# LIFE +ATENDIMENTO: APLICATIVO MÓVEL PARA UMA EMPRESA DE SAÚDE REALIZAR O ACOMPANHAMENTO E PREENCHIMENTO DE QUESTIONÁRIOS DE SEUS CLIENTES

**Estevão Dal Vesco Machado[1], Dr. Rafael Vieira Coelho[2]**

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas Instituto Federal de Educação Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Campus Farroupilha

**Resumo.** Este trabalho descreve o desenvolvimento de uma aplicação *mobile* para uma empresa de saúde. Tem como objetivo resolver a dificuldade observada pela empresa no preenchimento de formulários sobre a Segurança do Trabalho ao visitar clientes. Esse processo demanda força braçal para passar os dados do computador para o papel. Como solução, foi desenvolvida uma aplicação *mobile.* Através de uma *API,* ela busca os dados do cliente a ser visitado e gera o questionário para ser preenchido, mesmo sem acesso à Internet. Foi utilizado no seu desenvolvimento o Android Studio como *IDE,* a linguagem de programação JAVA, o banco de dados Room e o conceito de métodos de requisição para acessar a *API*. Por ser *mobile*, a premissa é que o aplicativo esteja sempre disponível para o usuário, proporcionando comodidade e praticidade na hora do preenchimento dos dados do cliente.

***Abstract.*** *This final paper presents the development of a mobile application for a healthcare company. Its objective is to solve the difficulty observed by the company in filling out forms on Occupational Safety during the visit to its customers. This process demands manpower to transfer the data from the computer to paper. As a solution, a mobile application was developed. Through an API, it searches for the data of the customer to be visited and generates a questionnaire to be filled in, even without internet access. During its development, Android Studio was used as the IDE, the programming language chosen was JAVA, the database used was Room and the concept of request methods to access the API were used. As it is mobile, the premise is that the application is always available to the user, providing convenience and practicality when filling in customer data.*

[1] estevao_mac@hotmail.com
[2] rafael.coelho@farroupilha.ifrs.edu.br

## 1. Introdução

Considerando o avanço tecnológico que tem acontecido atualmente, cada vez mais áreas necessitam (ou buscam aprimorar) a tecnologia presente nas suas tarefas. Atividades rotineiras são facilitadas pelo surgimento de ferramentas e aplicativos, os quais são utilizados diariamente e podem acelerar a realização de uma tarefa. Além disso, o uso de dispositivos móveis para acesso à Internet é superior ao uso de computadores, como mostra na Pesquisa sobre o Uso das Tecnologias de Informação e Comunicação nos Domicílios Brasileiros – TIC Domicílios 2020 (CGI.br., 2021), onde 99% dos brasileiros acessam pelo celular, em contrapartida aos 42% que acessam pelo computador.

Em um dispositivo móvel, as aplicações que utilizamos denominam-se *mobile apps* (aplicações móveis). As vantagens que estes aplicativos oferecem, quando comparados a sistemas *web*, são diversas: facilidade de uso, menor custo de acesso, melhor uso dos recursos disponíveis e acesso *offline* (COSTA, 2018, apud Porto, 2012). Devido a essas vantagens, cada vez mais aplicativos são desenvolvidos para atender necessidades de diversas áreas.

Uma empresa de saúde, visando digitalizar seu processo, solicitou o desenvolvimento de um aplicativo *mobile* para preencher questionários e realizar anotações na visita aos seus clientes. Anterior à existência do aplicativo, o processo era demorado e demandava uma força braçal, pois era necessário verificar os dados do cliente na plataforma *desktop*, passar essas anotações para o papel, preparar o questionário (que varia de cliente para cliente) e depois repassar as informações adquiridas de volta para a plataforma.

Tendo em vista a natureza de uma aplicação *mobile*, é necessário levar em conta o *design* da aplicação, devido ao tamanho de dispositivos móveis e de suas funções *touch screen*. Esses fatores são essenciais para obter uma melhor *User Experience* (experiência do usuário).

> A interface do usuário é uma tela gráfica sensível ao toque em um dispositivo móvel. É a primeira coisa que o usuário vê e dá a impressão da capacidade do aplicativo. [...] a interface do usuário deve ser projetada de forma interativa, eficaz, intuitiva e amigável. (YAZID, M. A.; JANTAN, A. H., 2017, p. 199)

A empresa em questão possui todo o seu processo alinhado na aplicação *desktop.* Para realizar a comunicação entre *desktop* e *mobile*, a alternativa escolhida foi desenvolver uma *API*. Uma *API*, sigla para *Application Programming Interface* (Interface de Programação de Aplicações), fornece uma abstração para um problema e especifica como os clientes devem interagir com os componentes de software que implementam uma solução para esse problema (REDDY, Martin, 2011, p.1). Através do seu uso, é possível realizar a comunicação e troca de dados entre duas fontes diferentes através de uma conexão de rede. Com isso, podemos fazer com que aplicações “conversem” entre si, habilitando um leque de possibilidades.

Considerando o projeto, foi definido como objetivo geral: desenvolver um aplicativo *mobile* para que os funcionários de uma empresa de saúde possam receber, preencher e enviar os formulários de visita. Sendo a visita uma das primeiras etapas do processo (onde é preenchido o questionário sobre Segurança do Trabalho dos Estabelecimentos), a experiência que o uso de um aplicativo proporciona facilita o trabalho do encarregado. Tendo em conta uma ampla gama de clientes, com possíveis visitas diferentes sendo realizadas no mesmo dia, é imprescindível que seja algo simples e rápido. Por isso, foram definidos os seguintes objetivos específicos: facilitar e acelerar a etapa da visita ao cliente no processo geral; proporcionar um aplicativo amigável; proporcionar uma imersão tecnológica à empresa.

