do nome da conta, do e-mail e do código de 16 dígitos nos campos da parte inferior da tela de Nova Conta.

Ao clicar no botão ”Adicionar Conta”, as informações são organizadas em um texto único e assim como no primeiro método, este texto é validado para verificar se é possível gerar um código de 6 dígitos a partir dele. Caso passe na verificação, a configuração da conta é finalizada e a aplicação retorna à tela de Início.

5.1.7 Informações

Selecionando o item ”Informações” no menu inferior, a aplicação exibe a tela mostrada na Figura 5.3(b) e descrita no Caso de Uso UC007, mostrado na Tabela 4.7. Esta tela informa, na parte superior, o e-mail do usuário logado e a quantidade de contas configuradas.

Pelo botão Sair é possível fazer *logout*, processo descrito no Caso de Uso UC008 e detalhado na Tabela 4.8, no qual o usuário se desconecta e retorna à tela de *login*. Na parte inferior da tela também há uma seção informativa que explica brevemente ao usuário o propósito do aplicativo e o orienta sobre como configurar a autenticação em dois fatores.

Figura 5.3: Telas de adição de contas e informações

(a) Nova Conta (b) Informações

Fonte: Elaborado pelo autor.

5.2 Considerações finais

Neste capítulo foram mostradas as telas da aplicação desenvolvida e a forma que o usuário pode navegar entre elas. Também foi possível visualizar os resultados finais dos casos de uso projetados no Capítulo 4.

Capítulo

6

Conclusões

6.1 Considerações Finais

Como descrito inicialmente no Capítulo 1, é importante que pessoas que utilizam a Internet e possuem contas *online* entendam que uma única senha pode não garantir sua segurança, e busquem formas e incrementá-la.

É possível concluir que os objetivos principais do projeto foram atingidos, pois o aplicativo desempenha todas as funções propostas, como a captura do conteúdo de QR Codes válidos, a geração de códigos que expiram após um tempo determinado e a sua atualização automática. A geração dos códigos se provou efetiva pois foi possível ativar a autenticação em dois fatores pelo aplicativo em contas de diversos sites e aplicativos testados, como Discord e Twitter.

Todas as etapas do desenvolvimento deste projeto foram essenciais para sua conclusão: desde o estudo de conceitos e fundamentos que formam a base da aplicação e da análise de soluções semelhantes já disponíveis, até o planejamento e a eventual implementação do aplicativo.

O desenvolvimento em Android nativo garante que a maioria das pessoas que possuem um *smartphone* possa utilizar o aplicativo, assim como também garante que a aplicação tenha um ótimo desempenho e fácil acesso aos recursos de hardware.

O banco de dados utilizado também garante um bom desempenho, devido à sua simplicidade, ao mesmo tempo que possui o necessário para que os dados da aplicação sejam atualizados de forma instantânea e transparente para o usuário.

6.2 Trabalhos Futuros

Possíveis trabalhos futuros para a incrementação do aplicativo desenvolvido neste projeto incluem a adição de compatibilidade para iPhone, a possibilidade de exportar e importar contas configuradas por meio de arquivos de *backup*, e adicionar uma seção de ajuda destinada à outros desenvolvedores, pela qual possam se orientar para implementar em suas próprias aplicações a compatibilidade com aplicativos autenticadores TOTP.

Referências Bibliográficas

AINSWORTH, B. *One-Time Password (OTP) Authentication Methods - HOTP + TOTP*. 2022. Disponível em: <https://www.sharetru.com/blog/one-time-password-otp-authentication-methods-you-should-know-hotp-totp>. Acesso em 25 de maio de 2023.

ALVES, W. *Java programming*. [S.l.]: Saraiva, 2014.

AUTHY (Ed.). *Authy vs. Google Authenticator*. 2017. Disponível em: <https://authy.com/blog/authy-vs-google-authenticator>. Acesso em 25 de maio de 2023.

AUTHY (Ed.). *Authy - Two Factor Authentication*. 2023. Disponível em: <https://authy.com>. Acesso em 25 de maio de 2023.

