Access Urban

Um aplicativo de Acessibilidade Urbana

Everton Luis Nascimento Oliveira

Pablo Vieira Florentino

Orientador

Instituto Federal da Bahia – IFBA
Curso de Análise e Desenvolvimento de
Sistemas Campus Salvador

Salvador, Bahia, Brasil
Julho 2026

Sumário

1. Visão Geral

1.1 Declaração do Problema

Até meados da década de 70, deficiência era predominantemente compreendida a partir do modelo médico, que a associava à lesão corporal, à limitação individual e à necessidade de tratamento, reabilitação ou correção do sujeito. Nessa perspectiva, o indívuduo com deficiência era frequentemente colocada em posição de dependência, tutela e afastamento da vida social, como se suas limitações estivessem restritas exclusivamente a sua condição física e não às condições sociais que dificultam a sua participação plena. Assim, quanto maior fosse o comprometimento físico, sensorial ou intelectual, menor seria a possibilidade de exercicio de direitos e autonomia, conforme critica Werneck (2014).

A partir das mobilizações políticas de pessoas com deficiência, especialmente no contexto britânico da década de 1970, fortaleceu-se o modelo social da deficiência. Esse modelo deslocou o centro da discussão da lesão individual para as barreiras produzidas pela sociedade, evidenciando que a exclusão não decorre apenas das condições corporais, mas da forma como os espaços, as instituições e as relações sociais são organizados (DINIZ, 2007). Desse modo, pensar os direitos das pessoas com deficiência e mobilidade reduzida passou a exigir não apenas cuidado médico, mas também inclusão social, participação política, acessibilidade e reconhecimento da diversidade humana.

Nesse sentido, a acessibilidade urbana constitui elemento essencial para a efetivação do direito à cidade. A legislação brasileira estabelece a acessibilidade como condição para o uso seguro e autônomo dos espaços, mobiliários, transportes, serviços e equipamentos urbanos por pessoas com deficiência ou mobilidade reduzida (BRASIL, 2000; BRASIL, 2004; BRASIL, 2015). Complementarmente, a NBR 9050 apresenta parâmetros técnicos voltados à eliminação de barreiras em edificações, mobiliário, espaços e equipamentos urbanos (ABNT, 2015). Contudo, mesmo diante de avanços normativos, a acessibilidade ainda enfrenta obstáculos concretos no cotidiano das cidades brasileiras, revelando a distância entre o direito formalmente reconhecido e sua efetivação prática.

As barreiras urbanas não devem ser compreendidas apenas como falhas físicas da cidade, mas como mecanismos de exclusão social. Calçadas irregulares, ausência de rampas, obstáculos no percurso, transporte inadequado, sinalização insuficiente e atitudes discriminatórias limitam a autonomia, restringem a circulação e dificultam o acesso ao trabalho, à educação, ao lazer e aos serviços públicos. Como indicam Lima, Carvalho-Freitas e Santos (2013), a falta de acessibilidade produz repercussões psicossociais relevantes, como dependência de terceiros, isolamento social, redução de oportunidades e comprometimento da autoestima. Nessa mesma direção, Kitchin (1998) argumenta que o espaço pode atuar como produtor e mantenedor da exclusão, ao definir quem pode circular, permanecer e participar plenamente da vida urbana.

No caso de Salvador, a acessibilidade urbana deve ser analisada considerando sua formação histórica, sua topografia acidentada, a presença de áreas patrimoniais e a desigual distribuição da infraestrutura urbana. A cidade apresenta uma organização socioespacial marcada por

contrastes entre áreas mais estruturadas e territórios com maiores precariedades urbanas, o que interfere diretamente nas condições de circulação, permanência e acesso aos espaços públicos (SANTOS, 1959; CARVALHO; PEREIRA, 2008). Para pessoas com deficiência e mobilidade reduzida, essas desigualdades tornam-se ainda mais evidentes, pois barreiras arquitetônicas e urbanísticas limitam o direito de ir e vir e dificultam a participação plena na vida urbana. Assim, a acessibilidade em Salvador não deve ser tratada como adaptação pontual, mas como parte de uma política urbana integrada, capaz de articular mobilidade, inclusão, preservação patrimonial e direito à cidade (REIS, 2015; VASCONCELLOS, 2019).

Dessa forma, a acessibilidade urbana em Salvador deve ser compreendida como questão de cidadania, justiça social e direito à cidade. Não basta que determinados espaços sejam parcialmente adaptados; é necessário que a cidade seja planejada, fiscalizada e transformada para acolher a diversidade de corpos, modos de locomoção e necessidades de seus habitantes. A superação das barreiras arquitetônicas, urbanísticas e atitudinais exige compromisso do poder público, participação social e mudança cultural, pois a acessibilidade não deve ser vista como favor ou benefício, mas como condição básica para a dignidade humana e para o exercício pleno da cidadania (SASSAKI, 2006; MANTOAN, 2005).

Portanto, discutir a acessibilidade urbana das pessoas com deficiência e mobilidade reduzida significa reconhecer que o direito à cidade só se concretiza quando todos podem circular, acessar, permanecer e participar dos espaços urbanos com segurança, autonomia e dignidade. A cidade inclusiva não é aquela que apenas adapta parte de sua estrutura depois que as barreiras já foram produzidas, mas aquela que incorpora a diversidade humana desde sua concepção. Nesse sentido, garantir acessibilidade em Salvador e nas demais cidades brasileiras é afirmar que pessoas com deficiência e mobilidade reduzida não devem ocupar um lugar periférico na vida urbana, mas participar plenamente dela como sujeitos de direitos.

1.2 Proposta de Solução de Software

A proposta deste software surge como resposta aos persistentes desafios de acessibilidade enfrentados por pessoas com deficiência física e mobilidade reduzida em Salvador. Fundamentado nos princípios de design universal e participação cidadã, o aplicativo tem como objetivo principal transformar a relação entre os usuários e o espaço urbano, criando um mecanismo eficaz para identificação e registro colaborativo de barreiras arquitetônicas. Inspirado no conceito de crowdsourcing urbano desenvolvido por Goodchild (2007), a plataforma permitirá que os próprios cidadãos documentem obstáculos como calçadas irregulares, falta de rampas e pontos de transporte público inacessíveis, gerando um mapa dinâmico e atualizado das condições de acessibilidade na cidade.

A importância desta solução tecnológica reside em sua capacidade de empoderar a comunidade, seguindo o princípio "Nada sobre nós, sem nós" defendido por Charlton (2000). Atualmente, muitas das barreiras urbanas permanecem invisíveis aos olhos do poder público, criando um ambiente hostil que limita a mobilidade e a autonomia das pessoas com deficiência. O software rompe com essa invisibilidade ao transformar cada usuário em um agente ativo no monitoramento

da acessibilidade, permitindo não apenas o registro de problemas, mas também o compartilhamento de informações sobre rotas viáveis e experiências de deslocamento. Essa troca de conhecimentos práticos entre os usuários se mostra particularmente valiosa em um contexto urbano como o de Salvador, onde as condições de acessibilidade variam significativamente entre diferentes regiões.

Além de seu impacto imediato na vida cotidiana dos usuários, a plataforma possui um importante papel como ferramenta de advocacy e transformação urbana. Os dados georreferenciados coletados pelo sistema fornecem evidências concretas que podem subsidiar políticas públicas mais eficientes, direcionando investimentos para as áreas mais críticas e fortalecendo a fiscalização das leis de acessibilidade. Como demonstrado por Ferreira et al. (2020), iniciativas colaborativas desse tipo têm o potencial de aumentar significativamente a transparência e a responsabilidade governamental na gestão urbana. A longo prazo, espera-se que o software contribua não apenas para a melhoria da infraestrutura física, mas também para a construção de uma cultura urbana mais inclusiva, onde o direito à cidade seja garantido para todos.

O desenvolvimento desta solução tecnológica se alinha com os Objetivos de Desenvolvimento Sustentável da ONU, particularmente com a meta de cidades inclusivas, seguras, resilientes e sustentáveis. Ao combinar tecnologia móvel, participação cidadã e gestão de dados urbanos, o projeto representa uma abordagem inovadora para um problema antigo, oferecendo tanto soluções imediatas para os usuários quanto ferramentas para transformações estruturais. Mais do que um simples aplicativo, esta plataforma se configura como um instrumento de mudança social, capaz de promover a inclusão socioespacial e fortalecer a autonomia das pessoas com deficiência no espaço urbano de Salvador.

1.3 Tecnologias Adotadas

1.3.1 Ambiente de Desenvolvimento: Android Studio

O Android Studio foi utilizado como ambiente integrado de desenvolvimento por oferecer uma estrutura adequada à criação, organização, execução e depuração de aplicações Android. Por ser a IDE oficial mantida pelo Google, concentra recursos essenciais ao ciclo de desenvolvimento mobile, como editor de código, gerenciamento de dependências, editor visual de interfaces, ferramentas de inspeção, integração com emuladores e suporte aos principais componentes do ecossistema Android.

Sua adoção contribuiu diretamente para a estruturação do projeto, permitindo o desenvolvimento das telas em XML, a implementação das classes em Java, a configuração de permissões, bibliotecas externas e arquivos de manifesto, além da execução da aplicação em ambiente de testes. O suporte ao Gradle também foi relevante, pois possibilitou gerenciar versões de dependências, SDKs e plugins necessários ao funcionamento do sistema.

Apesar de sua robustez, o uso do Android Studio exigiu cuidados técnicos durante a configuração do ambiente. Questões relacionadas à compatibilidade entre versões do Gradle, Android Gradle Plugin, SDK mínimo, bibliotecas externas e emuladores impactaram diretamente o processo de compilação. Esses ajustes evidenciam que, em projetos Android, a estabilidade da aplicação

depende não apenas do código-fonte, mas também da coerência entre as ferramentas que compõem o ambiente de desenvolvimento.

1.3.2 Linguagem de Programação: Java

A linguagem Java foi adotada por sua maturidade, estabilidade e compatibilidade consolidada com o desenvolvimento Android. Embora o Kotlin seja atualmente a linguagem recomendada pelo Google para novos projetos, Java permanece amplamente suportada e ainda se mostra adequada para aplicações que exigem clareza estrutural, manutenção contínua e integração com bibliotecas já consolidadas.

No desenvolvimento da aplicação, Java foi empregado na construção da lógica das telas, no tratamento de eventos da interface, na validação de dados, na organização de modelos e na comunicação entre os componentes do sistema. A orientação a objetos favoreceu a separação de responsabilidades, permitindo a criação de classes específicas para representar entidades, controlar regras de cadastro e login, manipular dados e interagir com serviços externos.

A escolha da linguagem também se relaciona à sua forte tipagem e à capacidade de identificar erros ainda em tempo de compilação, característica importante para reduzir inconsistências no desenvolvimento. Por outro lado, sua utilização demandou maior rigor na organização do código, principalmente no tratamento de exceções, manipulação de componentes visuais, uso de bibliotecas externas e implementação de operações assíncronas, necessárias para evitar travamentos na interface do usuário.

1.3.3 Framework de Desenvolvimento da API: Spring Boot

O Spring Boot é um framework pertencente ao ecossistema Spring, amplamente utilizado no desenvolvimento de aplicações Java e APIs REST. Sua principal finalidade consiste em simplificar a configuração, a implementação e a execução de sistemas, por meio de recursos como configuração automática, gerenciamento de dependências e servidor de aplicação incorporado.

No projeto desenvolvido, sua utilização fundamenta-se na necessidade de estabelecer uma camada intermediária entre o aplicativo Android, o banco de dados e os demais serviços utilizados pelo sistema. A API Spring Boot será responsável por receber as requisições do aplicativo, validar os dados, aplicar as regras de negócio e executar as operações de consulta, cadastro, atualização e exclusão das informações.

A adoção de uma arquitetura baseada em API REST permite que os recursos do sistema sejam disponibilizados por meio de endpoints acessados pelo protocolo HTTP. Essa abordagem promove o desacoplamento entre a interface móvel e a camada de processamento dos dados, favorecendo a organização do código, a manutenção do sistema e a implementação de novas funcionalidades.

O Spring Boot também possibilita a integração com módulos como Spring Web, Spring Data JPA, Bean Validation e Spring Security. Esses recursos auxiliam, respectivamente, na criação dos endpoints, na comunicação com o banco de dados PostgreSQL, na validação das informações recebidas e na proteção dos recursos por meio de mecanismos de autenticação e autorização.

Portanto, o Spring Boot apresenta-se como uma tecnologia adequada para o aplicativo desenvolvido, pois contribui para uma arquitetura mais segura, modular e escalável. Além disso, evita que o aplicativo Android realize conexões diretas com o banco de dados, garantindo maior controle sobre o acesso, a integridade e o processamento das informações relacionadas aos usuários e às barreiras urbanas.

1.3.4 Banco de Dados: PostgreSQL

O PostgreSQL foi escolhido como sistema gerenciador de banco de dados por oferecer confiabilidade, integridade referencial e suporte a operações relacionais complexas. A estrutura do projeto demanda o armazenamento consistente de informações como usuários, bairros, perfis de acessibilidade e barreiras urbanas, o que torna necessário um banco capaz de representar relações entre entidades e aplicar restrições de integridade.

A modelagem relacional permite organizar os dados em tabelas com chaves primárias, chaves estrangeiras, campos obrigatórios, valores únicos e regras de validação. Essa organização é fundamental para evitar duplicidade de registros, inconsistências cadastrais e perda de informações. No caso das barreiras urbanas, o banco também armazena coordenadas geográficas, como latitude e longitude, permitindo associar cada ocorrência a um ponto específico do mapa.

Outro fator relevante é o suporte do PostgreSQL às propriedades ACID, que asseguram maior confiabilidade nas transações realizadas. Além disso, a possibilidade de expansão com a extensão PostGIS torna o banco adequado para evoluções futuras envolvendo consultas espaciais mais avançadas, como busca por proximidade, agrupamento territorial ou análise geográfica das barreiras cadastradas. Os principais desafios técnicos envolveram a definição correta dos relacionamentos, a padronização dos nomes das tabelas e campos, a configuração da conexão com a API e o cuidado com dados sensíveis, especialmente credenciais e informações pessoais dos usuários.

