

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Edson Júnio Sousa Sobrinho

**APO Digital: Evolução da comunicação entre
os componentes**

Uberlândia, Brasil

2019

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Edson Júnio Sousa Sobrinho

**APO Digital: Evolução da comunicação entre os
componentes**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Sistemas de Informação.

Orientador: Luiz Claudio Theodoro

Coorientador: Maria Adriana Vidigal Lima

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2019

Agradecimentos

Primeiramente agradeço a Deus, por ter me concedido saúde, força e os dons que me serviram na realização deste projeto.

Agradeço aos Professores MsC. Luiz Claudio Theodoro e Dra. Maria Adriana Vidigal Lima por aceitarem conduzir o meu trabalho de pesquisa e desenvolvimento. Sou grato pela confiança depositada na minha proposta, pela paciência, motivação e orientação durante todo o trabalho

Agradeço a minha família por sempre me incentivar e acreditar que eu seria capaz de superar os obstáculos que a vida me apresentou.

Agradeço a todos os meus professores do curso de Sistemas de Informação da Universidade Federal de Uberlândia pela excelência e qualidade técnica de cada um.

Também agradeço à todos os meus amigos e colegas de curso, pela oportunidade do convívio e pela cooperação mútua durante estes anos.

“Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista.

Se você quer chegar a onde a maioria não chega, faça o que a maioria não faz.”

Bill Gates

Lista de abreviaturas e siglas

API	Application Programming Interface
APO	Avaliação de pós-ocupação
IPEA	Instituto de Pesquisa Econômica Aplicada
JPA	Java Persistence API
JSON	JavaScript Object Notation
REST	Representational State Transfer
CRUD	Create, Read, Update and Delete
CORS	Cross-origin resource sharing
SSH	Secure Shell

Sumário

1	INTRODUÇÃO	6
1.1	Justificativa e Motivação	7
1.2	Objetivos	8
1.3	Organização da Monografia	9
2	REVISÃO BIBLIOGRÁFICA	10
2.1	Serviço Web	10
2.2	Tecnologias para Serviços web	11
2.3	Banco de dados Não Relacionais	11
2.4	Maven	12
2.5	Intellij IDEA	12
2.6	Java 8	13
3	SISTEMA APO DIGITAL	15
4	SERVIÇO WEB PARA COMUNICAÇÃO ENTRE COMPONENTES DO SISTEMA APO DIGITAL	19
4.1	Aprimoramento do componente APO Digital	19
4.2	Infraestrutura de ambiente em produção	21
4.3	Comunicação com os aplicativos via REST	24
4.4	Estrutura de comunicação entre aplicativos móveis e APO Digital	26
4.5	Padrões de Projeto APO Digital	35
5	CONCLUSÃO	37
	REFERÊNCIAS	38
	ANEXO A – QUESTIONÁRIO COMO VOCÊ MORA?	40

1 Introdução

Este trabalho pretende dar continuidade à construção de software de apoio aos projetos realizados no laboratório do Núcleo [MORA] - **Pesquisa em Habitação**, coordenado pela professora Dra. Simone Barbosa Villa. O [MORA] reúne professores e alunos dos cursos de Arquitetura e Urbanismo, Design, Artes Visuais, Geografia, Ciência da Computação e Sistemas de Informação da Universidade Federal de Uberlândia e tem como missão propor ações que contribuam efetivamente com a melhoria da qualidade de habitação com foco em inovações tecnológicas, sustentabilidade, qualidade espacial e avaliação de pós-ocupação (APO). A qualidade se caracteriza pela satisfação das necessidades dos usuários em relação ao desempenho dos ambientes construídos (ROMERO; ORNSTEIN, 2003). A relevância da APO e a importância do conhecimento de seus resultados nas decisões de projeto e construção em curto, médio e longo prazo têm sido amplamente pesquisadas (BECKER, 1989; COATES; ARAYICI; OZTURK, 2012; COUNCIL; ENVIRONMENT; COUNCIL, 2002; HADJRI; CROZIER, 2009; PREISER; WHITE; RABINOWITZ, 2016). A APO reúne um conjunto de métodos e técnicas para analisar a construção e o uso da edificação considerando primordialmente o ponto de vista do usuário.

Especificamente na área da pós-ocupação, o grupo MORA já realizou diversas coletas de dados em campo, via questionários em papel e em meio digital. Foi construída uma plataforma computacional, denominada **Sistema APO Digital**, composta por: (i) aplicativos para aquisição dos dados em dispositivos móveis e (ii) aplicação web para gerenciamento de questionários e estudo dos dados coletados, além da emissão de relatórios. Este trabalho pretende colaborar no aprimoramento da comunicação de dados entre os componentes do Sistema APO Digital, notadamente na construção de um serviço web para receber os dados coletados. Este serviço deve obter os dados em formato JSON e executar um processo de reorganização dos mesmos para que possam ser armazenados no banco de dados principal.