Este artigo apresentará a definição e desenvolvimento de um aplicativo móvel, denominado LIFE +Atendimento, para uma empresa de saúde realizar o acompanhamento e preenchimento de questionários de seus clientes. A Seção 2 contém o referencial teórico, apresentando a empresa e o processo já existente; a Seção 2 apresenta a metodologia, contemplando as ferramentas e tecnologias utilizadas e o planejamento; a Seção 3 trata dos resultados, apresentando o aplicativo; e por fim, a Seção 4 aborda as considerações finais.

## 2. Referencial Teórico

O projeto apresentado neste artigo enquadra-se na área da Segurança do Trabalho. É um conjunto de normas e procedimentos legalmente exigidos às

empresas e funcionários visando prevenir doenças ocupacionais, acidentes de trabalho e proteger a integridade física do trabalhador (Portal da Indústria, 2021). Durante a visita do Gestor da Segurança do Trabalho, ele coleta as informações pertinentes do cliente através do questionário para serem trabalhadas posteriormente. As informações coletadas variam entre os clientes visitados, pois dependendo a área de atuação são necessários dados diferentes, como por exemplo a existência de um Programa de Proteção Respiratória caso exista exposição a riscos químicos e biológicos.

A empresa utiliza a aplicação *desktop* OTMSuite. A OTMSuite é uma plataforma de execução de aplicações que permite a geração de aplicativos de forma padronizada, simples e dinâmica (Otimiza, 2021). Através desta aplicação, a empresa tem todo o seu *workflow* definido e atrelado a geração de relatórios e criação de cadastros. Uma das opções disponíveis é a criação do questionário a ser apresentado pelo aplicativo, assim como a visualização das respostas.


| Seq | Quesito | Resposta |
|---|---|---|
| 5 | DADOS DA PESSOA ENTREVISTADA | 1 |
| 10 | A EMPRESA POSSUI CIPA CONSTITUÍDA? | SIM |
| 15 | SE A EMPRESA POSSUI CIPA, QUAL O DIMENSIONAMENTO? | 18 |
| 15 | SE A EMPRESA NÃO CONSTITUI CIPA, POSSUI UM DESIGNADO TREINADO? | NÃO |
| 20 | A EMPRESA POSSUI UM TÉCNICO DE SEGURANÇA DO TRABALHO (TST)? | SIM |
| 30 | O TÉCNICO DE SEGURANÇA DO TRABALHO É CLT? | SIM |
| 35 | SE O TÉCNICO DE SEGURANÇA DO TRABALHO NÃO É CLT, COLETAR DADOS EMPRESA TERCEIRIZADA: | |
| 35 | SE O TÉCNICO DE SEGURANÇA DO TRABALHO É CLT, COLETAR DADOS DA PESSOA: | PEDRO PAULO CLT 111 CONTATO 9999999 |
| 40 | A EMPRESA POSSUI O PROGRAMA DE CONSERVAÇÃO AUDITIVA (PCA) IMPLEMENTADO? | SIM |
| 50 | A EMPRESA POSSUI O PROGRAMA DE PROTEÇÃO RESPIRATÓRIA (PPR) IMPLEMENTADO? | NÃO |
| 60 | A EMPRESA POSSUI AVALIAÇÕES ERGONÔMICAS DO TRABALHO? | SIM |
| 70 | A EMPRESA POSSUI O PROGRAMA DE PREVENÇÃO CONTRA INCÊNDIO (PPCI)? | NÃO |
| 80 | A EMPRESA POSSUI SISTEMA DE ADEQUAÇÃO DE MÁQUINAS CONFORME A NR12? | SIM |
| 90 | ANOTAÇÕES | |
| 100 | DATA DA VISITA | 16/09/2020 |