1.3.5 API de Mapas: Google Maps SDK para Android

O Google Maps SDK para Android constitui um dos principais recursos funcionais da aplicação, pois viabiliza a representação espacial das barreiras urbanas. A proposta do sistema depende da interação com o mapa, permitindo que o usuário identifique um ponto da cidade, registre uma ocorrência e associe esse registro a uma localização geográfica precisa.

A tecnologia permite exibir mapas interativos, capturar coordenadas, adicionar marcadores e personalizar elementos visuais conforme as características das barreiras cadastradas. Dessa forma, o mapa deixa de ser apenas um recurso de visualização e passa a funcionar como

interface principal de interação entre o usuário e o espaço urbano. O uso de marcadores personalizados também favorece a identificação visual do nível de impedimento, possibilitando diferenciar ocorrências de baixo, médio e alto impacto.

A escolha do Google Maps SDK decorre de sua estabilidade, ampla adoção em aplicações móveis e integração com os serviços de localização do Android. No entanto, sua implementação envolve desafios específicos, como a obtenção e restrição da chave de API, a ativação dos serviços necessários no Google Cloud, o tratamento das permissões de localização e a sincronização entre o ponto selecionado no mapa e os dados preenchidos no formulário de cadastro. Esses aspectos exigem atenção para garantir precisão, segurança e boa experiência de uso.

1.3.6 Autenticação: Firebase Authentication

O Firebase Authentication foi considerado como solução para o gerenciamento da identidade dos usuários, oferecendo mecanismos prontos para cadastro, login e recuperação de acesso.

Sua utilização contribui para reduzir a complexidade de implementar manualmente fluxos de autenticação, especialmente em cenários que envolvem e-mail e senha. Além disso, por ser um serviço gerenciado, oferece recursos de segurança, integração com o Android e possibilidade de expansão para outros provedores de login, caso necessário.

Dentro da solução, a autenticação tem papel estratégico na rastreabilidade das informações. Cada barreira registrada precisa estar associada a um usuário, tanto para fins de organização dos dados quanto para controle, validação e eventual manutenção dos registros. Como desafios técnicos, destacam-se a configuração do projeto no Firebase Console, a ativação correta dos métodos de login, a integração do arquivo de configuração ao Android Studio e a definição clara da relação entre a autenticação e a base de dados da aplicação.

1.3.7 Armazenamento de Mídia: Firebase Storage

O Firebase Storage foi adotado para o armazenamento de mídias vinculadas aos registros de barreiras, como imagens e vídeos. Esse tipo de conteúdo tende a ocupar maior volume de dados e, por isso, não é recomendável armazená-lo diretamente em tabelas relacionais. A alternativa mais adequada é manter os arquivos em um serviço próprio de armazenamento e registrar no banco apenas sua referência, como URL ou caminho.

Essa estratégia melhora a organização da arquitetura, reduz a sobrecarga sobre o banco de dados e facilita o gerenciamento dos arquivos enviados pelos usuários. As mídias desempenham função importante no sistema, pois servem como evidência visual das barreiras encontradas, contribuindo para tornar os registros mais claros, verificáveis e úteis para consulta posterior.

A escolha do Firebase Storage se justifica pela integração com aplicações Android, pela escalabilidade e pela possibilidade de definir regras de segurança para leitura e escrita dos

arquivos. Sua implementação, entretanto, exige cuidados relacionados ao controle de permissões, validação do tipo e tamanho dos arquivos, tratamento de falhas em conexões móveis instáveis e associação correta entre cada mídia enviada e o respectivo registro de barreira no banco de dados.

1.4 Trabalhos Relacionados

1.4.1 Ldn Access

O Ldn Access é uma plataforma colaborativa desenvolvida em Londres com foco no mapeamento de barreiras urbanas que dificultam a circulação de pessoas com deficiência e mobilidade reduzida. A solução permite que os próprios usuários registrem problemas encontrados no espaço urbano, como calçadas irregulares, ausência de rampas, obstáculos em vias públicas e demais condições que comprometem a mobilidade autônoma.

Seu funcionamento baseia-se no uso de geolocalização, permitindo associar cada barreira identificada a um ponto específico da cidade. Dessa forma, a plataforma contribui para a construção de uma base de informações alimentada pela comunidade, aproximando a população do processo de identificação dos problemas urbanos. O principal diferencial da solução está em reconhecer o cidadão como agente ativo no levantamento das condições de acessibilidade, fortalecendo a participação social no debate sobre mobilidade inclusiva.

1.4.2 Acessu Urbano

O Acessu Urbano é um aplicativo brasileiro voltado ao compartilhamento de informações sobre acessibilidade em estabelecimentos e espaços urbanos. A plataforma permite que os usuários avaliem e registrem aspectos relacionados às condições de acesso, incluindo a existência de rampas, banheiros adaptados e outros recursos relevantes para pessoas com deficiência ou mobilidade reduzida.

1.4.3 Wheelmap

O Wheelmap é uma plataforma colaborativa de alcance global destinada ao mapeamento das condições de acessibilidade de locais públicos, com atenção especial às necessidades de pessoas que utilizam cadeira de rodas. Desenvolvida na Alemanha, a solução permite que usuários avaliem espaços e estabelecimentos de acordo com seu grau de acessibilidade.

Um dos aspectos mais relevantes da plataforma é o uso de uma classificação visual baseada nas cores verde, amarelo e vermelho. Essa estratégia simplifica a interpretação das informações, indicando se determinado local é acessível, parcialmente acessível ou não acessível. A clareza dessa representação favorece tanto a consulta rápida quanto a colaboração dos usuários, tornando o Wheelmap uma referência importante entre as soluções digitais voltadas ao mapeamento colaborativo da acessibilidade.

1.4.4 Moovit

O Moovit é uma aplicação de mobilidade urbana voltada ao planejamento de deslocamentos por transporte público. Diferentemente das plataformas centradas exclusivamente no registro de barreiras, sua principal finalidade é fornecer informações sobre linhas, itinerários, horários estimados, rotas e condições operacionais dos sistemas de transporte.

No campo da acessibilidade, o Moovit contribui ao disponibilizar informações sobre recursos presentes em ônibus, metrôs, estações e demais componentes da rede de transporte público. Esses dados auxiliam pessoas com deficiência ou mobilidade reduzida na escolha de trajetos mais adequados às suas necessidades. Assim, embora sua proposta seja mais ampla e voltada à mobilidade urbana em geral, a ferramenta possui relevância por apoiar a tomada de decisão durante os deslocamentos cotidianos.

1.4.5 AccessNow

O AccessNow é uma plataforma colaborativa de alcance internacional voltada ao compartilhamento de informações sobre a acessibilidade de locais em diferentes cidades. A aplicação permite que os usuários classifiquem estabelecimentos e espaços urbanos como acessíveis, parcialmente acessíveis ou não acessíveis, formando uma base de dados construída de maneira coletiva.

A ferramenta se destaca por ampliar a visibilidade das condições de acessibilidade em escala global, permitindo que usuários consultem previamente informações sobre os locais que pretendem frequentar. Sua lógica colaborativa reforça a importância da experiência direta das pessoas com deficiência ou mobilidade reduzida na avaliação dos espaços urbanos. Dessa forma, o AccessNow contribui para transformar vivências individuais em informações públicas úteis para a comunidade.

1.4.6 Discussão crítica

As soluções analisadas demonstram a relevância do uso de tecnologias digitais no enfrentamento das barreiras urbanas e informacionais que afetam pessoas com deficiência ou mobilidade reduzida. Plataformas como Wheelmap e AccessNow evidenciam a importância da colaboração comunitária na avaliação da acessibilidade de locais, enquanto o Ldn Access aproxima essa lógica do registro direto de barreiras presentes no espaço urbano. O Acessu Urbano, por sua vez, contribui ao tratar a acessibilidade em espaços e estabelecimentos no contexto brasileiro, e o Moovit amplia essa discussão ao relacioná-la ao planejamento dos deslocamentos por transporte público.

Apesar dessas contribuições, observa-se que parte dessas soluções se concentra na avaliação geral de locais ou estabelecimentos, classificando-os conforme seu grau de acessibilidade. Essa abordagem é relevante, mas nem sempre evidência de forma específica as barreiras encontradas no percurso urbano, como calçadas danificadas, ausência de rampas, obstáculos físicos, sinalização inadequada ou comportamentos excludentes. Além disso, soluções de alcance global

nem sempre contemplam as particularidades territoriais, sociais e urbanísticas de cidades específicas, especialmente aquelas marcadas por desigualdades socioespaciais, topografia complexa e infraestrutura urbana irregular.

Nesse contexto, a solução proposta diferencia-se por direcionar seu foco ao registro simplificado de barreiras arquitetônicas e atitudinais em pontos específicos do mapa. Em vez de avaliar apenas se um local é acessível ou não, a proposta busca identificar a ocorrência concreta da barreira, seu nível de impedimento, o tipo de usuário impactado, sua descrição e, quando possível, uma mídia associada. Essa estrutura permite produzir informações mais direcionadas sobre os obstáculos vivenciados no cotidiano urbano.

Outro aspecto relevante está na delimitação territorial da proposta, voltada à cidade de Salvador. Essa escolha permite considerar particularidades locais que muitas plataformas generalistas não abordam de forma específica. Ao possibilitar o mapeamento colaborativo de barreiras urbanas no contexto soteropolitano, a solução contribui para ampliar a visibilidade dos problemas de acessibilidade enfrentados por pessoas com deficiência e mobilidade reduzida, fortalecendo a relação entre tecnologia, participação social e direito à cidade.

Assim, enquanto os trabalhos relacionados demonstram diferentes caminhos para informar, classificar ou planejar deslocamentos acessíveis, a solução desenvolvida busca atuar de forma mais objetiva sobre o registro das barreiras urbanas. Seu diferencial está na simplicidade do cadastro, na associação direta com o mapa e na possibilidade de reunir dados organizados sobre impedimentos concretos à circulação, contribuindo para uma compreensão mais situada e participativa da acessibilidade urbana.

Tabela 1 – Comparativo entre aplicativos relacionados e a solução proposta

Aplicativos/ Funcionalidades	Mapeamento colaborativo	Geolocalização	Registro de barreiras urbanas	Classificação por cores/severidade	Validação colaborativa
Ldn Access	X	X	-	-	-
Acessu Urbano	X	X	X	-	-
Wheelmap	X	X	-	X	-
Moovit	-	X	-	-	-
AccessNow	X	X	-	-	-
Access Urban	X	X	X	X	X

2. Requisitos

2.1 Requisitos Funcionais

ID	Requisito Funcional
RF1	O sistema deve permitir que os usuários façam login utilizando sua conta no aplicativo.

RF2	O sistema deve permitir que os usuários realizem o próprio cadastro no aplicativo.
RF3	O sistema deve permitir que os usuários solicitem a redefinição da senha por meio do e-mail cadastrado.
RF4	O sistema deve permitir que os usuários autenticados visualizem e atualizem seus dados cadastrais, exceto o endereço de e-mail.
RF5	O sistema deve exibir um mapa interativo da cidade de Salvador contendo as barreiras urbanas cadastradas pelos usuários.
RF6	O sistema deve permitir que os usuários visualizem sua localização atual no mapa, mediante autorização de acesso à localização do dispositivo.
RF7	O sistema deve permitir que os usuários realizem buscas por bairros, endereços ou locais da cidade de Salvador.
RF8	O sistema deve permitir que usuários autenticados cadastrem uma barreira urbana após selecionarem uma localização válida no mapa.
RF9	O sistema deve validar se a localização selecionada para o cadastro da barreira está dentro dos limites territoriais da cidade de Salvador.
RF10	O sistema deve exibir um marcador temporário no mapa após o usuário selecionar o local onde deseja cadastrar uma nova barreira.
RF11	O sistema deve permitir que o usuário informe, durante o cadastro da barreira, o título, o tipo da barreira, o tipo de usuário impactado, o nível de impedimento e a descrição da ocorrência.
RF12	O sistema deve permitir que os usuários adicionem imagens ou vídeos ao cadastro de uma barreira urbana.
RF13	O sistema deve representar visualmente as barreiras no mapa por meio de marcadores com cores correspondentes ao nível de impedimento: amarelo para baixo, laranja para médio e vermelho para alto.
RF14	O sistema deve permitir que os usuários visualizem os detalhes das barreiras cadastradas, incluindo localização, descrição, nível de impedimento, público impactado e mídias associadas.
RF15	O sistema deve permitir que os usuários acessem, editem e excluam as barreiras cadastradas por eles próprios.
RF16	O sistema deve permitir que usuários autenticados confirmem a permanência de uma barreira ou reportem registros que apresentem informações incorretas ou desatualizadas.
RF17	O sistema deve exibir a quantidade de confirmações e registros de barreiras associados a cada barreira cadastrada.