O Sistema APO Digital encontra-se registrado no Instituto Nacional de Propriedade Intelectual, estando a cotitularidade já firmada entre Universidade Federal de Uberlândia e Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG). A aplicação web para a construção e gerenciamento dos questionários denomina-se *APOWeb* e proporciona um ambiente para a construção de APOs específicas, que podem ser aplicadas em uma região, tipo de moradia (condomínios verticais ou horizontais, conjuntos habitacionais de interesse social, espaços comerciais), e que tem como objetivo avaliar um conjunto restrito de conceitos. A *APOWeb* foi construída para ser utilizada apenas por usuários especialistas: arquitetos e pesquisadores, e encontra-se hospedada no servidor

web do Laboratório de Computação da FACOM/UFU.

1.1 Justificativa e Motivação

A aplicação de uma APO se dá com o uso de aplicativos desenvolvidos para dispositivos móveis. Os aplicativos fornecem o meio gráfico utilizado pelos usuários moradores para responder às diversas questões das respectivas APOs. Os aplicativos foram projetados para funcionar em dispositivos com sistema operacional Android, e desenvolvidos com o auxílio da plataforma Corona SDK (ZAMMETTI, 2013), que é uma tecnologia de criação de jogos digitais multiplataforma em duas dimensões (2D) e a linguagem de programação Lua (IERUSALIMSKY, 2006).

Fundamentando-se no ambiente já construído para o Sistema APO Digital, composto pela *APOWeb* e pelos aplicativos de coleta de dados, este trabalho propõe estudar e desenvolver uma estratégia de comunicação entre os componentes do sistema, de forma a definir um serviço web de coleta, que possa ser oferecido aos diversos aplicativos responsáveis pelas coletas de dados, considerado o uso de novas tecnologias e padrões de desenvolvimento web.

A evolução dos softwares já existentes a partir da incorporação de novas tecnologias que permitam maior facilidade na troca de dados viabiliza uma plataforma computacional mais completa e mais eficiente, que pode oferecer tanto à equipe do [MORA], quanto aos alunos do curso de Arquitetura e Urbanismo, um ambiente de criação e aplicação de APOs específicas, conforme as necessidades e situações que se pretende avaliar. A plataforma do Sistema APO Digital pode apoiar atividades nas disciplinas da área de Conforto Ambiental, Sustentabilidade e Eficiência Energética (ao possibilitar a coleta de informações sobre a qualidade ambiental dos espaços habitados), como também naquelas da área de Projeto Arquitetônico.

Os aplicativos que são utilizados para a coleta de dados podem ser programados em plataformas e linguagens diversas, e novos questionários e modelos de coleta de dados podem ser utilizados pelos pesquisadores, considerando dados textuais. Desta forma, a solução de comunicação entre aplicativos e a *ApoWeb* pretende ser ampla e baseada num padrão único de comunicação. Assim, a solução de comunicação baseou-se na criação de uma nova API com recursos que atenderiam ao esboço atual de estrutura *front-end* e *back-end*.

Após diversas aplicações de questionários em diferentes localidades, identificou-se a necessidade de ter uma solução que permitisse a coleta de dados sem o uso de rede, com armazenamento local nos dispositivos móveis. Isso permite que haja coleta de dados em locais onde não há conexão com a internet ou há níveis mínimos de conexão.

1.2 Objetivos

O objetivo deste trabalho é viabilizar uma comunicação eficiente entre os aplicativos de aquisição de dados da avaliação de pós-ocupação e a aplicação *APOWeb*, usando as tecnologias SpringBoot, MongoDB, Maven e Java 8, que permitem agilidade no processo de configuração da aplicação, segurança na manipulação dos dados e um alto poder de desenvolvimento para aplicação e implementação das propostas apresentadas para a melhoria do *back-end* da plataforma *APO Digital*.

O *back-end* da proposta da plataforma *APO Digital* utiliza a tecnologia Spring Boot, que dá início a uma tendência geral no mundo do desenvolvimento onde se minimiza drasticamente os problemas no processo de configuração de aplicação e permite a concentração de esforços na parte de codificação do software. O *back-end* deve fornecer recursos aos aplicativos, e os mesmos devem fazer requisições que identifiquem quais recursos e como serão utilizados, baseando-se no padrão REST (*Transferência Representacional de Estado*) para as requisições.

A partir da definição de requisições REST, as transferências de dados serão realizadas segundo o padrão de objetos Java JSON. A formatação *JSON* ou *JavaScript Object Notation* é extremamente leve para troca de dados e baseia-se em um subconjunto da linguagem JavaScript. É um formato totalmente independente baseado no modelo de coleção de pares nome/valor e proporciona rapidez, confiabilidade e legibilidade na troca de informações.