**Figura 1: Visualização das respostas na OTMSuite**
**Fonte: Produção Própria**

## 3. Metodologia

### 3.1. Ferramentas e tecnologias

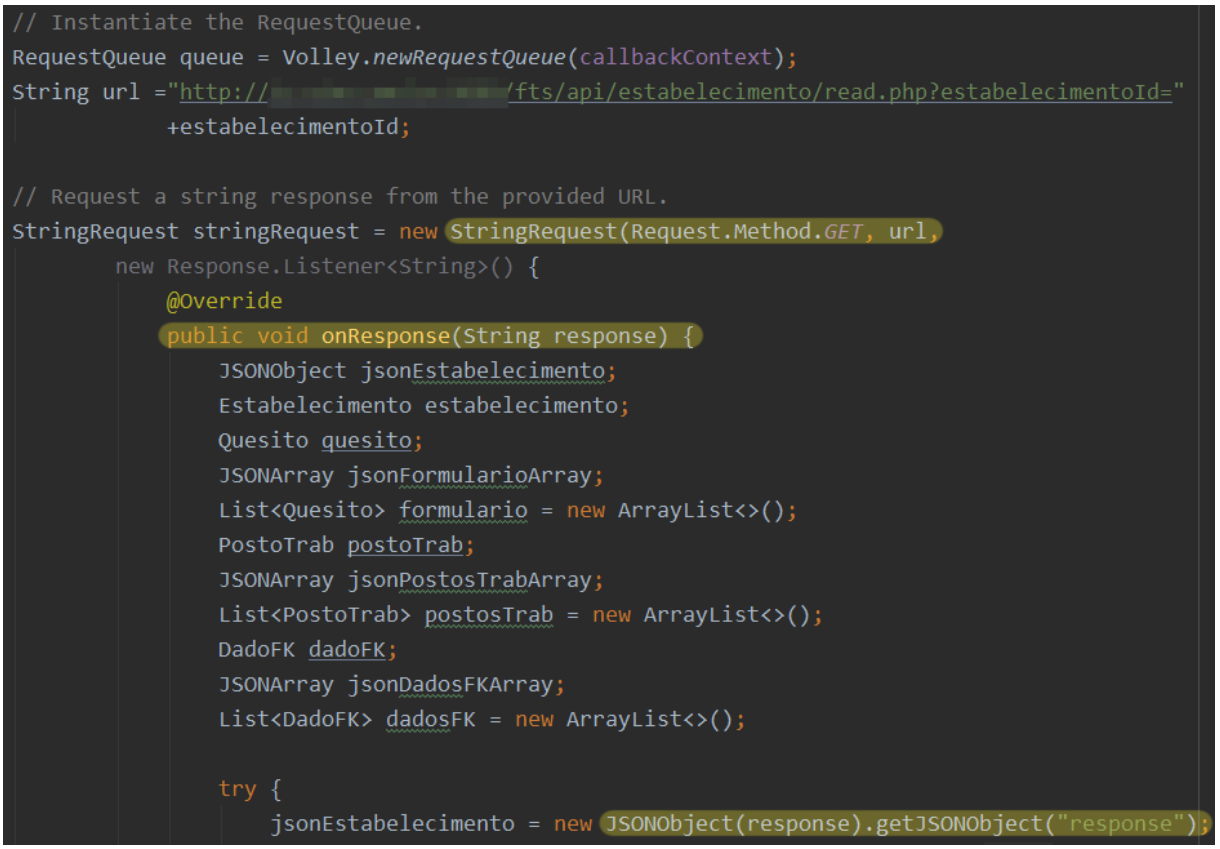
O IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado) utilizado foi o Android Studio. Desenvolvido pela Google, baseado IntelliJ IDEA, é o ambiente oficial para desenvolvimento de apps Android (Android Studio, 2021). Além de ser gratuito, a equipe já possuía conhecimento prévio de cursos utilizando esse ambiente.

Para programar no Android Studio, a linguagem utilizada pode ser Java ou Kotlin, sendo Java a escolhida para este projeto. Assim como o IDE, a linguagem também é gratuita, e é a plataforma de desenvolvimento moderna mais conhecida do mundo (Oracle, 2021). Por ser programação orientada a objetos (POO), consegue-se manter uma maior organização no código, fornecendo uma manutenção fácil e rápida de ser feita. Outro ponto bastante utilizado nesse projeto foi o conceito da reutilização de métodos, removendo a repetição de código desnecessário.

No quesito banco de dados, foi utilizado o Room. O banco de dados Room oferece uma camada de abstração sobre o SQLite para permitir acesso fluente ao banco de dados e, ao mesmo tempo, aproveitar toda a capacidade do SQLite (Android, 2021). Existem três componentes principais no Room: o banco de dados, as entidades (que representam as tabelas) e os objetos de acesso a dados (DAOs, na sigla em inglês). Ao utilizar o Room, é possível economizar tempo e esforço necessário para realizar as consultas, pois deixamos a biblioteca cuidar dos detalhes.

Para a comunicação entre o aplicativo e a aplicação *desktop*, foi desenvolvida uma *API* em PHP. O PHP (um acrônimo recursivo para PHP: Hypertext Preprocessor) é uma linguagem de script open source de uso geral que executa no lado do servidor (server-side) (PHP, 2021). O PHP também tem suporte para comunicação com outros serviços utilizando protocolos, e o escolhido para o nosso caso foi o HTTP.

O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso (Mozilla, 2021). Com a nossa *API* rodando no servidor da empresa, é possível realizar essas requisições para transferir dados entre as aplicações. Os dois métodos de requisição utilizados foram: *GET* e *POST*. O método *GET* solicita a representação de um recurso específico. Requisições utilizando o método *GET* devem retornar apenas dados (Mozilla, 2021). Já o método *POST* é usado para enviar objetos, causando uma alteração no mesmo.



```
// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(callbackContext);
String url ="http://[redacted]/fts/api/estabelecimento/read.php?estabelecimentoId="
            +estabelecimentoId;

// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                JSONObject jsonEstabelecimento;
                Estabelecimento estabelecimento;
                Quesito quesito;
                JSONArray jsonFormularioArray;
                List<Quesito> formulario = new ArrayList<>();
                PostoTrab postoTrab;
                JSONArray jsonPostosTrabArray;
                List<PostoTrab> postosTrab = new ArrayList<>();
                DadoFK dadoFK;
                JSONArray jsonDadosFKArray;
                List<DadoFK> dadosFK = new ArrayList<>();

                try {
                    jsonEstabelecimento = new JSONObject(response).getJSONObject("response");
```

**Figura 2: Chamada da *API* com método *GET***
**Fonte: Produção Própria**

A Figura 1 apresenta uma parte do código contendo uma chamada de *API* utilizando o método de requisição *GET*. Para realizar essa requisição de forma fácil, foi utilizada a biblioteca Volley. Através do identificador do estabelecimento, podemos realizar uma comunicação com a *API* para a mesma retornar os dados do estabelecimento em questão. Dentro do método *onResponse*, podemos trabalhar com o objeto recebido na requisição (no nosso caso, um JSON).

O JSON (J*avaScript Object Notation* - Notação de Objetos JavaScript) é uma formatação leve de troca de dados (JSON, 2021). Com uma estrutura simplificada, o JSON é fácil de ser entendido por nós e simples de ser interpretado pelas máquinas. O JSON utiliza uma estrutura familiar para as linguagens de programação, e, por isso, torna-se ideal para a troca de dados. O JSON é composto por duas estruturas: uma coleção de pares nome/valor e uma lista ordenada de valores.