2.2 Requisitos Não-Funcionais

ID	Requisito Não-Funcional	Categoria
RNF1	O sistema deve apresentar tempo de resposta adequado durante a navegação entre telas, o carregamento de dados e a execução das principais funcionalidades.	Performance
RNF2	As operações de comunicação com a API, Firebase e Google Maps devem ser executadas de forma assíncrona, evitando o bloqueio da interface do usuário.	Performance
RNF3	O carregamento, a atualização e a filtragem das barreiras exibidas no mapa devem ocorrer de forma eficiente, mantendo a fluidez da aplicação.	Performance
RNF4	O sistema deve manter suas funcionalidades disponíveis sempre que os serviços de autenticação, armazenamento, banco de dados e API estiverem operacionais.	Disponibilidade
RNF5	O aplicativo deve tratar falhas de conexão com a internet ou indisponibilidade dos serviços externos sem encerrar inesperadamente, apresentando mensagens adequadas ao usuário.	Confiabilidade
RNF6	O acesso às funcionalidades restritas deve ser permitido somente após a autenticação válida do usuário por meio do Firebase Authentication.	Segurança
RNF7	As senhas dos usuários não devem ser armazenadas no banco de dados PostgreSQL, ficando o gerenciamento das credenciais sob responsabilidade do Firebase Authentication.	Segurança
RNF8	O sistema deve garantir que apenas o usuário responsável pelo cadastro de uma barreira possa editar ou excluir o respectivo registro.	Segurança
RNF9	O aplicativo deve fornecer feedback visual ao usuário durante operações de carregamento, cadastro, edição, exclusão, confirmação e registro de barreiras.	Usabilidade
RNF10	As mensagens de erro, aviso e confirmação devem utilizar linguagem clara, objetiva e adequada ao contexto da operação realizada.	Usabilidade
RNF11	A interface deve utilizar textos legíveis, contraste adequado, áreas de toque compatíveis com dispositivos móveis e descrições de acessibilidade nos componentes relevantes.	Acessibilidade
RNF12	O aplicativo deve ser compatível com as versões do Android definidas no projeto e adaptar sua interface a diferentes tamanhos e resoluções de tela.	Compatibilidade
RNF13	O código-fonte deve manter separação de responsabilidades por meio da arquitetura MVVM, organização em camadas e documentação dos principais componentes, facilitando futuras correções e evoluções do sistema.	Manutenibilidade

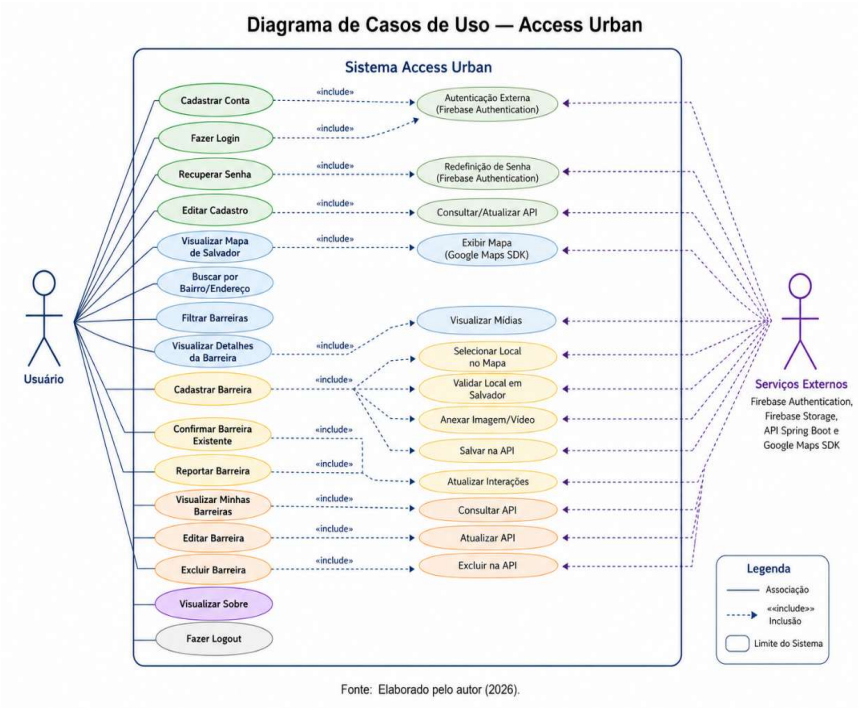
RNF14	O sistema deve validar os dados informados pelos usuários antes do envio ao servidor, impedindo o cadastro de informações obrigatórias vazias ou em formato inválido.	Integridade de Dados
RNF15	O sistema deve preservar a consistência das informações armazenadas, evitando registros duplicados ou associações inválidas entre usuários, barreiras, mídias e interações.	Integridade de Dados
RNF16	O aplicativo deve solicitar somente as permissões necessárias para o funcionamento de recursos como localização, câmera e acesso a mídias, informando ao usuário a finalidade de cada permissão.	Privacidade
RNF17	A aplicação deve permitir a evolução de funcionalidades e a substituição de serviços com impacto reduzido sobre os demais componentes, mantendo baixo acoplamento entre as camadas do sistema.	Extensibilidade

3. Design

3.1 Projeto UML

3.1.1 Diagrama de Casos de Uso

Figura 1 - Diagrama de casos de uso



Fonte: Elaborado pelo autor (2026).

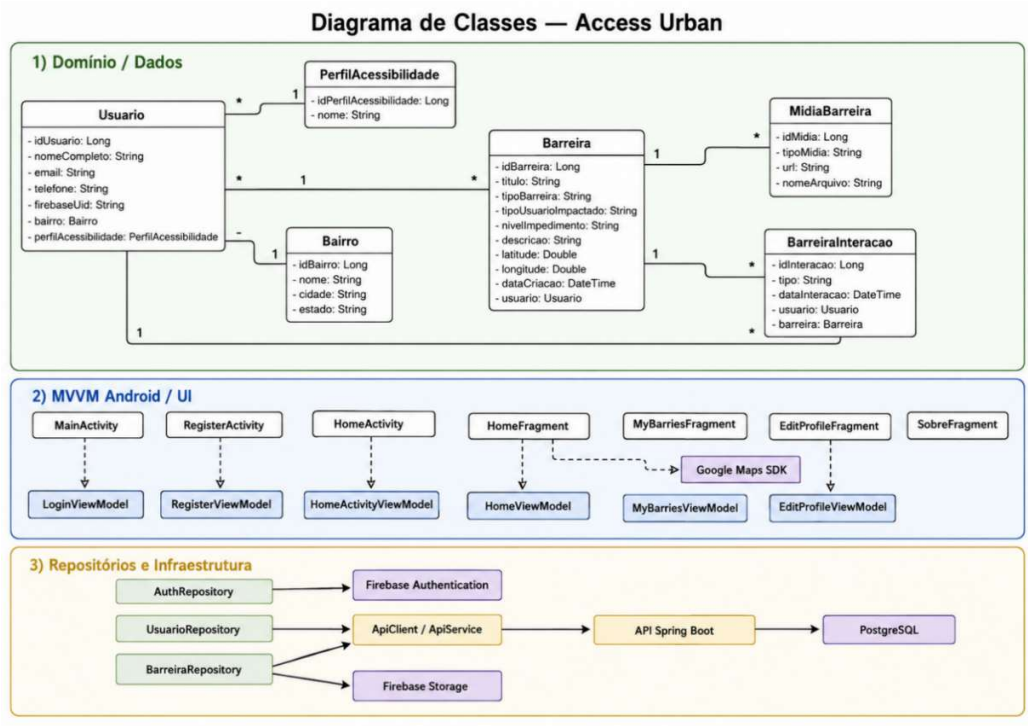
O diagrama de casos de uso apresenta as principais funcionalidades do sistema, evidenciando a interação do usuário com os recursos de autenticação, perfil, navegação, consulta e gerenciamento de barreiras urbanas. A representação também contempla a dependência de serviços externos, como Firebase Authentication, Firebase Storage, API Spring Boot e Google Maps SDK, responsáveis, respectivamente, pela autenticação, armazenamento de mídias, persistência de dados e visualização cartográfica.

As funcionalidades iniciais permitem ao usuário cadastrar conta, realizar login, recuperar senha, editar dados pessoais e encerrar a sessão. Após autenticado, o usuário pode visualizar o mapa de Salvador, buscar locais ou bairros, filtrar barreiras e consultar detalhes das ocorrências cadastradas. O cadastro de barreiras ocorre a partir da seleção de um ponto no mapa, possibilitando o registro de informações como título, tipo de usuário impactado, nível de impedimento, descrição e mídia associada.

O diagrama também contempla funcionalidades de interação colaborativa, como confirmar ou reportar barreiras, além do gerenciamento individual dos registros por meio da opção “Minhas Barreiras”, que permite visualizar, editar ou excluir ocorrências cadastradas pelo próprio usuário. Dessa forma, o modelo evidencia uma arquitetura funcional centrada no mapeamento colaborativo de barreiras arquitetônicas e atitudinais, articulando recursos de geolocalização, autenticação, armazenamento e comunicação com a API para apoiar a acessibilidade urbana.

3.1.2 Diagrama de Classe

Figura 2 - Diagrama de classe



Fonte: Elaborado pelo autor (2026).

O diagrama de classes apresenta a organização estrutural do sistema, dividida em três camadas principais: domínio/dados, interface Android baseada em MVVM e repositórios/infraestrutura. A camada de domínio reúne as entidades centrais da aplicação, responsáveis por representar usuários, perfis de acessibilidade, bairros, barreiras, mídias associadas e interações realizadas sobre os registros.

A entidade Usuario concentra os dados cadastrais e mantém associação com Bairro e PerfilAcessibilidade, além de se relacionar com as barreiras cadastradas. A classe Barreira representa o núcleo funcional do sistema, armazenando informações como título, tipo da barreira, público impactado, nível de impedimento, descrição, coordenadas geográficas e usuário responsável pelo registro. As classes MidiaBarreira e BarreiraInteracao complementam esse modelo, permitindo vincular imagens ou vídeos às ocorrências e registrar ações colaborativas, como confirmações e registros de barreiras.

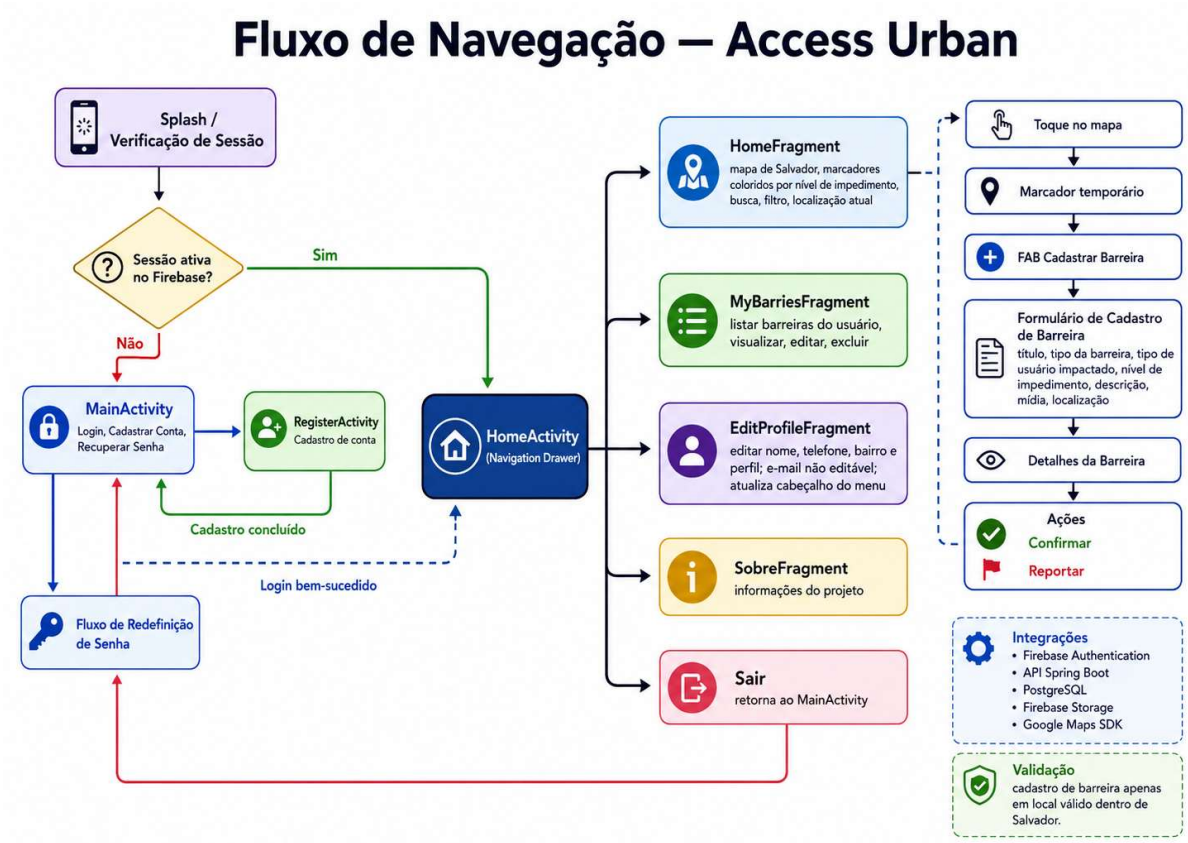
Na camada Android, a aplicação é estruturada segundo o padrão MVVM, separando os componentes de interface, como Activities e Fragments, dos respectivos ViewModels. Essa organização reduz o acoplamento entre apresentação e regras de negócio, favorecendo maior clareza, eficiência nos testes e manutenção do código. Os ViewModels atuam como intermediários entre a interface e os repositórios, processando ações relacionadas à autenticação, cadastro, perfil, mapa e gerenciamento de barreiras.

A camada de repositórios e infraestrutura concentra a comunicação com serviços externos e recursos de persistência. O AuthRepository integra-se ao Firebase Authentication, enquanto os repositórios de usuário e barreira acessam a API por meio do ApiClient e do ApiService. A API Spring Boot centraliza as regras de negócio e realiza a persistência dos dados no PostgreSQL. Já o Firebase Storage armazena as mídias vinculadas às barreiras, e o Google Maps SDK fornece os recursos cartográficos utilizados na visualização e no cadastro georreferenciado das ocorrências.

Dessa forma, o diagrama evidencia uma arquitetura modular e em camadas, na qual as responsabilidades são distribuídas entre entidades de domínio, componentes de interface, ViewModels, repositórios e serviços externos. Essa separação contribui para a escalabilidade, manutenção e evolução da aplicação, além de organizar tecnicamente as funcionalidades relacionadas ao mapeamento colaborativo de barreiras urbanas.

3.2 Definição do Fluxo de Navegação

Figura 3 - Fluxo de navegação do aplicativo



Fonte: Elaborado pelo autor (2026).

O fluxo de navegação inicia com a verificação da existência de uma sessão autenticada do usuário. Caso exista uma sessão ativa, a aplicação direciona o usuário para a HomeActivity; caso contrário, apresenta a mainsActivity, responsável pelas opções de login, cadastro e redefinição de senha. Esse procedimento organiza o acesso inicial ao sistema e garante que as funcionalidades internas sejam disponibilizadas apenas a usuários autenticados.