O uso do formato JSON vem de encontro com a tendência de substituição de base de dados relacionais por uma estrutura de armazenamento mais flexível e dinâmica, como a estratégia orientada a documento. Desta forma, utilizando-se a abordagem orientada a documento proposta pelo MongoDB, o serviço de recepção dos dados coletados nos aplicativos e o serviço de recuperação/armazenamento de dados da *APOWeb* utilizam ambos o mesmo formato JSON. Ou seja, os dados obtidos em formato JSON podem ser mais facilmente compreendidos e manipulados.

A evolução da plataforma *APO Digital* tem como objetivo final proporcionar melhores recursos e tecnologias para aumentar a abrangência das aplicações de questionários habitacionais, bem como estender os campos de pesquisas habitacionais e sociais, facilitar o trabalho dos profissionais de arquitetura e afins que aplicam os questionários e por fim apresentar soluções e alternativas aos principais problemas relacionados a habitação na sociedade atual.

1.3 Organização da Monografia

O trabalho em questão foi dividido em 5 capítulos e está organizado da seguinte forma:

O capítulo 1 trata da motivação e objetivos do desenvolvimento desta aplicação, retratando os detalhes do projeto adotado pelo núcleo MORA de Pesquisa em Habitação e justificando os requisitos e as necessidades de desenvolvimento de um projeto para facilitar as aplicações de questionários.

O capítulo 2 retrata a revisão bibliográfica dos temas pertinentes a este trabalho, que foram de grande importância para todo o desenvolvimento do projeto: desenvolvimento de aplicações back-end para web, padrões de projeto, ferramentas como *Maven* e *IntelliJ*, Frameworks como *SpringBoot*, *Hibernate*, *JPA*.

O capítulo 3 aborda o desenvolvimento do trabalho e suas etapas. Foi apresentado neste capítulo o estado atual do sistema APO Digital, bem como as tecnologias adotadas, a proposta de comunicação entre os componentes, o escopo geral das funcionalidades da aplicação e todas as necessidades exigidas no desenvolvimento do projeto.

O capítulo 4 define a nova *API* criada e o algoritmo principal para a comunicação entre a aplicação web (*back-end*) e os aplicativos móveis (*front-end*). Traz também o processo de utilização da *API* entre as classes do projeto e o processo de deploy da aplicação web.

O capítulo 5 finaliza o trabalho abordando a importância da integração entre as frentes de pesquisa e o modelo de software desenvolvido, e as principais perspectivas de novos trabalhos para o Sistema APO Digital.

2 Revisão Bibliográfica

2.1 Serviço Web

O desenvolvimento da tecnologia de serviços web trouxe inúmeros benefícios para o desenvolvimento de aplicações baseadas em rede e permitiu que muitas empresas utilizassem estes serviços para integrar aplicações, normalmente desenvolvidas em plataformas de programação distintas. A grande vantagem do uso dos serviços Web é permitir a interoperabilidade, obtida através do uso de protocolos e padrões amplamente difundidos para a criação de recursos web.

Um serviço web é uma aplicação auto-contida, identificada por um URI (Uniform Resource Identifier), em que interfaces e ligações são definidas, descritas e localizadas por componentes que utilizam um mesmo protocolo (por exemplo: XML, JSON) (AUSTIN et al., 2002). Um serviço web deve ser capaz de interagir com diferentes aplicações proporcionando: independência de plataforma de hardware e software; baixo acoplamento devido à elevada granulosidade dos módulos e reusabilidade dos módulos característicos que aumenta a velocidade de integração destes (CRUZ, 2019).

O desenvolvimento de serviços web é normalmente baseado em elementos de software pré-existent, como *frameworks*, sendo o reuso de software então uma abordagem dominante para a construção de sistemas web. Os frameworks são coleções de classes abstratas e concretas que podem ser adaptadas e estendidas, como uma estrutura em esqueleto. As abstrações são implementadas com recursos como herança em classes de objetos que definem o comportamento do framework e estendem suas funcionalidades.

O framework *Spring*¹ surgiu em 2003 e fornece modelos de programação e configuração para aplicações Java em diferentes plataformas. Tem como objetivo prover infraestrutura para o nível de aplicação, permitindo que a equipe de programação possa se concentrar no problema a ser resolvido, a lógica do negócio. Ele é dividido em módulos, oferecendo suporte para desenvolvimento web, processamento de dados, transações e integração com outras tecnologias. Além disso, possibilita a injeção de dependências e provê suporte a testes de funcionalidades implementadas com seus módulos.

A atividade de gerenciamento de dependências pode ser bastante complexa no início de um projeto. Neste sentido, o framework Spring fornece uma ferramenta, denominada *Spring Boot* que permite iniciar rapidamente projetos Spring ao definir o gerenciamento das dependências do projeto a partir do princípio da Convenção sobre Configuração. Este

¹ Spring Framework
<https://spring.io/projects/spring-framework#learn>

princípio tem como objetivo diminuir o número de decisões a serem tomadas pelo programador no momento de configurar as diferentes áreas de uma aplicação.