Estas estruturas abrangem dados universais, permitindo que possamos transferir tipos diferentes de dados na mesma estrutura e entre diferentes linguagens. No nosso caso, estamos utilizando o JSON para enviar e receber os dados dos estabelecimentos, pares de nome/valor de acordo com os valores definidos no banco de dados.

### 3.2. Planejamento

Como primeira etapa do planejamento, foi definido o modelo cascata para o desenvolvimento do projeto. Esse modelo considera as atividades fundamentais do processo de especificação, desenvolvimento, validação e evolução, e as representa como fases distintas (SOMMERVILLE, 2011, p. 19). Uma alteração realizada foi adicionar uma realimentação ao modelo, para podermos obter um *feedback* entre uma fase e outra, nos permitindo desenvolver exatamente o que foi solicitado. Abaixo, na Figura 2, é possível ver a representação do modelo.

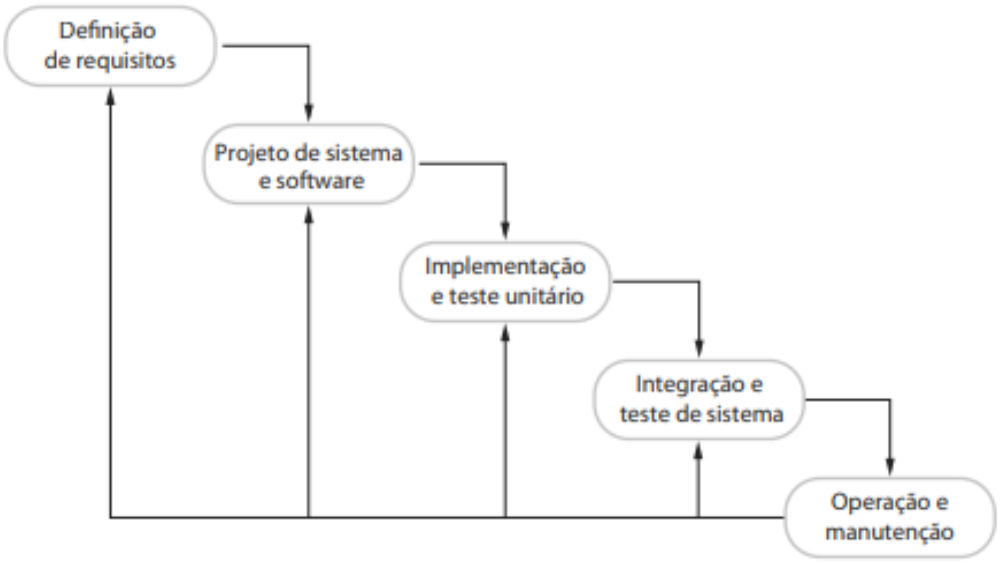



**Figura 3: Modelo em Cascata**
**Fonte: Sommerville (2011, p. 20)**

Durante reuniões com o consultor responsável pelo projeto na empresa, foram realizados o levantamento de requisitos e a definição do escopo do aplicativo, atividades fundamentais ao desenvolvimento do aplicativo. Os requisitos foram separados entre requisitos funcionais, não funcionais e de sistema.

De acordo com Sommerville (2011), os requisitos funcionais definem o que o sistema fornece, como ele reage a entradas e como se comporta em determinadas situações. Os não funcionais apresentam restrições a serviços ou funções do sistema e os requisitos de sistema referem-se a funções, serviços e restrições operacionais do sistema de *software*.

A seguir, nas Quadros 1, 2 e 3, são apresentadas os quadros referentes ao levantamento de requisitos funcionais, não funcionais e de sistema:

**Quadro 1: Requisitos Funcionais**

| Código | Nome |
|---|---|
| RF1 | Permitir download de estabelecimento com questionário e postos de trabalho |
| RF2 | Permitir preenchimento do questionário |
| RF3 | Permitir preenchimento de observações dos postos de trabalho |
| RF4 | Permitir upload de estabelecimento com questionário e postos de trabalho |

**Fonte: Produção Própria**

**Quadro 2: Requisitos não funcionais**

| Código | Nome |
|---|---|
| RNF1 | Disponibilidade |
| RNF2 | Segurança |
| RNF3 | Portabilidade |

**Fonte: Produção Própria**

**Quadro 3: Requisitos de sistema**

| Código | Nome |
|---|---|
| RS1 | O sistema deve ter interface de login para validar o usuário |
| RS2 | O usuário só poderá interagir com estabelecimentos dos quais ele tem permissão de acesso |
| RS3 | O sistema deve armazenar no banco de dados os dados dos estabelecimentos com questionário e postos de trabalho |
| RS4 | Não será possível salvar alterações no questionário caso as questões obrigatórias estiverem nulas |
| RS5 | O preenchimento do questionário e das observações deve ser permitido mesmo *offline* |

**Fonte: Produção Própria**

De acordo com os requisitos, de forma geral o aplicativo oferece uma forma prática de preencher um formulário. Porém, uma característica determinante para o seu uso é o RS5, apresentado no Quadro 3: Requisitos de sistema. A necessidade de uso do aplicativo mesmo sem conexão com a internet foi o marco para a decisão de desenvolver um aplicativo *mobile.*

Como escopo do sistema, temos: Após o login obrigatório, o usuário carrega o estabelecimento. Tanto o *download* como o *upload* são feitos por chamadas de *api*, que buscam os dados na tabela da empresa. O sistema mantém as informações em um banco de dados diretamente no dispositivo, permitindo o acesso *offline*. O usuário pode selecionar um estabelecimento dentre os carregados, preencher o questionário e editar as observações dos postos de trabalho. Após o preenchimento, o usuário salva as alterações realizadas localmente e pode realizar o *upload* das mesmas.