A MainActivity concentra o processo de autenticação, permitindo o acesso por e-mail e senha, além do redirecionamento para a RegisterActivity e para o fluxo de recuperação de senha. No cadastro, o usuário informa seus dados pessoais, perfil de acessibilidade, bairro de residência e credenciais de acesso. Após a validação das informações, os dados são encaminhados aos serviços responsáveis pela autenticação e persistência, integrando Firebase Authentication, API Spring Boot e PostgreSQL.

Posterior ao login, a HomeActivity passa a controlar a navegação principal por meio de um Navigation Drawer, permitindo o acesso aos fragments do sistema. O HomeFragment é a tela central da aplicação, responsável pela exibição do mapa da cidade de Salvador, cadastro e consulta de barreiras, busca por locais ou bairros, aplicação de filtros e visualização dos

marcadores. Os marcadores são diferenciados por cores, conforme o nível de impedimento registrado: baixo, médio ou alto.

O cadastro de barreiras ocorre diretamente a partir da interação com o mapa. Ao tocar em um ponto válido, o sistema cria um marcador temporário e habilita o formulário de cadastro, no qual são informados título, tipo de barreira, tipo de usuário impactado, nível de impedimento, descrição e mídia associada. Antes da persistência, a aplicação valida se a localização pertence à área permitida de Salvador. Em seguida, os dados são enviados à API, armazenados no PostgreSQL e, quando houver imagem ou vídeo, vinculados ao Firebase Storage.

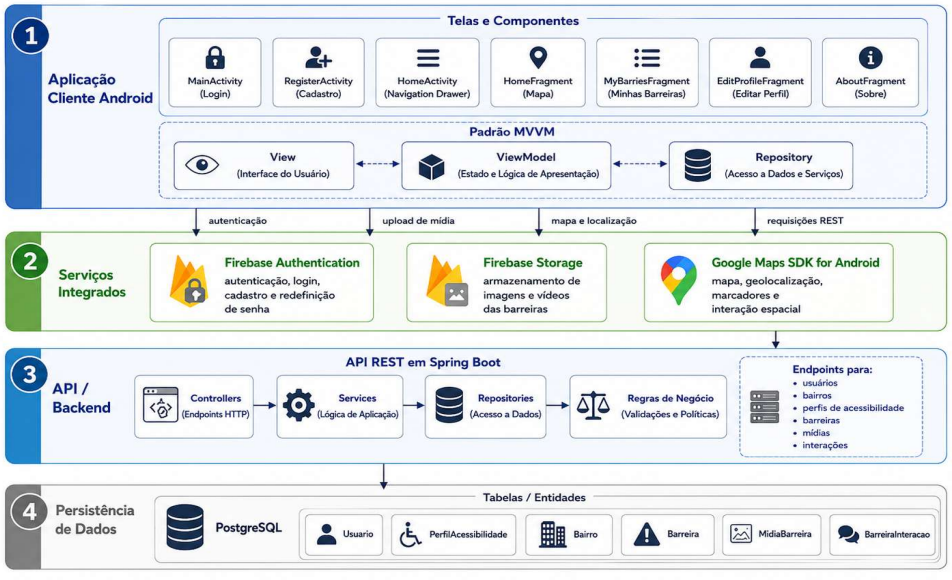
A navegação também contempla a visualização detalhada das barreiras, permitindo consultar informações da ocorrência, mídias e interações colaborativas. Nessa tela, usuários autenticados podem confirmar a permanência da barreira ou reportar inconsistências no registro. Já o MyBarriesFragment reúne as barreiras cadastradas pelo próprio usuário, oferecendo ações de visualização, edição e exclusão.

Além das funcionalidades relacionadas ao mapa, o fluxo inclui o EditProfileFragment, destinado à atualização dos dados cadastrais permitidos, e o SobreFragment, voltado à apresentação das informações institucionais do projeto. A opção de Sair encerra a sessão autenticada e retorna o usuário à tela inicial de acesso.

Dessa forma, o fluxo de navegação organiza as principais operações do sistema em uma sequência lógica, integrando autenticação, gerenciamento de usuário, visualização cartográfica, cadastro de barreiras e participação colaborativa. Essa estrutura favorece a usabilidade da aplicação e mantém a separação entre as etapas de acesso, navegação, registro e consulta das informações.

3.3 Visão Arquitetural

Figura 4 - Visão arquitetural do sistema



Fonte: Elaborado pelo autor (2026).

A arquitetura do sistema foi definida com o objetivo de separar responsabilidades, reduzir o acoplamento entre os componentes e organizar a comunicação entre a aplicação móvel, os serviços externos, a API e a camada de persistência. Para isso, adotou-se uma estrutura em camadas composta pelo cliente Android, pelos serviços integrados, pela API REST desenvolvida em Spring Boot e pelo banco de dados PostgreSQL. Essa divisão permite que cada parte do sistema exerça uma função específica: a aplicação móvel concentra a interação com o usuário; os serviços externos tratam autenticação, mídias e recursos cartográficos; a API centraliza as regras de negócio; e o banco relacional armazena os dados estruturados da aplicação. De forma complementar, a arquitetura emprega padrões como Singleton, para controlar instâncias únicas de serviços ou clientes de conexão, e Observer, quando a interface acompanha alterações de estado ou respostas emitidas pelo viewModel.

No cliente Android, a adoção do padrão MVVM — Model, View e ViewModel — contribui para uma melhor organização do código-fonte. As Activities e Fragments compõem a camada de visualização, sendo responsáveis pela exibição das telas e pela captura das ações do usuário. Os ViewModels atuam como intermediários entre a interface e os repositórios, controlando estados, processando eventos e evitando que regras de apresentação e acesso a dados fiquem concentradas diretamente nas telas. Essa separação favorece a manutenção, a verificabilidade e a evolução da aplicação, especialmente em funcionalidades que envolvem autenticação, edição de perfil, visualização de mapa e gerenciamento de barreiras.

A escolha pelo MVVM se justifica pela necessidade de manter a interface desacoplada da lógica de processamento. Em uma aplicação que manipula dados de usuário, localização, mídias e registros colaborativos, concentrar todas as operações em Activities ou Fragments tornaria o código mais extenso, repetitivo e difícil de manter. Com o MVVM, os componentes de interface permanecem voltados à apresentação, enquanto os ViewModels e repositórios organizam o fluxo de dados e a comunicação com os serviços externos. Como limitação, esse padrão exige maior planejamento estrutural, pois aumenta a quantidade de classes e demanda uma definição clara das responsabilidades de cada camada.

A camada de repositórios exerce papel fundamental na comunicação entre o aplicativo e as fontes de dados. O AuthRepository se integra ao Firebase Authentication para realizar operações de cadastro, login, recuperação de senha e encerramento de sessão. O BarreiraRepository e o UsuarioRepository interagem com a API REST para consultar, cadastrar, atualizar e excluir informações. Essa estratégia centraliza o acesso aos dados e evita que as telas dependam diretamente de detalhes técnicos de comunicação com Firebase, API ou banco de dados.

A API REST foi adotada como meio de comunicação entre o aplicativo Android e o backend por oferecer uma forma padronizada, escalável e amplamente utilizada de troca de informações por meio do protocolo HTTP. Com essa abordagem, o aplicativo envia requisições para endpoints específicos e recebe respostas em formato estruturado, geralmente JSON. A adoção de uma API evita que o cliente Android acesse diretamente o banco de dados, reduzindo riscos de segurança, exposição de credenciais e inconsistências nas regras de negócio. Além disso, permite que futuras versões do sistema, como uma aplicação web ou painel administrativo, consumam os mesmos serviços.

O Spring Boot foi escolhido para o desenvolvimento do backend por simplificar a construção de

APIs em Java e oferecer recursos adequados para aplicações em camadas. Através de componentes como controllers, services e repositories, torna-se possível organizar os endpoints, concentrar validações, aplicar regras de negócio e realizar a persistência dos dados. Essa estrutura contribui para um backend mais modular, seguro e de manutenção mais controlada. Entre os desafios técnicos dessa escolha estão a configuração da conexão com o PostgreSQL, o mapeamento correto das entidades, o tratamento de erros, a validação das requisições e a preparação da aplicação para execução local ou em ambiente de nuvem.

A utilização conjunta do Firebase e do PostgreSQL ocorre porque essas tecnologias atendem a necessidades distintas dentro da arquitetura. O Firebase Authentication é responsável pelo gerenciamento da identidade do usuário, controlando credenciais, login, recuperação de senha e sessão autenticada, sem exigir que a aplicação implemente manualmente mecanismos sensíveis de segurança. O PostgreSQL, por sua vez, armazena os dados relacionais e estruturados do sistema, como usuários, bairros, perfis de acessibilidade, barreiras, referências de mídias e interações realizadas pelos usuários. Assim, o Firebase não substitui o banco relacional, pois sua função principal está associada à autenticação; da mesma forma, o PostgreSQL não substitui diretamente o Firebase Authentication, pois exigiria a implementação própria de recursos de segurança, criptografia, recuperação de senha e controle de sessão.

O Firebase Storage é utilizado de forma complementar para armazenar arquivos de mídia, como imagens e vídeos vinculados às barreiras cadastradas. Nesse caso, o banco relacional mantém apenas as referências necessárias para recuperação dos arquivos, evitando o armazenamento direto de mídias pesadas no PostgreSQL. Essa estratégia torna a arquitetura mais organizada, pois separa credenciais, arquivos e dados estruturados em serviços adequados às suas respectivas finalidades.

Essa combinação apresenta vantagens importantes, como maior especialização das responsabilidades, melhor organização dos dados e aproveitamento de serviços gerenciados para autenticação e armazenamento de arquivos. Entretanto, também impõe limitações técnicas. É necessário manter consistência entre o usuário autenticado no Firebase e seu respectivo registro no PostgreSQL, além de garantir que as mídias enviadas ao Firebase Storage sejam corretamente vinculadas às barreiras armazenadas no banco. Também há maior complexidade na configuração das integrações, no controle de permissões e no tratamento de falhas entre serviços distintos.

O Google Maps SDK para Android compõe a camada de serviços integrados e fornece os recursos cartográficos necessários à funcionalidade central do sistema. Por meio dele, são exibidos o mapa, a localização do usuário, os marcadores das barreiras e a seleção de pontos para novos cadastros. Sua integração com o fluxo da aplicação permite que o registro das barreiras seja feito de forma georreferenciada, associando cada ocorrência a uma latitude e longitude específicas. Essa dependência também exige cuidados técnicos, como configuração da chave de API, controle de permissões de localização e validação dos pontos selecionados no mapa.

Em síntese, a arquitetura adotada busca equilibrar organização, segurança e escalabilidade. O uso do MVVM melhora a estrutura interna do aplicativo; a API REST com Spring Boot centraliza o processamento e protege o acesso ao banco; o PostgreSQL garante persistência relacional

consistente; o Firebase simplifica autenticação e armazenamento de mídias; e o Google Maps SDK viabiliza a interação espacial com as barreiras urbanas. Embora essa composição aumente a complexidade de integração, ela oferece uma base mais adequada para a evolução do sistema, permitindo a inclusão de novas funcionalidades sem comprometer a organização geral da solução.

3.3.1 Detalhamento da API

A API implementada atua como camada intermediária entre o aplicativo Android e o banco de dados PostgreSQL. Desenvolvida com Spring Boot, ela disponibiliza endpoints REST responsáveis pelo cadastro e pela consulta de usuários, carregamento de bairros e perfis de acessibilidade, gerenciamento das barreiras urbanas, registro das mídias associadas e processamento das interações colaborativas realizadas pelos usuários. Essa estrutura impede que o aplicativo Android acesse diretamente o banco de dados e mantém separadas as responsabilidades de interface, processamento das requisições, aplicação das regras de negócio e persistência das informações.

A comunicação entre o aplicativo e a API ocorre por meio de requisições HTTP, utilizando dados estruturados em formato JSON. No aplicativo Android, as chamadas são executadas pelas classes de comunicação HTTP e pelos DAOs, enquanto os Repositories e ViewModels coordenam o fluxo das operações e disponibilizam os resultados para as Activities e os Fragments. No backend, o AccessUrbanController recebe as solicitações, valida os dados informados e encaminha as operações aos repositórios responsáveis pelo acesso ao PostgreSQL.

O Firebase Authentication permanece responsável pelo cadastro das credenciais, autenticação, manutenção da sessão e recuperação de senha. Na versão atual, o vínculo entre a conta autenticada e os dados armazenados no PostgreSQL é realizado por meio do firebaseUid. Esse identificador é utilizado nas operações relacionadas ao perfil do usuário, às barreiras cadastradas e às interações colaborativas. As rotas de consulta de bairros, perfis de acessibilidade e verificação prévia de e-mail não exigem identificação do usuário.

A API também executa validações relacionadas à integridade dos dados. Entre essas validações estão a verificação de e-mails duplicados, a confirmação da existência dos bairros e perfis selecionados, a busca de usuários ativos pelo identificador do Firebase, a atualização somente dos campos cadastrais permitidos e o controle das operações realizadas sobre as barreiras. O endereço de e-mail não é alterado pelo endpoint de edição cadastral, pois permanece vinculado à conta utilizada no Firebase Authentication.

Principais endpoints da API Spring Boot

Método	Endpoint	Finalidade
GET	/api/perfis-acessibilidade	Retorna os perfis de acessibilidade ativos

		utilizados nos formulários de cadastro e edição do usuário.
GET	/api/bairros	Retorna a lista de bairros utilizada nos campos de cadastro, edição de perfil e demais recursos da aplicação.
GET	/api/usuarios/existe-email?email={email}	Verifica se o endereço de e-mail informado já possui cadastro antes da criação da conta no Firebase Authentication.
POST	/api/usuários	Registra no PostgreSQL os dados complementares do usuário após a criação da conta no Firebase Authentication.
GET	/api/usuarios/firebase/{firebaseUid}	Consulta os dados cadastrais do usuário para preenchimento do cabeçalho do Navigation Drawer e da tela de edição de perfil.
PUT	/api/usuarios/firebase/{firebaseUid}	Atualiza os dados cadastrais permitidos, como nome, telefone, bairro e perfil de acessibilidade, sem alterar o e-mail.
POST	/api/usuarios/ultimo-acesso	Registra a data e o horário do último acesso após a autenticação do usuário.
GET	/api/barreiras	Retorna as barreiras ativas, suas classificações, localizações, mídias e totais de interações para exibição no mapa.
POST	/api/barreiras	Registra uma nova barreira urbana com os dados informados pelo usuário e a localização selecionada no mapa.
PUT	/api/barreiras/{idBarreira}	Atualiza os dados de uma barreira cadastrada pelo próprio usuário.

