

# SuperID - Projeto Integrador 3

Fevereiro de 2025 - Versão 1.0

**Autores deste documento:**

**Prof. Mateus Dias;**

**Profa. Renata Arantes;**

**Prof. Luã Marcelo**

Pontifícia Universidade Católica de Campinas

Curso de Engenharia de Software

## Visão geral

O projeto integrador deste semestre consiste em desenvolver um ecossistema chamado gerenciador de autenticações. Isso significa desenvolver diversos processos que vão desde a criação de uma conta, armazenamento seguro de senhas e finalmente o uso das credenciais para realizar um login.

A complexidade desse assunto foi reduzida de forma que a concentração do aluno seja em aprender a construir um aplicativo para smartphones e sua integração com um backend na Internet. Este projeto tem finalidades educacionais, logo, não atende aos padrões mais exigentes de segurança da informação.

Com o objetivo de ter uma visão mais técnica, misturada com os requisitos funcionais, conforme as equipes evoluírem no conhecimento técnico sobre a linguagem Kotlin, Android e Google Firebase, se tornará mais fácil a compreensão e entendimento de todas os requisitos para a implementação.

Esse sistema conta com duas partes principais:

- Aplicativo Mobile (Android, desenvolvido em Kotlin), utilizado para a gestão e armazenamento seguro de credenciais do usuário final da solução (cliente final).
- Integração Web (via API e Firebase Functions), permitindo que sites parceiros utilizem o SuperID como método de login sem senha, além de outros autenticadores como Google etc

## Objetivos

1. Aprender a desenvolver um aplicativo nativo em Android com Kotlin;
2. Aprender como usar a nuvem da Google (Google Cloud Platform) e seus serviços como Firebase Firestore, Firebase Functions, Firebase Storage e outros;
3. Trabalhar com um tema fascinante que é a segurança da informação.

## Requisitos funcionais baseados em jornadas

Separamos nesta seção os requisitos de uma forma sequencial em lista para ficar fácil o entendimento e a implementação deste aplicativo.

## RF1 - Criando a conta (usando o aplicativo pela primeira vez- Cadastro Inicial)

- O usuário abre o aplicativo e recebe uma breve explicação sobre o que é o SuperID. Essa breve explicação deverá ser implementada como a equipe desejar. Pode ser um tour, um vídeo ou um texto simples. Evidentemente, quanto mais útil e caprichado for, melhor.
- Em seguida, o usuário deve aceitar os termos de uso (a serem definidos pela equipe).
- O cadastro requer: Nome, Email e Senha Mestre.
- O email deve ser validado via Firebase Authentication.
- O aplicativo salva o UID e IMEI do dispositivo no Firebase Firestore.

### Explicação detalhada dessa jornada:

Na primeira vez que o usuário abrir o aplicativo, haverá um pequeno texto informando ao usuário que o SuperID é um jeito inovador de fazer login sem usar senhas e também serve para armazenar as suas senhas tradicionais de maneira segura. Então, o usuário deve concordar com os termos de uso (sua equipe deve elaborar os termos) e prosseguir com a criação da conta.

Criar uma conta consiste em: Informar um Nome, Email e uma Senha mestre. Ao informar esses dados, será criada uma conta remotamente num serviço da Google chamado Firebase Authentication. Para criar a conta no firebase via código kotlin no Android, é necessário informar: nome, email e a senha mestre. Note que neste momento é necessário usar um próprio recurso do Firebase Auth para VALIDAR se o email do usuário é real. Se o usuário não validar o email não conseguirá usar a funcionalidade Login Sem Senha, mas poderá usar as outras.

Na etapa de criação da conta, imediatamente ao salvar os dados no Firebase Authentication da Google, o aplicativo também deverá salvar um DOCUMENTO contendo dados como UID e IMEI do aparelho no Firebase Firestore (que é o banco de dados do Firebase orientado a documentos).

Ao criar a conta, o usuário estará apto a seguir com o uso do app.

## RF2 - Manter banco de dados de senhas de acesso (Cadastrar, alterar e excluir) - Gerenciamento de senha

- Após o login, o usuário pode cadastrar, alterar e excluir senhas pessoais (usar o app como um gerenciador de senhas pessoais).
- As senhas são organizadas em categorias (ex.: Sites Web, Aplicativos, Teclados de Acesso Físico).
- O aplicativo gera um accessToken de 256 caracteres (Base64) para cada senha.
- As senhas devem ser criptografadas antes do armazenamento (definição da criptografia a ser utilizada é de responsabilidade da equipe). Portanto, cabe a cada equipe estudar os modelos de criptografia existentes e adotar um. Vamos salientar que existem criptografias simétricas e assimétricas com vários algoritmos. Cada equipe deverá estudar e escolher uma solução. Portanto, nenhum professor orientador definirá ou ajudará qualquer equipe nessa escolha ou nesse tema.

### Explicação detalhada dessa jornada:

Uma vez que o usuário terminou de criar a conta ou voltou no aplicativo e fez o login de acesso com a senha cadastrada (usando o Firebase Auth), ele poderá cadastrar/alterar/excluir senhas de acesso em serviços, aplicativos e para tudo que ele desejar.