Abaixo, na Figura 3, é apresentado o Diagrama de Casos de Uso para visualização do funcionamento do aplicativo dentro dos requisitos definidos.

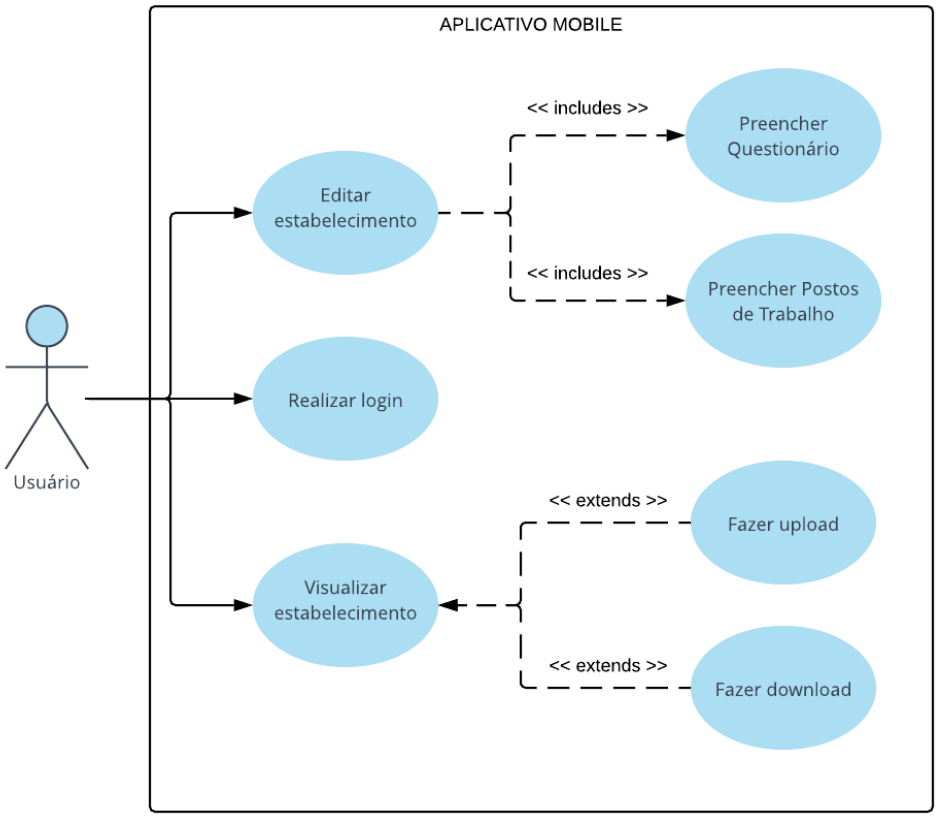



**Figura 4: Diagrama de Casos de Uso do aplicativo *mobile***
**Fonte: Produção Própria**

Como o aplicativo tem interação direta com o usuário, foi identificado apenas um ator. A seguir é apresentada uma breve descrição dos casos de uso:

- **Usuário:** Representa o funcionário da empresa de saúde responsável por realizar a visita e entrevista no cliente;
- **Realizar *login:*** O usuário pode realizar o login de acordo com a sua conta cadastrada no sistema *desktop* para acessar os estabelecimentos atrelados a sua conta;
- **Visualizar estabelecimento:** O usuário pode visualizar os estabelecimentos dos quais tem acesso;
- **Fazer *upload:*** Após o preenchimento correto de um estabelecimento, o usuário pode realizar o upload do mesmo;
- **Fazer *download:*** O usuário pode realizar o download dos estabelecimentos dos quais ele tem acesso;

- **Editar estabelecimento:** Após realizar o download de um estabelecimento, o usuário pode acessar a área de edição;
- **Preencher Questionário:** Ao editar o estabelecimento, o usuário pode preencher as questões definidas para o estabelecimento em questão;
- **Preencher Postos de Trabalho:** Além de editar o Questionário, o usuário também pode adicionar anotações dos Postos de Trabalho do estabelecimento.

Após definir os requisitos e os casos de uso, foi necessário pensar na estrutura do banco de dados a ser utilizada, já que o aplicativo precisa armazenar as informações relevantes dos clientes para trabalhar com elas. Utilizando a base de dados já existente dos clientes e considerando a necessidade dos mesmos no aplicativo, foi elaborado o modelo entidade e relacionamento (ER) apresentado abaixo (Figura 4).