DELETE	/api/barreiras/{idBarreira}	Exclui ou desativa uma barreira cadastrada pelo usuário, removendo-a das consultas ativas do aplicativo.
GET	/api/usuarios/firebase/{firebaseUid}/barreiras	Retorna somente as barreiras cadastradas pelo usuário autenticado para exibição no MyBarriesFragment.
POST	/api/barreiras/{idBarreira}/midias	Associa uma nova imagem ou um novo vídeo a uma barreira existente, preservando as mídias cadastradas anteriormente.
POST	/api/barreiras/{idBarreira}/interacoes	Registra a confirmação de existência ou o registro de uma barreira e retorna os totais atualizados.
GET	/api/barreiras/{idBarreira}/interacoes/resumo	Retorna os totais atuais de confirmações e registros de barreiras para atualização do painel de detalhes da barreira.

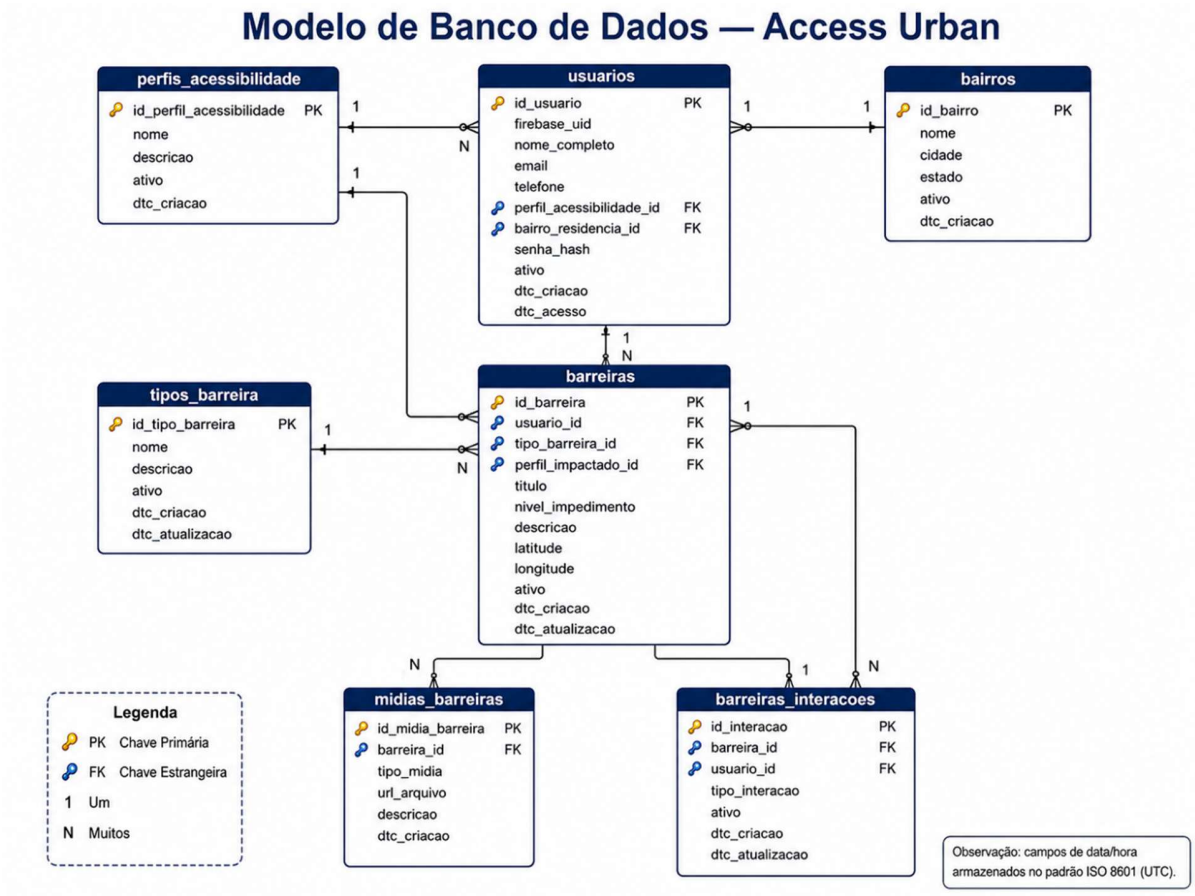
Os endpoints relacionados aos usuários permitem realizar a validação de e-mail antes do cadastro, persistir os dados complementares, consultar o perfil por meio do identificador do Firebase e atualizar somente os campos autorizados. As operações relacionadas às barreiras possibilitam listar as ocorrências ativas, recuperar as barreiras pertencentes ao usuário, adicionar novas mídias e manter as informações apresentadas no mapa e no MyBarriesFragment.

As interações colaborativas são processadas pelo endpoint de confirmação e registro de barreira. Qualquer usuário autenticado pode informar que uma barreira permanece existente ou reportar o registro quando identificar informações incorretas ou desatualizadas. Após o registro, a API consolida os totais e devolve os valores atualizados ao aplicativo. Enquanto o painel de detalhes permanece aberto, o endpoint de resumo é consultado periodicamente para atualizar os contadores sem exigir o fechamento da tela ou o recarregamento completo das barreiras.

As respostas da API utilizam os códigos HTTP correspondentes ao resultado da operação e retornam dados ou mensagens em formato JSON. As falhas de validação, inconsistências de negócio e erros de processamento são tratadas pelo ApiExceptionHandler, permitindo que o aplicativo converta as respostas em estados de carregamento, sucesso ou erro. Esses estados são observados pelos ViewModels e apresentados ao usuário por meio de mensagens claras e adequadas ao contexto da operação.

3.4 Modelo de Banco de Dados

Figura 6 - Modelo de banco de dados



Fonte: Elaborado pelo autor (2026).

O modelo de banco de dados da versão atual do Access Urban foi estruturado para armazenar os dados cadastrais dos usuários, os catálogos utilizados pela aplicação, as barreiras urbanas registradas no mapa, as referências das mídias e as interações colaborativas. A autenticação permanece sob responsabilidade do Firebase Authentication, enquanto o PostgreSQL mantém os dados estruturados necessários ao funcionamento do sistema. Dessa forma, o campo firebase_uid estabelece o vínculo entre a conta autenticada no Firebase e o registro correspondente na tabela de usuários.

Em relação ao modelo apresentado anteriormente, foram removidas as estruturas de notificação a órgãos públicos e de sincronização offline, pois essas funcionalidades não fazem parte da versão implementada. Também não são utilizadas tabelas específicas para tipos de usuário impactado e níveis de impedimento. Na implementação atual, o público impactado é representado por uma chave estrangeira para perfis_acessibilidade, enquanto o nível de impedimento é armazenado diretamente na tabela barreiras com os valores baixo, médio e alto.

Usuários (usuarios)

A tabela usuarios armazena os dados complementares dos usuários cadastrados, incluindo nome completo, e-mail, telefone, perfil de acessibilidade, bairro de residência, situação do cadastro e datas de controle. O campo firebase_uid possui restrição de unicidade e permite relacionar o usuário do PostgreSQL à conta gerenciada pelo Firebase Authentication. O campo senha_hash foi mantido apenas por compatibilidade com versões anteriores e aceita valor nulo, pois as senhas não são utilizadas pelo banco de dados na autenticação atual.

Perfis de Acessibilidade (perfis_acessibilidade)

A tabela perfis_acessibilidade mantém o catálogo de perfis utilizados no cadastro e na edição do usuário. A mesma estrutura também é utilizada para identificar o público impactado por uma barreira, por meio do campo perfil_impactado_id da tabela barreiras. Essa reutilização evita a duplicação de categorias e mantém a classificação padronizada em toda a aplicação.

Bairros (bairros)

A tabela bairros armazena os bairros da cidade de Salvador utilizados nos formulários de cadastro e edição de perfil. Cada registro contém nome, cidade, estado, situação ativa e data de criação. A combinação entre nome, cidade e estado é única, impedindo a duplicação do mesmo bairro dentro da mesma localidade.

Tipos de Barreira (tipos_barreira)

A tabela tipos_barreira mantém as categorias disponíveis para classificação das ocorrências, como barreira arquitetônica e barreira atitudinal. Além do nome e da descrição, a tabela possui campos de controle de atividade, criação e atualização, permitindo que novas categorias sejam incorporadas sem alterar a estrutura principal da tabela barreiras.

Barreiras (barreiras)

A tabela barreiras constitui a entidade central do mapeamento urbano. Cada registro pertence ao usuário que realizou o cadastro e está associado a um tipo de barreira e a um perfil de acessibilidade impactado. A entidade armazena título, nível de impedimento, descrição, latitude, longitude, situação ativa e datas de criação e atualização. O nível de impedimento é limitado aos valores baixo, médio e alto, utilizados pelo aplicativo para representar os marcadores nas cores amarela, laranja e vermelha. A exclusão realizada pelo aplicativo é tratada de forma lógica por meio do campo ativo, preservando o histórico do registro.

Mídias das Barreiras (midias_barreiras)

A tabela midias_barreiras registra os metadados das imagens e dos vídeos associados às ocorrências. Os arquivos são armazenados no Firebase Storage, enquanto o PostgreSQL

mantém a URL de acesso, o tipo da mídia, uma descrição opcional e a data de criação. O relacionamento de um para muitos permite que uma mesma barreira possua várias imagens ou vídeos, inclusive quando novas mídias são adicionadas durante a edição.

Interações nas Barreiras (barreiras_interacoes)

A tabela barreiras_interacoes registra as avaliações colaborativas realizadas pelos usuários sobre as barreiras cadastradas. Cada interação relaciona um usuário a uma barreira e identifica se a ação corresponde a uma confirmação de existência ou reportar inconsistência na barreira avaliada, representado no banco pelo campo tipo_interacao. Os campos ativo, dtc_criacao e dtc_atualizacao permitem controlar a situação e o histórico da interação. A restrição de unicidade composta por barreira, usuário e tipo de interação evita a duplicação do mesmo registro.

4. Testes de Software

4.1 Projeto de Teste

O projeto de teste do foi elaborado com o objetivo de verificar se as funcionalidades atualmente implementadas atendem aos requisitos funcionais e não funcionais definidos para o sistema. Os testes buscam avaliar a estabilidade, a segurança, a confiabilidade, o desempenho e a usabilidade da aplicação, considerando a integração entre o aplicativo Android, a arquitetura MVVM, o Firebase Authentication, o Firebase Storage, o Google Maps SDK para Android, a API REST desenvolvida com Spring Boot e o banco de dados PostgreSQL.

A primeira etapa dos testes corresponde ao processo de autenticação. Nessa fase, são avaliados o login com credenciais válidas, a tentativa de acesso com e-mail ou senha incorretos, o comportamento diante de campos obrigatórios não preenchidos e o redirecionamento após a autenticação bem-sucedida. Quando as credenciais são consideradas válidas pelo Firebase Authentication, o usuário deve ser encaminhado para a HomeActivity, onde são carregadas as informações complementares armazenadas no PostgreSQL.

Também deve ser verificada a manutenção da sessão do usuário. Caso exista uma sessão válida no Firebase Authentication, o sistema deve permitir o acesso à área interna da aplicação sem exigir uma nova autenticação. Quando não houver uma sessão ativa, o usuário deve permanecer na MainActivity, com acesso apenas às opções de login, cadastro e recuperação de senha.

O processo de cadastro de usuários precisa ser testado por meio da RegisterActivity. Nessa etapa, são verificadas as validações dos campos obrigatórios, a seleção do bairro e do perfil de acessibilidade, o formato do e-mail, o preenchimento da senha e a confirmação dos dados antes do envio. O sistema também deve impedir a criação de contas duplicadas por meio da verificação prévia da existência do e-mail na API.

Quando o endereço informado já estiver cadastrado, o sistema deve apresentar uma mensagem

adequada e interromper o processo antes da criação de uma nova conta no Firebase Authentication. Essa validação contribui para evitar inconsistências entre os dados mantidos pelo serviço de autenticação e os registros armazenados no PostgreSQL.

Nos testes de cadastro bem-sucedido, deve ser verificado se a conta é criada corretamente no Firebase Authentication e se os dados complementares do usuário, incluindo nome completo, telefone, bairro, perfil de acessibilidade e firebase_uid, são enviados para a API e persistidos no banco de dados. Após a conclusão, a aplicação deve retornar à tela de login, permitindo que o usuário utilize as credenciais cadastradas.

O comportamento da opção de retorno também deve ser validado. Ao sair da tela de cadastro antes da conclusão do processo, o sistema deve retornar para a MainActivity sem criar uma conta no Firebase Authentication e sem armazenar dados incompletos no PostgreSQL.

A funcionalidade de recuperação de senha deve ser testada a partir da opção disponível na tela de login. O sistema deve solicitar o e-mail do usuário, encaminhar a solicitação ao Firebase Authentication e informar que as instruções de redefinição foram enviadas. Também devem ser avaliados os comportamentos relacionados a campos vazios, formato inválido de e-mail e falhas temporárias de comunicação.

A HomeActivity, responsável pela navegação principal do sistema, deve ser testada após a autenticação. Devem ser verificadas a abertura correta da tela, a exibição do Navigation Drawer, o carregamento das informações do usuário no cabeçalho e a navegação entre as funcionalidades disponíveis. O cabeçalho deve apresentar os dados recuperados por meio da API, como nome, e-mail e bairro de residência.