Essas senhas serão organizadas em categorias. Por exemplo: Sites da Web, Aplicativos locais, Num pads (aqueles teclados de acesso que o usuário precisa digitar num teclado numérico para abrir uma porta). As categorias podem ser criadas pelo usuário. No entanto, o aplicativo já vem cadastrado previamente com as seguintes categorias por padrão: Sites Web, Aplicativos e Teclados de Acesso Físico. A categoria Sites Web não poderá ser excluída, é obrigatória e padrão no aplicativo.

Quando o usuário escolher cadastrar uma nova senha, deverá informar o login (pode ser opcional) e a senha de acesso. Afinal de contas, existem sistemas físicos por exemplo do tipo Numpad não necessitam de uma credencial, é apenas a senha numérica (como o caso de teclados de acesso físico) e que o usuário apenas quer se lembrar.

Em seguida informar uma descrição (opcional) e escolher salvar.

Neste momento essa senha deverá ser guardada no documento que representa a conta do usuário no Firebase Firestore (numa coleção, subcoleção, enfim, tecnicamente a sua equipe terá que descobrir como) de maneira criptografada (a forma de criptografia deve ser estudada pela

sua equipe e não será objeto de dúvidas em orientações - os professores não vão sanar dúvidas de como cifrar e decifrar senhas ou quais algoritmos criptográficos deverão ser usados e porquê. Sua equipe tem a missão de conhecer os tipos de criptografia mais convencionais e aprender a utilizá-los em sistemas de software).

Sempre que for cadastrada uma senha, o aplicativo deverá gerar uma string aleatória que chamaremos de token com 256 caracteres em Base64. Ou seja, essa string faz parte do cadastro da senha e o nome desse campo no documento do firestore será: accessToken.

Todas as vezes que uma senha for utilizada para autenticação em algum site, esse accessToken deverá ser alterado e isso será explicado nos requisitos pertinentes a seguir.


### RF3 - Usuário acessa um site parceiro para fazer o Login sem Senha

- Um site parceiro oferece a opção de login via SuperID e o usuário tem conta no SuperID
- O site faz uma requisição para a Firebase Function performAuth, recebendo um QRCode (com um loginToken).
- O usuário escaneia o QRCode pelo aplicativo, confirmando a autenticação.
- O site pode consultar a Firebase Function getLoginStatus para verificar se o login foi concluído ou não

#### Explicação detalhada dessa jornada:

Quando o usuário acessar o aplicativo terá uma opção visível (um botão) com uma espécie de ícone de qr code que é o login sem usar a senha específica daquele site. Se o site que o usuário deseja fazer login for parceiro da SuperID, ou seja, que aceite logins por esse método, no site em si, o usuário escolherá que quer fazer o login usando o SuperID (como se fosse mais uma opção além daquelas como fazer login com o Google, com Facebook etc), não preencher o email e apenas pressionar um botão continuar.

Nesse momento o site parceiro fará uma requisição http numa função (Firebase Function) chamada **performAuth** e enviará na requisição seu próprio endereço, exemplo [www.xptositeweb.com.br](http://www.xptositeweb.com.br) e um token chamado API Key que sempre será restrito, exclusivo e secreto para cada site parceiro da SuperID. Nesse momento a função cadastrará um documento numa coleção chamada **login** com os campos: apiKey do site, data e horário atual de criação desse documento e mais um novo campo chamado **loginToken** também de 256 caracteres (string gerada pela função performAuth) e o retorno da função deve ser uma imagem em Base64 que



simplesmente seja um QRCode (uma imagem) e seu conteúdo é apenas o conteúdo do campo loginToken.

Portanto, no banco de dados Firebase Firestore, existirá uma coleção de documentos chamada **partners** (já pré criada pela equipe de vocês) onde cada documento representa um site que é cliente da SuperID. Cada documento deve ter no mínimo: URL do site sem http ou https, raiz, por exemplo: [www.puc-campinas.edu.br](http://www.puc-campinas.edu.br) (sem / ou subdomínios - não poderá ter por exemplo nenhum parceiro como: login.empresaxpto.com.br) precisa começar com www e uma string de 128 caracteres em base64 chamada apiKey (chave de api) essa chave é exclusiva para cada parceiro e representa simplesmente um token de acesso e um email do profissional da empresa responsável pela segurança ou integração dos serviços.

Logo, o site receberá essa imagem em Base64 e converterá numa imagem pedindo que o usuário abra o aplicativo SuperID e escolha a opção login sem senha. Sempre que o usuário abrir o SuperID, será exigida a senha mestre de acesso se ele mantiver o app aberto, não será solicitada novamente. Então, o usuário com o celular na mão deverá colocar a senha mestre se estiver abrindo o app apenas para isso e escolher a opção login sem senha. Nesse momento o app abrirá a câmera para escanear a imagem. Ao ler o QRCode (depois que o usuário digitou a senha mestra) o aplicativo fechará a câmera e procurará na coleção login as informações referentes e associadas àquele campo loginToken. Encontrando aquele documento, será acrescentado naquele documento um campo user (onde o dado será o UID do Firebase Auth daquela conta de usuário) e a data hora do momento do login.