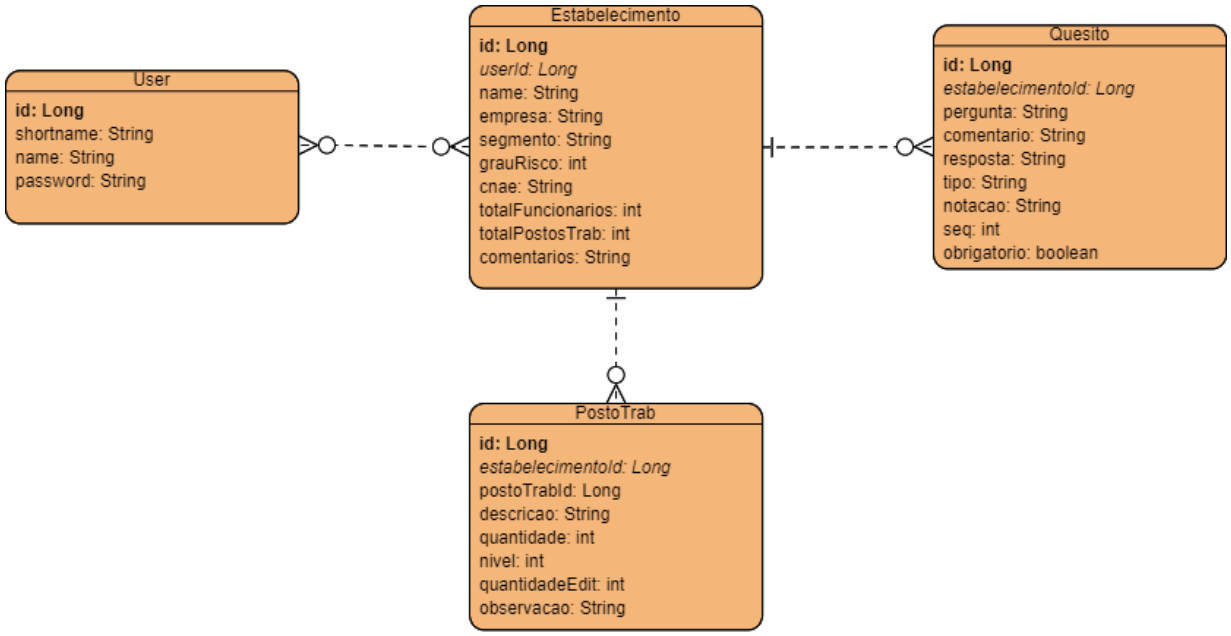



**Figura 5: Entidade e Relacionamento**
**Fonte: Produção Própria**

A tabela *User* contém os usuários cadastrados, para validar o *login* e os dados os quais ele tem acesso. A tabela Estabelecimento armazena as informações

referentes aos estabelecimentos dos clientes a serem visitados. A tabela PostoTrab possui os postos de trabalho referentes ao estabelecimento, que podem ter subníveis. A tabela Quesito contempla as questões a serem apresentadas no formulário. Dentre suas colunas, a notação contém as informações de como o campo será estruturado, podendo ser textual, numérico, *checkbox*, data ou múltipla escolha (em forma de tabela ou *dropdown*).

## 4. Resultados

Ao iniciar o aplicativo, a primeira tela apresentada será a de *login*. Na Figura 5 é apresentada a tela de *login* com o aplicativo sendo executado através de um emulador. Se houver conexão com a internet, a autenticação será realizada por uma *API*, onde é validada a senha e o usuário de acordo com o cadastro feito pelo aplicativo *desktop*. Em caso de usuário inválido, apresenta mensagem. Em caso de sucesso, o usuário é armazenado para possíveis *logins offline* e a tela inicial é apresentada.

Ao realizar *login offline*, a autenticação será realizada de acordo com os usuários armazenados localmente. Em caso de usuário inválido, apresenta mensagem e um aviso para realizar o primeiro *login* com conexão à internet para o usuário poder ser armazenado. Em caso de sucesso, apresenta a tela inicial.


**Figura 6: Tela de login do aplicativo**
**Fonte: Produção Própria**

Na Figura 6 pode-se observar a tela inicial do aplicativo. Na parte superior temos o nome temporário do aplicativo e um botão que abre a opção de *logout*. Logo abaixo é possível ver o logo da empresa junto do nome do usuário conectado. Na parte central temos as três opções que o usuário pode realizar: carga de estabelecimento, edição de estabelecimento e envio de estabelecimento. Cada uma dessas opções leva o usuário para uma tela diferente.


**Figura 7: Tela inicial**
**Fonte: Produção Própria**

Ao clicar na opção “Carga”, o usuário é redirecionado para a tela de “Carregar Estabelecimentos”, apresentada na Figura 7. Agora, além do botão no canto direito superior apresentar a opção de *logout*, temos também a opção de retornar para a tela inicial. Os estabelecimentos apresentados para a carga são apenas os que o usuário tem acesso (definido na aplicação *desktop*) e que não estejam carregados em outro dispositivo, evitando dois usuários de editarem o mesmo. Ao selecionar um dos estabelecimentos, uma mensagem de confirmação é apresentada, solicitando se o usuário realmente deseja realizar a carga do mesmo.


**Figura 8: Tela de Carregar Estabelecimento**
**Fonte: Produção Própria**

Ao clicar na opção “Edição”, o usuário é redirecionado para a tela de “Editar Estabelecimento”. Os estabelecimentos apresentados são apenas os que o usuário já realizou a carga e ainda não realizou o envio. Ao selecionar o estabelecimento para editar, o usuário é levado para a tela contendo os detalhes do estabelecimento, onde pode-se observar algumas informações relevantes recebidas pela *API* e contendo acima uma barra de navegação para trocar entre as abas “Informações”, “Postos de Trabalho” e “Formulário”. Ambas as telas podem ser observadas na figura 8.