O EditProfileFragment precisa ser testado quanto ao carregamento e à atualização dos dados cadastrais. Os campos devem ser preenchidos com as informações atuais do usuário, permitindo a alteração do nome completo, telefone, bairro e perfil de acessibilidade. O endereço de e-mail deve permanecer visível, porém bloqueado para edição, pois corresponde à credencial vinculada ao Firebase Authentication.

Após a confirmação das alterações, o sistema deve enviar os dados para a API, atualizar o registro correspondente no PostgreSQL e refletir imediatamente as novas informações no cabeçalho do Navigation Drawer. Também devem ser avaliadas situações em que os campos obrigatórios estejam vazios ou contenham informações inválidas.

O HomeFragment, definido como tela padrão da aplicação, deve ser testado quanto à integração com o Google Maps SDK. Devem ser avaliados o carregamento do mapa da cidade de Salvador, a centralização inicial da visualização, a exibição dos marcadores correspondentes às barreiras cadastradas e a atualização das informações apresentadas.

Os marcadores precisam utilizar cores relacionadas ao nível de impedimento de cada ocorrência. As barreiras classificadas com nível baixo devem ser representadas pela cor amarela, as de nível

médio pela cor laranja e as de nível alto pela cor vermelha. Essa diferenciação deve permanecer visível e coerente após o carregamento, cadastro ou atualização das ocorrências.

Os recursos de busca e filtragem também devem ser avaliados. A busca deve permitir localizar bairros, endereços e locais disponíveis na cidade de Salvador, atualizando a posição exibida no mapa. Os filtros devem restringir as barreiras apresentadas de acordo com os critérios disponíveis, sem alterar ou excluir os dados armazenados.

A funcionalidade de localização atual necessita ser testada mediante a concessão e a recusa das permissões necessárias. Quando a autorização for concedida, a aplicação deve localizar o dispositivo e centralizar o mapa adequadamente. Caso a permissão seja negada, o sistema deve permanecer estável e apresentar uma orientação compatível com a situação, sem provocar o encerramento inesperado da aplicação.

O fluxo de cadastro de barreiras deve ser iniciado pela seleção de um ponto no mapa. Ao tocar em uma localização válida, o sistema deve exibir um marcador temporário e informar que o local foi selecionado. O acesso ao formulário deve ocorrer por meio do botão flutuante de cadastro. Caso nenhum ponto tenha sido selecionado, o sistema deve impedir a abertura indevida do formulário e orientar o usuário.

Também deve ser testada a validação territorial da localização. O sistema deve permitir o cadastro somente quando o ponto selecionado estiver dentro da área definida para a cidade de Salvador. Tentativas de selecionar locais fora da região permitida ou em posições incompatíveis com a proposta da aplicação devem ser tratadas adequadamente.

No formulário de cadastro, precisam ser verificadas as validações dos campos utilizados na versão atual do sistema, incluindo título, tipo da barreira, tipo de usuário impactado, nível de impedimento, descrição, localização geográfica e mídia. O sistema deve impedir o envio quando informações obrigatórias não forem preenchidas.

Os testes relacionados às mídias devem verificar a seleção e o envio de imagens ou vídeos para o Firebase Storage. Após o upload, a referência correspondente deve ser associada à barreira por meio da API. Também devem ser avaliadas situações de falha durante o envio, indisponibilidade de conexão e interrupção do processo.

Após a conclusão do cadastro, a nova barreira precisa ser persistida no PostgreSQL, exibida no mapa e disponibilizada na área destinada às barreiras do usuário. A atualização da interface deve ocorrer sem exigir uma nova autenticação ou o reinício completo da aplicação.

A visualização dos detalhes das barreiras também deve ser testada. Ao tocar em um marcador existente, o sistema deve apresentar as informações cadastradas, incluindo título, tipo da barreira, público impactado, nível de impedimento, descrição e mídias associadas. As imagens e os vídeos devem ser recuperados corretamente a partir das referências armazenadas.

Os mecanismos de interação colaborativa necessitam ser avaliados por meio das funcionalidades de confirmar e reportar barreira cadastrada. Um usuário autenticado deve conseguir confirmar que uma barreira permanece existente ou reportar um registro que apresente informações incorretas ou desatualizadas. Após a interação, os contadores devem ser atualizados e apresentados na interface.

Também deve ser verificado o comportamento quando o usuário altera o tipo de interação realizada. O sistema deve manter a consistência dos dados e evitar registros conflitantes para a mesma barreira e o mesmo usuário. As informações atualizadas devem permanecer disponíveis após o fechamento e a reabertura da tela de detalhes.

O MyBarriesFragment requer ser testado quanto ao carregamento exclusivo das barreiras pertencentes ao usuário autenticado. A lista não deve apresentar registros cadastrados por outros usuários. Cada item deve permitir o acesso aos detalhes e disponibilizar as ações de edição e exclusão conforme as regras atuais da aplicação.

Nos testes de edição, deve ser verificado se os dados atuais da barreira são carregados corretamente e se as alterações são enviadas para a API. Após a conclusão, as informações atualizadas devem ser refletidas no MyBarriesFragment, no mapa e na tela de detalhes.

A exclusão deve ser validada quanto à confirmação da ação, ao envio da solicitação para a API e à remoção do registro das áreas ativas da aplicação. A barreira excluída não deve continuar disponível no mapa nem na lista do usuário após a atualização dos dados.

A funcionalidade de logout também precisa ser validada. Ao selecionar a opção Sair, a aplicação deve encerrar a sessão mantida pelo Firebase Authentication e retornar para a MainActivity. Após o encerramento, o usuário não deve conseguir acessar novamente a área interna sem realizar uma nova autenticação.

Além dos testes funcionais, o projeto contempla verificações não funcionais relacionadas à segurança. Deve ser confirmado que as senhas não são armazenadas no PostgreSQL e que o gerenciamento das credenciais permanece sob responsabilidade do Firebase Authentication. Também deve ser avaliado se as operações de edição e exclusão de barreiras são disponibilizadas somente para o responsável pelo cadastro.

Os testes de desempenho devem observar o tempo necessário para realizar login, carregar os dados do usuário, consultar a API, exibir o mapa, apresentar os marcadores, aplicar filtros, cadastrar barreiras, enviar mídias e atualizar as listas. O aplicativo deve permanecer responsivo durante as operações assíncronas, evitando o bloqueio da interface.

Nos testes de usabilidade, precisam ser avaliadas a clareza das mensagens de erro e sucesso, a organização dos formulários, a legibilidade dos textos, a identificação dos botões, a facilidade de utilização do Navigation Drawer e a compreensão das ações disponíveis no mapa. Os feedbacks apresentados durante carregamentos e operações devem permitir que o usuário

compreenda o estado atual do sistema.

Também devem ser realizados testes de confiabilidade relacionados a falhas de comunicação. A aplicação deve tratar adequadamente situações em que a API, o Firebase Storage, o Firebase Authentication ou os serviços do Google Maps estejam temporariamente indisponíveis. Essas condições não devem provocar encerramentos inesperados, e o usuário deve receber mensagens compatíveis com a falha identificada.

Por fim, os resultados obtidos devem ser comparados com os requisitos definidos para a aplicação. Cada funcionalidade deve ser considerada aprovada quando apresentar o comportamento esperado, armazenar ou recuperar corretamente as informações e manter a integração entre o aplicativo Android, os serviços Firebase, a API Spring Boot, o Google Maps SDK e o PostgreSQL.

5. Implantação

5.1 Projeto de Implantação

O Access Urban foi desenvolvido com base em uma arquitetura distribuída e organizada em camadas integradas. A solução atualmente implementada é composta pelo aplicativo Android, pelos serviços Firebase Authentication e Firebase Storage, pelo Google Maps SDK para Android, pela API REST desenvolvida com Spring Boot e pelo banco de dados PostgreSQL. A separação entre esses componentes reduz o acoplamento entre a interface, a autenticação, as regras da aplicação, o armazenamento de mídias e a persistência dos dados, contribuindo para maior organização do código, facilidade de manutenção e evolução das funcionalidades.

A primeira etapa da implantação consiste na preparação do banco de dados PostgreSQL. Nessa fase, é criado o banco utilizado pelo Access Urban e são executados os scripts responsáveis pela criação das tabelas, chaves primárias, chaves estrangeiras, restrições e índices necessários ao funcionamento da aplicação. O banco armazena os dados complementares dos usuários, os bairros, os perfis de acessibilidade, as barreiras urbanas cadastradas, as referências das mídias e as interações realizadas nas ocorrências.

A autenticação dos usuários não é executada pelo PostgreSQL. Essa responsabilidade pertence ao Firebase Authentication. Por esse motivo, a tabela de usuários mantém o campo firebase_uid, utilizado para relacionar o cadastro armazenado no banco de dados à conta criada no Firebase. As senhas não são armazenadas no PostgreSQL, pois o gerenciamento das credenciais permanece sob responsabilidade do serviço de autenticação.

Após a criação e configuração do banco de dados, realiza-se a implantação da API REST desenvolvida com Spring Boot.

Essa estrutura impede que o aplicativo Android estabeleça conexão direta com o banco de dados. Dessa forma, as credenciais de acesso ao PostgreSQL permanecem configuradas

exclusivamente no backend, reduzindo a exposição de informações sensíveis e mantendo uma separação adequada entre a aplicação móvel e a camada de persistência.

Também são disponibilizados os endpoints utilizados para consultar bairros e perfis de acessibilidade, verificar a existência de e-mails cadastrados, registrar usuários, recuperar dados por meio do firebase_uid, atualizar informações cadastrais, registrar o último acesso, cadastrar barreiras, listar as ocorrências exibidas no mapa, consultar as barreiras pertencentes ao usuário, editar registros, excluir barreiras, associar mídias e processar confirmações e registros de barreiras.

As respostas fornecidas pela API são utilizadas pelo aplicativo para identificar o resultado das operações. Dessa forma, os ViewModels podem controlar estados de carregamento, sucesso e erro, permitindo que a interface apresente mensagens adequadas ao usuário. As validações realizadas pelo backend contribuem para a integridade das informações e evitam operações inconsistentes no banco de dados.

A etapa seguinte envolve a configuração dos serviços Firebase utilizados pelo Access Urban. O Firebase Authentication é responsável pela criação das contas, realização do login, manutenção da sessão, recuperação de senha e encerramento do acesso do usuário.

Durante o cadastro, o aplicativo verifica inicialmente se o endereço de e-mail informado já está registrado. Após a criação da conta no Firebase Authentication, os dados complementares, como nome completo, telefone, bairro, perfil de acessibilidade e firebase_uid, são enviados para a API e armazenados no PostgreSQL. Ao concluir o cadastro, o usuário retorna à tela de login para acessar a aplicação.

A funcionalidade de recuperação de senha também utiliza o Firebase Authentication. Ao selecionar a opção correspondente, o usuário informa o endereço de e-mail cadastrado e recebe as orientações necessárias para redefinir sua senha. Após a alteração, a nova credencial pode ser utilizada normalmente na tela de login.

Quando uma mídia é selecionada durante o cadastro ou a atualização de uma ocorrência, o arquivo é enviado para o serviço de armazenamento. A referência gerada é posteriormente associada ao registro da barreira, permitindo a recuperação e a exibição da mídia na tela de detalhes.

O PostgreSQL não armazena diretamente os arquivos de imagem ou vídeo. O banco mantém somente as informações necessárias para identificar e recuperar a mídia armazenada no Firebase Storage.

O arquivo AndroidManifest.xml deve conter as permissões necessárias para o funcionamento das funcionalidades implementadas, incluindo acesso à internet e à localização do dispositivo. As permissões relacionadas à localização são solicitadas durante a execução quando o usuário utiliza os recursos de posicionamento atual. O acesso às mídias também deve seguir as regras

aplicáveis à versão do Android utilizada no dispositivo.

A integração com o Google Maps SDK possibilita a exibição do mapa da cidade de Salvador no HomeFragment. Para isso, a chave da API do Google Maps deve estar configurada no projeto Android. O mapa apresenta os marcadores das barreiras urbanas cadastradas e utiliza cores diferentes para representar os níveis de impedimento baixo, médio e alto.

A HomeActivity representa a tela principal da aplicação e utiliza um Navigation Drawer para organizar a navegação. Após a autenticação, o usuário é direcionado para essa tela, onde o cabeçalho do menu lateral apresenta informações relacionadas ao cadastro, como nome, e-mail e bairro.

As funcionalidades internas são organizadas por fragments. O HomeFragment, definido como tela inicial, apresenta o mapa da cidade de Salvador e disponibiliza recursos de busca, filtragem, localização atual, visualização dos marcadores e consulta dos detalhes das barreiras. O usuário também pode selecionar um ponto no mapa para iniciar o cadastro de uma nova ocorrência.

Após a seleção do local, a aplicação exibe um marcador temporário e permite o acesso ao formulário de cadastro. O sistema verifica se a localização informada está dentro da área permitida da cidade de Salvador antes de prosseguir. No formulário, o usuário informa o título, o tipo da barreira, o público impactado, o nível de impedimento, a descrição e, quando necessário, adiciona uma imagem ou um vídeo.

O MyBarriesFragment apresenta somente as barreiras cadastradas pelo usuário autenticado. Nessa área, é possível visualizar os detalhes, editar as informações ou excluir uma ocorrência. As alterações realizadas são enviadas para a API e refletidas na lista e no mapa.

O EditProfileFragment permite consultar e atualizar os dados cadastrais autorizados pelo sistema, incluindo nome completo, telefone, bairro e perfil de acessibilidade. O endereço de e-mail permanece bloqueado para edição. Após o salvamento, as informações são atualizadas no PostgreSQL e refletidas no cabeçalho do Navigation Drawer.

O SobreFragment apresenta informações institucionais relacionadas ao Access Urban, incluindo a finalidade, os objetivos e os conceitos de acessibilidade urbana associados ao projeto. A opção Sair encerra a sessão mantida pelo Firebase Authentication e direciona o usuário novamente para a MainActivity.