Existirá uma função chamada getLoginStatus que o site poderá requisitar em poucos segundos para saber “quem” usou aquele QR Code a critério dele para prosseguir com o fluxo de negócio específico do site. Para saber isso, o site deve informar na requisição: loginToken. Poderá fazer até 3 requisições usando aquele loginToken num período de até 1 minuto. Em seguida aquele token não pode ser mais consultado e será excluído caso o site tentar consultar por 3 vezes seguidas ou o tempo extrapolar 1 minuto até a consulta ser realizada e um NOVO QR CODE deverá ser gerado (mas, o site deverá requisitar novamente a função performAuth).

Sua equipe deve criar por exemplo um site estático de algum tema para SIMULAR essa situação de login sem senha.

## RF4 - Recuperação de senha Mestre


- Se o usuário esquecer a senha mestre, poderá redefini-la via email, utilizando os recursos do Firebase Auth.

- A troca de senha será possível apenas se o email tiver sido previamente validado - é importante que essa informação fique visível o tempo todo no aplicativo do smartphone caso a conta ainda não tenha sido validada.

Caso o usuário tente utilizar o aplicativo e não se lembre da senha mestre cadastrada, poderá trocá-la por email. O próprio Firebase Auth possui um recurso pronto para implementar este processo. Portanto ao abrir o app e o usuário não lembrar a senha mestre, poderá cadastrar outra (SE TIVER o email validado previamente). É importante também que o email cobre do usuário que valide a sua conta, senão poderá ficar sem esse recurso.

## Regras obrigatórias para implementação do projeto

1. O código-fonte será armazenado em um repositório no GitHub. (Todos os membros da equipe devem contribuir regularmente e não serão aceitas outras formas de comprovação de trabalho e produção)
2. As branches principais seguirão a estrutura: main, develop, feature/nome-da-funcionalidade.
3. Para entrega final, será criado um release no GitHub por meio do uso de TAGs
4. O projeto incluirá documentação no formato README.md com instruções de instalação e uso.
5. O código deve conter comentários explicativos para facilitar a compreensão.
6. As tarefas serão organizadas no GitHub na ferramenta Projects. Portanto, APONTE as horas trabalhadas Github project para ficar fácil no final do semestre entregar o relatório de atividades autônomas. É um componente curricular obrigatório.

- 
7. O time deverá ter reuniões periódicas para acompanhamento do progresso com o professor(a) do PI3.
  8. Nome do repositório no github deve ser igual ao nome do grupo que foi sorteado no CANVAS, obrigatoriamente. Não serão aceitos outros nomes.
  9. Professores no repositório do projeto no github. Permita o acesso dos professores Mateus , Luã e Renata no repositório do seu projeto no github para que possam fazer as avaliações. Emails: [mateus.dias@puc-campinas.edu.br](mailto:mateus.dias@puc-campinas.edu.br), [renata.arantes@puc-campinas.edu.br](mailto:renata.arantes@puc-campinas.edu.br) e [lua.marcelo@puc-campinas.edu.br](mailto:lua.marcelo@puc-campinas.edu.br)
  10. Banca avaliadora. A banca é obrigatória para todas as equipes (membros). Aqueles que faltarem no dia da banca por QUALQUER motivo, terá reprovação imediata na disciplina de projeto Integrador. A não ser que seja por algum motivo justificável, presente no regimento interno da instituição.

## Lista das tecnologias e ferramentas para o projeto

Git com Github (obrigatório);

Github Projects para controle de tarefas e esforço (obrigatório);

Android Studio com Kotlin (obrigatório);

Bancos de Dados: Firebase Firestore (obrigatório);

Gerenciador de autenticação e contas: Firebase Authentication (obrigatório);

Backend / APIs que usem Firebase Functions (obrigatório);

Illustrator, Photoshop e outras ferramentas de edição para artes, é livre;

## Data Final da Entrega

Release único e final - 01 de junho de 2025 no GitHub com a TAG - 1.0-Final



## Importante: Interpretação dos requisitos.

A interpretação do documento de visão do projeto integrador e os requisitos que constam nele, são de responsabilidade do Time. Leia muitas vezes o documento, interprete os requisitos com a sua equipe.

## Referências

### I. Documentação Google Firebase

Disponível para acesso aqui: <https://firebase.google.com/>

### II. Playlist do Prof. Mateus sobre firebase functions e firestore no Youtube

[https://www.youtube.com/watch?v=gmor\\_IxXHN4&list=PLoxX9g3A8SC\\_bKorJD0ZmwfqFChZ9Iqq](https://www.youtube.com/watch?v=gmor_IxXHN4&list=PLoxX9g3A8SC_bKorJD0ZmwfqFChZ9Iqq)

Atenção: A playlist está com uma versão do Firebase de 2 anos atrás e não está atualizada. Pode ser que nem tudo que funcionava naquela época permaneça funcionando.