AUTHY (Ed.). *Twilio Authy Authenticator*. 2023. Disponível em: <https://play.google.com/store/apps/details?id=com.authy.authy>. Acesso em 25 de maio de 2023.

AUTHY (Ed.). *What Is Two-Factor Authentication (2FA)?* 2023. Disponível em: <https://authy.com/what-is-2fa/>. Acesso em 25 de maio de 2023.

AWS (Ed.). *O que é Java?* 2023. Disponível em: <https://aws.amazon.com/pt/what-is/java/>. Acesso em 25 de maio de 2023.

BOCARD, T. *O que são aplicativos? Definição da desenvolvedora Usemobile*. 2021. Disponível em: <https://usemobile.com.br/aplicativo-movel/>. Acesso em 25 de maio de 2023.

CISA (Ed.). *MULTI-FACTOR AUTHENTICATION*. 2022. Disponível em: <https://www.cisa.gov/resources-tools/resources/multifactor-authentication-mfa>. Acesso em 25 de maio de 2023.

DACRE, S. *Native vs Hybrid Apps: What Will Work for Me?* 2019. Disponível em: <https://freshworks.io/native-vs-hybrid-app/>. Acesso em 25 de maio de 2023.

FARRELL, J. *Java programming*. [S.l.]: Cengage Learning, 2022.

GOOGLE (Ed.). *Cloud Firestore*. 2023. Disponível em: <https://firebase.google.com/docs/firestore?hl=pt-br>. Acesso em 25 de maio de 2023.

GOOGLE (Ed.). *Cloud Storage para Firebase*. 2023. Disponível em: <https://firebase.google.com/docs/storage?hl=pt-br>. Acesso em 25 de maio de 2023.

GOOGLE (Ed.). *Firebase Authentication*. 2023. Disponível em: <https://firebase.google.com/docs/auth?hl=pt-br>. Acesso em 25 de maio de 2023.

GOOGLE (Ed.). *Firebase Products*. 2023. Disponível em: <https://firebase.google.com/products-build?hl=pt-br>. Acesso em 25 de maio de 2023.

GOOGLE (Ed.). *Firebase Realtime Database*. 2023. Disponível em: <https://firebase.google.com/docs/database?hl=pt-br>. Acesso em 25 de maio de 2023.

GOOGLE (Ed.). *Google Authenticator*. 2023. Disponível em: <https://apps.apple.com/us/app/google-authenticator/id388497605>. Acesso em 25 de maio de 2023.

HOLOVKO, O. *Mobile Apps in 2023: To develop or not to develop*. 2022. Disponível em: <https://www.promodo.com/blog/mobile-apps-in-2023-to-develop-or-not-to-develop>. Acesso em 25 de maio de 2023.

IBM (Ed.). *Tipos de aplicativos móveis*. 2023. Disponível em: <www.ibm.com/docs/pt-br/maas360?topic=apps-mobile-app-typesl>. Acesso em 25 de maio de 2023.

ID, C. *30 milhões de credenciais foram vazadas no Brasil em 2022, revela levantamento de ISH e SafeLabs*. 2023. Disponível em: <https://cryptoid.com.br/criptografia-identificacao-digital-id-biometria/30-milhoes-de-credenciais-foram-vazadas-no-brasil-em-2022-revela-levantamento-de-ish-e- Acesso em 12 de outubro de 2023.

KEMP, S. *Digital 2023: Global Overview Report*. 2023. Disponível em: <https://datareportal.com/reports/digital-2023-global-overview-report>. Acesso em 12 de outubro de 2023.

LAU, G. *Native vs Hybrid: Which Mobile App Platform Should You Choose?* 2022. Disponível em: <https://www.codemotion.com/magazine/frontend/mobile-dev/native-vs-hybrid-which-mobile-app-platform-should-you-choose/>. Acesso em 25 de maio de 2023.

LEDU, E. E. (Ed.). *Native vs. cross-platform app development: pros and cons*. 2018. Disponível em: <https://codeburst.io/native-vs-cross-platform-app-development-pros-and-cons-49f397bb38ac>. Acesso em 25 de maio de 2023.