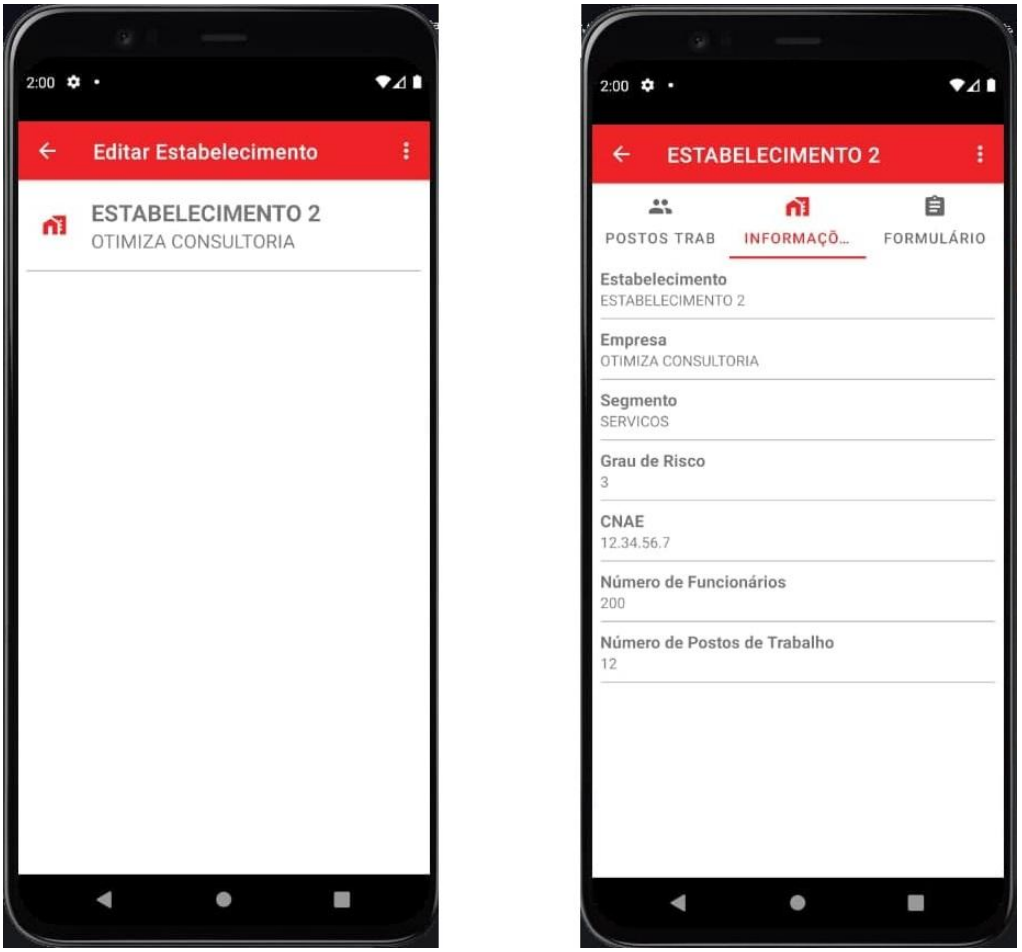



**Figura 9: Tela de Editar Estabelecimento e Estabelecimento 2 em Edição**
**Fonte: Produção Própria**

Na aba “Postos de Trabalho”, é possível observar os postos de trabalho de primeiro nível daquele estabelecimento, assim como um número representando o total de funcionários dentro daquele posto. Ao realizar um clique no posto de trabalho, são carregados subníveis do mesmo, dando mais detalhes para aquele posto. Por exemplo, no posto de trabalho “Manutenção”, podemos ter como subníveis “Manutenção Preventiva” e “Manutenção Corretiva”. Note também que existem dois campos para adicionar comentários, referentes ao nome do posto de trabalho e à quantidade de funcionários (Figura 9).


**Figura 10: Postos de Trabalho**
**Fonte: Produção Própria**

Na aba “Formulário”, são apresentadas as questões definidas na plataforma *desktop* e recebidas pela *API* (Figura 10). O aplicativo pega as informações recebidas e transforma em perguntas para apresentar ao usuários, que podem ser de seis tipos: textual, numérico, data, *dropdown*, *checkbox* e lista de valores (tabela para seleção).

Todas as perguntas possuem a mesma estrutura, um texto de cabeçalho e o campo para resposta. Além disso, é possível que elas contenham um comentário de detalhamento, que pode ser visualizado ao clicar no ícone “i” em vermelho. Caso a pergunta seja obrigatória e não esteja respondida, o campo fica marcado como inválido.


**Figura 11: Formulário**
**Fonte: Produção Própria**

Voltando para a tela inicial, ao clicar na opção “Envio”, o usuário é redirecionado para a tela de “Enviar Estabelecimento”, apresentada na Figura 11. Assim como a tela de “Editar Estabelecimento”, somente são apresentados os estabelecimentos carregados no dispositivo para o usuário atual. Ao clicar em um estabelecimento, é apresentada uma mensagem de confirmação solicitando se o usuário realmente deseja realizar o envio do estabelecimento. Caso o formulário contenha alguma pergunta obrigatória em branco, não é possível realizar o envio.

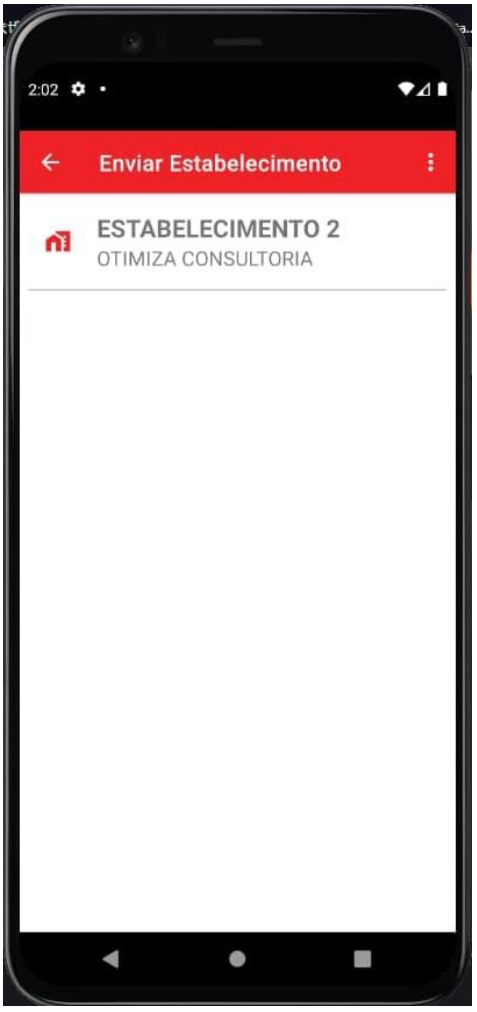



**Figura 12: Tela de Enviar Estabelecimento**
**Fonte: Produção Própria**

Tanto a carga quanto o envio de estabelecimento apresentam uma mensagem de sucesso ao finalizar. A qualquer momento o usuário pode encerrar a aplicação e ao realizar o *login* novamente terá acesso aos mesmos estabelecimentos e com as informações preenchidas como ele deixou. Mais de um usuário pode utilizar o mesmo dispositivo, porém cada um terá acesso aos seus estabelecimentos permitidos.