Dessa forma, o projeto de implantação representa a integração efetivamente utilizada entre o aplicativo Android, a API Spring Boot, o banco PostgreSQL, o Firebase Authentication, o Firebase Storage e o Google Maps SDK. A arquitetura adotada mantém separadas as responsabilidades de interface, autenticação, processamento das regras, armazenamento de mídias e persistência dos dados, proporcionando uma estrutura organizada e compatível com as funcionalidades atualmente implementadas.

6. Manual do Usuário

O Access Urban permite realizar cadastro e autenticação de usuários, recuperar senha, consultar e atualizar dados cadastrais, visualizar barreiras em um mapa interativo, pesquisar locais, aplicar filtros, cadastrar novas ocorrências, anexar imagens ou vídeos, consultar detalhes, confirmar a permanência de uma barreira, reportar informações incorretas e gerenciar os registros criados pelo próprio usuário.

Para utilizar os recursos do Access Urban, o dispositivo deve possuir conexão com a internet. A conectividade é necessária para autenticação, carregamento dos dados, comunicação com a API, exibição do mapa, envio de mídias e atualização das informações das barreiras.

- Utilizar um dispositivo Android compatível com a versão definida no projeto.
- Manter a conexão com a internet ativa durante as operações de login, cadastro, consulta e atualização.
- Conceder a permissão de localização quando desejar utilizar o recurso de centralização da posição atual.
- Conceder as permissões solicitadas pelo Android ao selecionar imagens ou vídeos para uma barreira.
- Preencher os campos obrigatórios conforme as orientações apresentadas em cada formulário.
- Aguardar a conclusão das operações antes de fechar a tela ou encerrar o aplicativo.

6.1 Tela de Login

A Tela de Login é o ponto inicial de acesso ao Access Urban. Nela, o usuário informa as credenciais cadastradas no Firebase Authentication, pode optar por lembrar a senha no dispositivo, solicitar a redefinição da credencial ou abrir o formulário de cadastro de uma nova conta.

Componentes da tela

Componente	Finalidade
Logomarca do Access Urban	Identifica visualmente o aplicativo.
Campo E-mail	Recebe o endereço de e-mail utilizado no cadastro.
Campo Senha	Recebe a senha da conta do usuário.
Ícone de visualização da senha	Alterna entre ocultar e exibir temporariamente os caracteres informados.
Caixa Lembrar senha	Mantém os dados definidos pela funcionalidade de lembrança para facilitar acessos posteriores no mesmo dispositivo.

Opção Esqueci minha senha	Inicia o fluxo de redefinição de senha por e-mail.
Botão LOGIN	Valida as credenciais e direciona o usuário para a tela principal.
Botão CADASTRE-SE	Abre a Tela de Cadastro para criação de uma nova conta.

Passo a passo

1. Digite no campo E-mail o endereço utilizado no cadastro.

2. Digite a senha no campo correspondente.

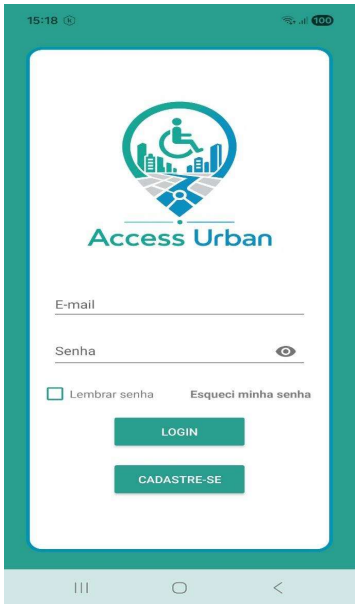
3. Toque no ícone de olho para conferir a senha, quando necessário.

4. Marque Lembrar senha somente quando desejar utilizar o recurso no dispositivo atual.

5. Toque em LOGIN.

6. Aguarde a autenticação. Quando os dados estiverem corretos, a HomeActivity será aberta e o mapa será exibido.

Figura 7 – Tela de Login



6.1.1 Recuperação de senha

A recuperação de senha é iniciada na própria Tela de Login e utiliza o endereço de e-mail associado à conta do usuário.

1. Toque em Esqueci minha senha.

2. Informe o e-mail cadastrado quando o aplicativo apresentar a solicitação.

3. Confirme o envio da solicitação.

4. Acesse a caixa de entrada do e-mail informado e procure a mensagem enviada pelo Firebase Authentication.

5. Abra o link recebido e defina uma nova senha conforme as orientações apresentadas.

6. Retorne ao Access Urban e utilize a nova senha para realizar o login.

Resultado esperado

Ao concluir o procedimento pelo link recebido, a senha da conta será atualizada no Firebase Authentication. O usuário poderá retornar à Tela de Login e utilizar a nova credencial, mantendo os dados cadastrais e as barreiras anteriormente associadas ao seu perfil.

Situações que podem impedir a redefinição

• E-mail digitado em formato inválido.

• Falha temporária de conexão com a internet.

• Mensagem de redefinição não localizada nas pastas do provedor de e-mail.

• Link de redefinição expirado ou já utilizado. Nesse caso, solicite um novo envio pelo aplicativo.

6.2 Cadastro de usuário

A Tela de Cadastro permite criar uma conta no Access Urban. O e-mail e a senha são gerenciados pelo Firebase Authentication, enquanto os dados complementares do usuário são enviados para a API e armazenados no banco de dados PostgreSQL.

Componentes da tela

Componente	Finalidade
Campo Nome completo	Recebe o nome completo do usuário.
Campo E-mail	Recebe o endereço que será utilizado para autenticação.
Campo Telefone	Recebe o número de contato do usuário.
Lista Perfil de acessibilidade	Permite selecionar o perfil de acessibilidade associado ao cadastro.
Lista Bairro	Permite selecionar o bairro de residência.
Campo Senha	Recebe a senha de acesso à conta.
Campo Confirmar senha	Repete a senha para reduzir erros de digitação.

Ícones de visualização	Permitem exibir ou ocultar temporariamente as senhas informadas.
Botão CADASTRAR	Valida os dados e inicia a criação da conta.
Botão VOLTAR	Retorna à Tela de Login sem concluir o cadastro.

Passo a passo

1. Na Tela de Login, toque em CADASTRE-SE.
2. Informe o nome completo.
3. Digite um endereço de e-mail válido e ainda não cadastrado no Access Urban.
4. Informe o telefone.
5. Abra a lista Perfil de acessibilidade e selecione a opção correspondente.
6. Abra a lista Bairro e selecione o bairro de residência.
7. Crie uma senha e informe-a no campo Senha.
8. Repita exatamente a mesma senha no campo Confirmar senha.
9. Toque em CADASTRAR e aguarde a conclusão.
10. Após o cadastro bem-sucedido, retorne à Tela de Login e utilize as credenciais criadas.

Figura 8 – Cadastro de usuário

6.3 Tela principal e mapa interativo

A Tela Principal é apresentada após a autenticação e utiliza o Google Maps SDK para exibir a cidade de Salvador e as barreiras cadastradas. A interface reúne recursos de pesquisa, filtragem, localização atual, visualização de marcadores e seleção de pontos para o cadastro de novas ocorrências.

Componentes da tela

Componente	Finalidade
Barra superior Home	Identifica a tela atual e disponibiliza o acesso ao menu lateral.
Ícone de menu	Abre o Navigation Drawer com as opções do aplicativo.
Campo de busca	Recebe o nome de um bairro, endereço ou local.
Botão de pesquisa	Executa a busca do conteúdo informado.
Botão de filtro	Abre as opções disponíveis para restringir as barreiras exibidas.
Mensagem de estado do filtro	Indica se todas as barreiras ou somente parte delas está sendo exibida.
Mapa interativo	Apresenta as vias, bairros, pontos de interesse e barreiras urbanas da cidade de Salvador.
Marcadores coloridos	Representam as barreiras conforme o nível de impedimento.
Botão de localização	Centraliza o mapa na posição atual do dispositivo, mediante permissão.
Controles de zoom	Aproximam ou afastam a visualização do mapa.

Passo a passo

1. Após realizar o login, aguarde o carregamento da Tela Principal.

2. Observe os marcadores exibidos no mapa. A cor amarela representa nível baixo, a laranja representa nível médio e a vermelha representa nível alto.

3. Para buscar um local, toque no campo de pesquisa e informe um bairro, endereço ou ponto de interesse.

4. Toque no botão de pesquisa e aguarde a atualização da área exibida no mapa.

5. Para aplicar filtros, toque no botão de filtro, selecione os critérios desejados e confirme a aplicação.

6. Utilize o botão de localização para centralizar o mapa na posição atual. Autorize o acesso à localização quando o Android solicitar.

7. Utilize os controles + e - para ajustar o nível de aproximação.

8. Toque em um marcador existente para abrir os detalhes da barreira.

9. Toque em um ponto válido do mapa quando desejar iniciar o cadastro de uma nova barreira.

Figura 9 – Tela principal e mapa interativo

6.4 Cadastrar barreira

O formulário Cadastrar barreira é utilizado para registrar uma nova ocorrência na localização previamente selecionada no mapa. A barreira recebe informações descritivas, classificação, público impactado e, opcionalmente, uma imagem ou um vídeo.

Componentes da tela

Componente	Finalidade
Campo Título	Recebe uma identificação curta e objetiva para a ocorrência.
Lista Tipo de barreira	Permite selecionar a categoria da barreira.
Lista Tipo de usuário impactado	Identifica o público afetado pela ocorrência.
Lista Nível de impedimento	Classifica o impacto como baixo, médio ou alto.
Campo Descrição	Recebe informações detalhadas sobre a situação encontrada.
Botão SELECIONAR IMAGEM/VÍDEO	Abre o seletor de mídia do dispositivo.
Indicador da mídia	Informa se uma imagem ou um vídeo foi selecionado.

Botão CANCELAR	Fecha o formulário sem concluir o cadastro.
Botão SALVAR	Valida os dados, envia a mídia quando houver e registra a barreira.

Passo a passo

1. Na Tela Principal, toque no ponto do mapa onde a barreira está localizada.

2. Aguarde a exibição do marcador temporário e da orientação de local selecionado.

3. Toque no botão flutuante de cadastro identificado pelo símbolo +.

4. Informe um título curto e representativo.

5. Selecione o tipo de barreira.

6. Selecione o tipo de usuário impactado.

7. Selecione o nível de impedimento: baixo, médio ou alto.

8. Descreva a condição encontrada de forma clara e objetiva.

9. Para adicionar uma evidência, toque em SELECIONAR IMAGEM/VÍDEO e escolha o arquivo no dispositivo.

10. Confira o indicador apresentado abaixo do botão de mídia.

11. Toque em SALVAR e aguarde a conclusão do envio e do cadastro.

12. Após o sucesso, a barreira será disponibilizada no mapa e na área Minhas Barreiras.

Figura 10 – Selecionar local

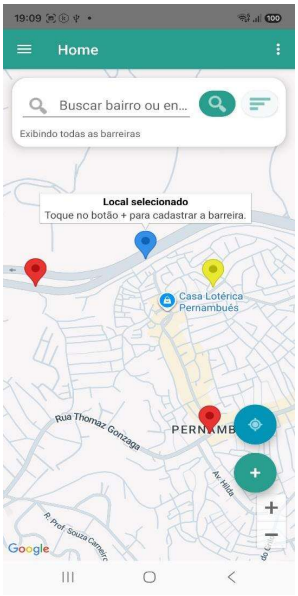
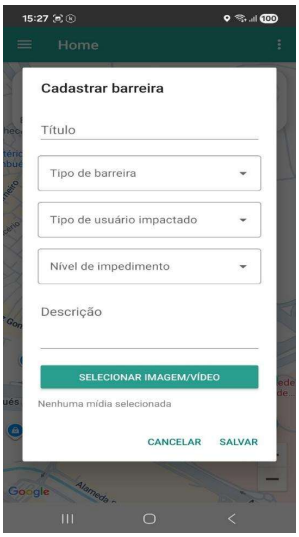


Figura 11 – Cadastrar Barreira



6.5 Visualizar barreira

A Tela de Detalhes apresenta as informações completas de uma barreira selecionada no mapa. Além dos dados descritivos e das mídias, a versão atual disponibiliza recursos de avaliação colaborativa, permitindo confirmar que a ocorrência continua existente ou reportar informações incorretas ou desatualizadas.

Componentes da tela

Componente	Finalidade
Título da barreira	Identifica a ocorrência selecionada.
Tipo	Apresenta a categoria cadastrada.
Impactado	Indica o tipo de usuário afetado.
Nível	Apresenta o nível de impedimento e sua representação por cor.
Descrição	Detalha a condição informada no cadastro.
Seção Mídias cadastradas	Exibe as imagens ou os vídeos associados à barreira.
Área Avaliação da comunidade	Reúne os dados e as ações de validação colaborativa.

Contador de Confirmações	Indica quantos usuários confirmaram que a barreira permanece existente.
Contador de Denúncias/Registros de barreiras	Indica quantos usuários reportaram informações incorretas ou desatualizadas.
Botão AINDA EXISTE	Registra a confirmação da permanência da barreira.
Botão REPORTAR	Registra uma barreira sobre a ocorrência.
Botão FECHAR	Encerra a visualização dos detalhes e retorna ao mapa.

Passo a passo

1. Na Tela Principal, toque no marcador da barreira que deseja consultar.
2. Leia o título, o tipo, o público impactado, o nível de impedimento e a descrição.
3. Consulte as mídias apresentadas na seção Mídias cadastradas.
4. Observe os totais exibidos na área Avaliação da comunidade.
5. Toque em AINDA EXISTE quando tiver confirmado que a barreira permanece no local.
6. Toque em REPORTAR quando identificar dados incorretos, desatualizados ou inadequados.
7. Aguarde a atualização dos contadores.
8. Toque em FECHAR para retornar ao mapa.

Figura 12 – Visualizar barreira

6.6 Menu lateral de navegação

O menu lateral organiza as principais áreas do Access Urban. Ele pode ser aberto pelo ícone de menu localizado na barra superior da Tela Principal. O cabeçalho apresenta informações do usuário autenticado e disponibiliza acesso direto à edição do cadastro.