MICROSOFT (Ed.). *Microsoft Authenticator*. 2023. Disponível em: <https://play.google.com/store/apps/details?id=com.azure.authenticator>. Acesso em 25 de maio de 2023.

MICROSOFT (Ed.). *Microsoft Authenticator*. 2023. Disponível em: <https://apps.apple.com/br/app/microsoft-authenticator/id983156458>. Acesso em 25 de maio de 2023.

M'RAIHI, D. et al. *HOTP: An HMAC-Based One-Time Password Algorithm*. 2005. Disponível em: <https://datatracker.ietf.org/doc/html/rfc4226>. Acesso em 25 de maio de 2023.

M'RAIHI, D. et al. *TOTP: Time-Based One-Time Password Algorithm*. 2011. Disponível em: <https://datatracker.ietf.org/doc/html/rfc6238>. Acesso em 25 de maio de 2023.

NEOWAY (Ed.). *Segurança da informação: O que é, e seus impactos e soluções nas empresas*. 2020. Disponível em: <https://blog.neoway.com.br/seguranca-da-informacao>. Acesso em 25 de maio de 2023.

ORACLE (Ed.). *What is Java technology and why do I need it?* 2022. Disponível em: <https://www.java.com/en/download/help/whatis_java.html>. Acesso em 25 de maio de 2023.

ORACLE (Ed.). *O que é NoSQL?* 2023. Disponível em: <https://www.oracle.com/br/database/nosql/what-is-nosql/>. Acesso em 25 de maio de 2023.

RIBEIRO, A. L. S. *Entendendo o Firebase e suas principais funcionalidades*. 2023. Disponível em: <https://www.alura.com.br/artigos/entendendo-firebase-principais-funcionalidades>. Acesso em 25 de maio de 2023.

RUBLON (Ed.). *HOTP vs TOTP: What's the Difference?* 2022. Disponível em: <https://rublon.com/blog/hotp-totp-difference>. Acesso em 25 de maio de 2023.

SANTOS, N. *Principais métodos de roubos de senhas*. 2022. Disponível em: <https://www.sec4u.com.br/principais-metodos-de-roubos-de-senhas>. Acesso em 25 de maio de 2023.

SILBERSCHATZ, A. *Java programming*. [S.l.]: GEN LTC, 2020.

SILVA, G. *O que é Firebase?* 2022. Disponível em: <https://coodesh.com/blog/dicionario/o-que-e-firebase>. Acesso em 25 de maio de 2023.

SMITH, N. *HOTP vs TOTP: What's the Difference?* 2018. Disponível em: <https://www.microcosm.com/blog/hotp-totp-what-is-the-difference>. Acesso em 25 de maio de 2023.

STATS, G. (Ed.). *Mobile Operating System Market Share Worldwide - February 2023*. 2023. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202302-202302-bar>. Acesso em 25 de maio de 2023.

STRICKLAND, F. *Estudo aponta 1,6 bilhão de casos de roubo de dados pessoais na internet*. 2021. Disponível em: <https://www.correiobraziliense.com.br/brasil/2021/06/4928596-estudo-aponta-16-bilhao-de-casos-de-roubo-de-dados-pessoais-na-internet.html>. Acesso em 25 de maio de 2023.

TUTIDA, D. *O que é segurança da informação? Saiba como garantir a sua*. 2021. Disponível em: <https://encontreumnerd.com.br/blog/o-que-e-seguranca-da-informacao>. Acesso em 25 de maio de 2023.

TWILIO (Ed.). *Twilio Acquires Authy*. 2015. Disponível em: <https://www.twilio.com/press/releases/twilio-acquires-authy-to-accelerate-strong-authentication-and-identity-adoption-for-web
Acesso em 25 de maio de 2023.

WELLINGTON, H. *Native Apps vs Hybrid Apps Comparison*. 2021. Disponível em: <https://saucelabs.com/resources/blog/native-apps-vs-hybrid-apps-comparison>. Acesso em 25 de maio de 2023.