Buscando soluções no mercado, pode-se encontrar ferramentas já prontas para a criação e preenchimento de formulários. Dentre elas temos como as mais famosas a Google Forms e a Microsoft Forms. Embora ambas atendam ao quesito de criação de formulários perfeitamente, elas deixam a desejar na questão de integração. A Google Forms iniciou testes *beta* de sua *API* no final de 2021 (Google, 2021), além de terem certa burocracia e custo de uso. Para esse projeto, era

necessário uma integração direta com a aplicação *desktop*, sendo possível receber os dados preenchidos diretamente no banco de dados da empresa.

Além disso, ambas têm a limitação de funcionamento *mobile offline*, outra questão essencial solicitada na definição dos requisitos. Muitos clientes visitados acabam não contando com conexão à internet, e é necessário que o aplicativo continue disponível mesmo nesses casos.

## 5. Considerações finais

Este artigo teve como objetivo apresentar as definições e o desenvolvimento de uma aplicação *mobile* solicitada por uma empresa de saúde para preencher questionários durante as visitas aos clientes. O sistema acaba por facilitar e acelerar o processo realizado pela empresa, por poder substituir papéis e canetas e automatizar o processo de troca de informações. Além disso, acaba proporcionalizando uma imersão tecnológica para a empresa.

No texto foram mostradas as tecnologias utilizadas, assim como suas definições e as telas da aplicação, com detalhamentos referentes às opções de cada tela apresentada.

O desenvolvimento foi todo realizado no Android Studio. Uma *API* desenvolvida paralelamente ao projeto do aplicativo foi utilizada para realizar a validação do *login* e a solicitação e envio de dados entre o aplicativo e a plataforma *desktop*. Para os dados locais, utilizou-se como forma de armazenamento o banco de dados Room.

O aplicativo contemplou todas as expectativas solicitadas pela empresa e logo após a conclusão do desenvolvimento, foi utilizado pela mesma. Durante a utilização surgiram novas solicitações, como pequenos ajustes na parte visual dos campos editáveis e um campo novo para adicionar comentários referentes ao estabelecimento. Os ajustes foram prontamente realizados e estão em fase de validação.

Como trabalho futuro, espera-se a disponibilidade de um servidor com protocolo HTTPS, que irá proporcionar acesso externo para a *API* e uma maior segurança. Contudo, será necessário realizar alterações no aplicativo em cada chamada de *API*. Além disso, busca-se aprimorar a interface para deixá-la não apenas mais bonita como também mais intuitiva, pois alguns usuários sentiram dificuldades na utilização da aplicação.

## REFERÊNCIAS


ANDROID. **Conheça o Android Studio.** Disponível em: <https://developer.android.com/studio/intro?hl=pt-br>. Acesso em: 10 de setembro de 2021.

ANDROID. **Salvar dados em um banco de dados local usando o Room.** Disponível em: <https://developer.android.com/training/data-storage/room#java>. Acesso em: 10 de setembro de 2021.

CETIC. **TIC DOMICÍLIOS 2020**. 18 de agosto de 2021. Disponível em: <https://cetic.br/media/analises/tic_domicilios_2020_coletiva_imprensa.pdf> Acesso em: 07 de dezembro de 2021.

COSTA, Mirlanda Sousa. **Sistemas web e mobile: uma visão geral para negócios empresariais.** Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano 03, Ed. 08, Vol. 09, pp. 82-99, Agosto de 2018. ISSN:2448-0959

GOOGLE. **Announcing the Google Forms API,** 12 de outubro de 2021. Disponível em: <https://developers.googleblog.com/2021/10/announcing-google-forms-api_01768004272.html>. Acesso em: 10 de setembro de 2021.

GUAY, Matthew. **The 8 best online form builder apps in 2021**, 4 de outubro de 2021. Disponível em: <https://zapier.com/learn/forms-surveys/best-online-form-builder-software/> Acesso em: 10 de setembro de 2021.

MOZILLA. **Métodos de requisição HTTP.** Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>. Acesso em: 10 de setembro de 2021.

OTIMIZA. **Tecnologia.** Disponível em: <https://www.otm.com.br/tecnologia/>. Acesso em: 5 de janeiro de 2022.

PHP. **O que é o PHP?** Disponível em: <https://www.php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 10 de setembro de 2021.

PORTAL DA INDÚSTRIA. **Segurança e Saúde no Trabalho.** Disponível em: <https://www.portaldaindustria.com.br/industria-de-a-z/seguranca-saude-trabalho/>. Acesso em: 5 de janeiro de 2022.

REDDY, Martin. **API Design for C++**. Estados Unidos, Elsevier Science, 2011.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª Ed. São Paulo: Pearson Prentice Hall, 2011.

YAZID, M. A.; JANTAN, A. H. **User Experience Design (UXD) of Mobile Application: An Implementation of a Case Study**. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), [S. l.], v. 9, n. 3-3, p. 197–200, 2017.

ZIMMER, Ben. BARRETT, Grant. METCALF, Allan. **“App” 2010 Word of the Year, as voted by American Dialect.** American Dialect Society, 7 de janeiro de 2011.