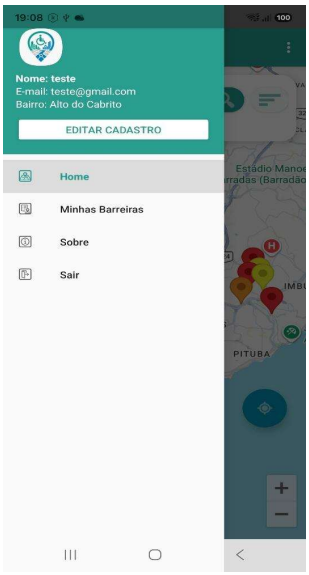
Componentes da tela

Componente	Finalidade
Imagem do usuário	Representa visualmente o perfil no cabeçalho do menu.
Nome	Exibe o nome do usuário autenticado.
E-mail	Exibe o e-mail vinculado à conta.
Bairro	Exibe o bairro de residência armazenado no cadastro.
Botão EDITAR CADASTRO	Abre a tela de consulta e atualização dos dados cadastrais.
Opção Home	Retorna ao mapa e às funcionalidades de busca, filtro e cadastro.
Opção Minhas Barreiras	Abre a lista de barreiras cadastradas pelo próprio usuário.
Opção Sobre	Apresenta informações institucionais e conceituais do projeto.
Opção Sair	Encerra a sessão e retorna à Tela de Login.

Passo a passo

1. Na Tela Principal, toque no ícone de menu localizado no canto superior esquerdo.
2. Confira os dados exibidos no cabeçalho.
3. Toque em EDITAR CADASTRO para atualizar as informações permitidas.
4. Toque em Home para retornar ao mapa.
5. Toque em Minhas Barreiras para consultar e gerenciar seus registros.
6. Toque em Sobre para visualizar informações do Access Urban.
7. Toque em Sair quando desejar encerrar a sessão.

Figura 13 – Menu lateral de navegação



6.7 Editar cadastro

A Tela Editar Cadastro permite consultar e atualizar os dados complementares do usuário. As informações atuais são carregadas automaticamente. O e-mail é exibido somente para consulta e permanece bloqueado para edição por estar associado à credencial gerenciada pelo Firebase Authentication.

Componentes da tela

Componente	Finalidade
Seta de retorno	Volta para a tela anterior sem iniciar uma nova operação.
Campo Nome completo	Permite alterar o nome do usuário.
Campo E-mail bloqueado	Exibe o e-mail da conta sem permitir alteração.
Campo Telefone	Permite atualizar o número de contato.
Lista Perfil de acessibilidade	Permite selecionar outro perfil disponível.
Lista Bairro	Permite atualizar o bairro de residência.
Botão SALVAR ALTERAÇÕES	Envia os novos dados para a API e atualiza o cadastro.

Passo a passo

1. Abra o menu lateral.

2. Toque em EDITAR CADASTRO.

3. Aguarde o carregamento das informações atuais.

4. Altere o nome completo, o telefone, o perfil de acessibilidade ou o bairro, conforme necessário.

5. Observe que o e-mail é exibido em tom desabilitado e não pode ser modificado nessa tela.

6. Toque em SALVAR ALTERAÇÕES.

7. Aguarde a confirmação. Após o salvamento, os dados atualizados serão refletidos no cabeçalho do menu lateral.

Figura 14 – Editar cadastro

6.8 Minhas Barreiras

A Tela Minhas Barreiras apresenta somente as ocorrências cadastradas pelo usuário autenticado. Cada item exibe um resumo das informações e disponibiliza as ações de edição e exclusão. O total de registros é apresentado na parte superior da tela.

Componentes da tela

Componente	Finalidade
Título Minhas Barreiras	Identifica a área de gerenciamento dos registros do usuário.
Total de barreiras cadastradas	Informa a quantidade de ocorrências pertencentes ao usuário.

Cartão da barreira	Agrupa o título, o nível, o público impactado e a descrição resumida.
Botão EDITAR	Abre o formulário com os dados atuais para atualização.
Botão EXCLUIR	Inicia a remoção da barreira selecionada.
Menu lateral	Permite retornar à Home ou acessar as demais áreas do aplicativo.

Passo a passo

1. Abra o menu lateral e toque em Minhas Barreiras.

2. Aguarde o carregamento da lista.

3. Confira o total apresentado no início da tela.

4. Localize a barreira desejada pelo título, nível, público impactado ou descrição.

5. Toque no cartão ou na área correspondente quando desejar consultar os detalhes, conforme o comportamento disponibilizado na versão atual.

6. Toque em EDITAR para abrir os dados da ocorrência e realizar as alterações permitidas.

7. Salve a edição e aguarde a atualização da lista e do mapa.

8. Toque em EXCLUIR quando desejar remover uma ocorrência cadastrada por você.

9. Confirme a exclusão quando o aplicativo solicitar e aguarde a atualização da lista.

Figura 15 – Minhas Barreiras

6.9 Sobre

A opção Sobre apresenta informações institucionais relacionadas ao Access Urban, sua finalidade, seus objetivos e os conceitos vinculados ao mapeamento colaborativo e à acessibilidade urbana.

Passo a passo

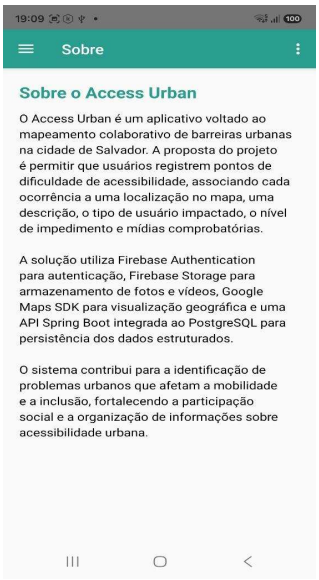
1. Abra o menu lateral.

2. Toque em Sobre.

3. Leia as informações apresentadas na tela.

4. Utilize o menu ou o botão de retorno para acessar outra área do aplicativo.

Figura 16 – Tela Sobre



6.10 Sair

A opção Sair encerra a sessão autenticada no Firebase Authentication e retorna o usuário para a Tela de Login. Esse procedimento deve ser utilizado principalmente ao finalizar o uso do aplicativo em dispositivos compartilhados.

Passo a passo

1. Abra o menu lateral.

2. Toque em Sair.

3. Aguarde o encerramento da sessão.

4. Confirme que a Tela de Login foi apresentada.

5. Para acessar novamente, informe as credenciais e toque em LOGIN.

Agradecimentos

Agradeço, primeiramente, a todos que, de alguma forma, contribuíram para a realização deste Trabalho de Conclusão de Curso, seja por meio de apoio, incentivo, orientação ou palavras de encorajamento ao longo desta caminhada.

Aos meus pais, Maria Lucia Nascimento Oliveira e Eliezer Santos Oliveira, expresso minha profunda gratidão pelo amor, apoio, dedicação e incentivo constantes. A presença, os ensinamentos e a confiança de vocês foram fundamentais para que eu pudesse enfrentar os desafios desta trajetória acadêmica e chegar até aqui.

Ao meu orientador, Pablo Vieira Florentino, agradeço pela orientação, paciência, disponibilidade e pelas contribuições essenciais para o desenvolvimento deste trabalho. Seu acompanhamento foi de grande importância para a construção e aprimoramento desta pesquisa.

A todos os professores, colegas, familiares e amigos que fizeram parte desta jornada, deixo também meus sinceros agradecimentos. Cada contribuição, direta ou indireta, teve valor significativo para a concretização deste trabalho.

Por fim, agradeço a todos que acreditaram em mim e que, de alguma maneira, contribuíram para a realização deste sonho.

Referências

[1] WERNECK, Cláudia. Modelo médico x Modelo social da deficiência. Em: Manual da mídia legal 3: Comunicadores pela Saúde / Escola de Gente – Rio de Janeiro: WVA Editora, 2004, páginas 16 a 20.

[2] WERNECK, Claudia. Manual sobre desenvolvimento inclusivo para a mídia e profissionais de comunicação. Rio de Janeiro: WVA, 2014.

[3] DINIZ, Debora. O que é deficiência. São Paulo: Brasiliense, 2007. (Coleção Primeiros Passos; 324).

[4] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 9050: acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos. Rio de Janeiro: ABNT, 2015.

[5] BRASIL. Lei nº 10.098, de 19 de dezembro de 2000. Estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida, e dá outras providências. Diário Oficial da União, Brasília, DF, 20 dez. 2000.

[6] BRASIL. Decreto nº 5.296, de 2 de dezembro de 2004. Regulamenta as Leis n° 10.048, de 8 de novembro de 2000, que dá prioridade de atendimento às pessoas que especifica, e 10.098, de 19 de dezembro de 2000, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida, e dá outras providências. Diário Oficial da União, Brasília, DF, 3 dez. 2004.

[7] BRASIL. Lei nº 13.146, de 6 de julho de 2015. Institui a Lei Brasileira de Inclusão da Pessoa com Deficiência. Brasília, DF: Presidência da República, 2015.

[8] LIMA, Sâmara Sathler; CARVALHO-FREITAS, Maria Nivalda; SANTOS, Larissa Medeiros. Repercussões psicossociais da acessibilidade urbana para as pessoas com deficiência física. Periódico PSICO, Porto Alegre, PUCRS, v.44, n.3, pp 362 – 371, jul. /Set 2013.

[9] VASCONCELLOS, Milton Silva. Salvador cidade deficiente: o acesso às praias para pessoas com deficiência física. Cadernos do CEAS: Revista crítica de humanidades, Salvador, n. 246, p. 196-226, 2019.

[10] SANTOS, Milton. O centro da cidade de Salvador: estudo de geografia urbana. Salvador: Livraria Progresso; Universidade da Bahia, 1959.

[11] CARVALHO, Inaiá Maria Moreira de; PEREIRA, Gilberto Corso. As “cidades” de Salvador. In: CARVALHO, Inaiá Maria Moreira de; PEREIRA, Gilberto Corso (org.). Como anda Salvador e sua região metropolitana. 2. ed. Salvador: EDUFBA, 2008.

[12] REIS, Rosana Santana do. Acessibilidade a edifícios históricos de interesse turístico por pessoas com mobilidade reduzida: um estudo de exemplos representativos situados na rota acessível do centro histórico de Salvador. 2015. Dissertação (Mestrado em Arquitetura e Urbanismo) – Universidade Federal da Bahia, Salvador, 2015.

[13] MANTOAN, Maria Teresa Eglér. Inclusão escolar: o que é? por quê? como fazer? São Paulo: Moderna, 2005.

[14] KITCHIN, Rob. Out of place, 'knowing one's place': space, power and the exclusion of disabled people. Disability & Society, v. 13, n. 3, p. 343-356, 1998.

[15] SASSAKI, Romeu Kazumi. Inclusão: construindo uma sociedade para todos. 7. ed. Rio de Janeiro: WVA, 2006. 198 p.

[16] IPEA. Políticas de Acessibilidade no Brasil: diagnóstico e desafios. Nota Técnica n° 45. Brasília: IPEA, 2022. p. 15.

[17] BRASIL. Tribunal de Contas da União. Relatório de Auditoria Operacional: Acessibilidade em obras públicas. Brasília: TCU, 2021. p. 27. (Relatório nº 025.774/2021-0)

[18] LEITE, Flavia Almeida Piva. Acessibilidade e direito à cidade: fundamentos urbanísticos para uma sociedade inclusiva. São Paulo: Editora Revista dos Tribunais, 2018.

[19] Pereira, R. H. M., Braga, C. K. V., Serra, Bernardo, & Nadalin, V. (2019). Desigualdades socioespaciais de acesso a oportunidades nas cidades brasileiras, 2019. *Texto para Discussão Ipea, 2535*. Instituto de Pesquisa Econômica Aplicada (Ipea).

[20] GOMES, M. F. C. Cidades inclusivas: planejamento urbano e acessibilidade. 2. ed. São Paulo: Annablume, 2018.

[21] FERREIRA, M. A. G. et al. Governança urbana e acessibilidade: desafios para cidades inclusivas. Salvador: Editora UFBA, 2020.

[22] GOODCHILD, M. F. Citizens as sensors: the world of volunteered geography. GeoJournal, v. 69, n. 4, p. 211-221, 2007. Disponível em: https://doi.org/10.1111/j.1467-9736.2007.00476.x. Acesso em: 01 de junho 2025.

[23] ORGANIZAÇÃO DAS NAÇÕES UNIDAS (ONU). Transformando nosso mundo: a Agenda 2030 para o desenvolvimento sustentável. 2015. Disponível em: https://brasil.un.org/pt-br/sdgs. Acesso em: 01 de junho 2025.

[24] GOSLING, James; JOY, Bill; STEELE, Guy; BRACHA, Gilad; BUCKLEY, Alex. The Java Language Specification: Java SE 8 Edition. Boston: Addison-Wesley, 2015.

[25] BURKE, B. Android Studio 4.1 Development Essentials – Kotlin Edition. Payload Publishing, 2020.

[26] DARGAN, J. Kotlin for Android App Development: Build Better, Safer, and More Performant Apps. Addison-Wesley, 2019.

[27] FOWLER, M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2021.

[28] MEIER, R. Professional Android 4 Application Development. Wrox Press, 2022.

[29] DOUCET, A. Mobile App Development with Firebase: Build Scalable and Real-time Apps. O'Reilly Media, 2020.

[30] TALBOT, D. Cloud Storage for Developers: Architecting Secure and Scalable Apps. Apress, 2021.

[31] LDN ACCESS. Plataforma colaborativa de acessibilidade urbana. Londres, 2021. Disponível em: https://ldnaccess.com. Acesso em: 02 de junho 2025.

[32] WHEELMAP. Mapa colaborativo de acessibilidade para cadeirantes. Alemanha, 2020. Disponível em: https://wheelmap.org. Acesso em: 02 de junho 2025.

[33] MOOVIT. Aplicativo de mobilidade urbana e acessibilidade. Israel, 2019. Disponível em: https://moovit.com. Acesso em: 03 de junho 2025.

[34] ACCESSNOW. Plataforma global de mapeamento de acessibilidade. Canadá, 2021. Disponível em: https://accessnow.com. Acesso em: 04 de junho 2025.