O Spring Boot facilita a criação de aplicativos autônomos baseados em Spring e com ótimo grau de produção (SPRINGIO, 2019). Tem-se uma visão opinativa da plataforma Spring e de bibliotecas de terceiros para que qualquer um possa começar de forma simplificada. A maioria dos aplicativos Spring Boot necessita de baixas quantidades de configuração do Spring.

2.2 Tecnologias para Serviços web

O Spring Boot está entre as principais tecnologias utilizadas para construção de aplicações web, principalmente por seu poder auto construtivo, este *framework* veio para derrubar inúmeras barreiras construídas ao longo de anos no mundo do desenvolvimento software, sua escalabilidade, robustez e integração permite aos desenvolvedores vantagens como rapidez, facilidade e clareza no seu desenvolvimento além de inúmeras vantagens que serão citadas no decorrer deste artigo.

2.3 Banco de dados Não Relacionais

Atualmente houve um grande crescimento no uso de bancos de dados não relacionais que também são conhecidos por NoSQL

NoSQL é um movimento que promove soluções de armazenamento de dados não relacionais. Ele é composto por diversas ferramentas que, de forma particular e específica, resolvem problemas como tratamento de grandes volumes de dados, execução de consultas com baixa latência e modelos flexíveis de armazenamento de dados, como documentos XML ou JSON.

As tecnologias NoSQL não têm como objetivo substituir os bancos de dados relacionais, mas apenas propor algumas soluções que em determinados cenários são mais adequadas. Desta forma é possível trabalhar com tecnologias NoSQL e banco de dados relacionais dentro de uma mesma aplicação.

O artigo (MEDIA, 2011) demonstra algumas tecnologias Nosql e suas próprias características. O Redis adota o modelo chave-valor, utiliza a memória para alocação de dados e é ideal para utilização de Cache. O MongoDB implementa o modelo baseado em documentos, tem foco no tratamento de grandes volumes de dados e é ideal para grande parte das aplicações web. Já o Neo4j, uma das ferramentas NoSQL mais maduras, tem o modelo baseado em grafos e seus principais casos de uso estão relacionados a motores de recomendação, análise de rotas geográficas e redes sociais. Por fim, tem-se o Cassandra,

que é uma implementação open source do modelo de dados do BigTable com a arquitetura distribuída do Dynamo.

As ferramentas apresentadas neste artigo serão: Redis (chave-valor), MongoDB (documento), Neo4J (grafo) e Cassandra (família de colunas). Elas foram especialmente selecionadas de forma a representar os principais modelos de dados e diferentes arquiteturas.

2.4 Maven

Atualmente o Maven é uma das ferramentas de build mais utilizadas no mercado de desenvolvimento o que foi de bastante importancia para a sua utilização no desenvolvimento do projeto APO Digital, o maven foi desenvolvido pela *Apache Software Foundation* (ASF) conhecida também apenas como Apache Foundation ou Fundação Apache e é uma organização sem fins lucrativos criada para suportar os projetos de código aberto.

O Apache Maven é uma ferramenta de gerenciamento e compreensão de projetos de software. Com base no conceito de um modelo de objeto de projeto (POM), o Maven pode gerenciar a construção, o relatório e a documentação de um projeto a partir de uma informação central.

O Maven baixa bibliotecas Java e seus plug-ins dinamicamente de um ou mais repositórios, como o Maven 2 Central Repository, e armazena-os em uma área de cache local. Este cache local de artefatos baixados pode também ser atualizado com artefatos criados por projetos locais. Repositórios públicos podem também ser atualizados.

O Maven é construído utilizando uma arquitetura baseada em plugin, que permite que ele faça uso de qualquer aplicação controlável através da entrada padrão. Teoricamente, isto permitiria qualquer um escrever plugins para fazer interface com ferramentas de construção (compiladores, ferramentas de teste de unidade, etc.) para qualquer outra linguagem. De fato, o suporte e uso para linguagens diferentes de Java tem sido mínimas. Atualmente existe um plugin para o framework .NET e é mantido, e um plugin nativo C/C++ é mantido para o Maven 2 (MAVEN.ORG, 2019).

2.5 IntelliJ IDEA

Toda a implementação do projeto foi feita utilizando a IDE IntelliJ IDEA, ferramenta atualmente classificada entre as líderes de mercado e altamente utilizada e recomendada pelos desenvolvedores de software. O IntelliJ foi desenvolvido pela JetBrains e está disponível como uma edição da comunidade licenciada do Apache 2.

A plataforma IntelliJ é uma plataforma OSS desenvolvida pela JetBrains para a

construção de IDEs e ferramentas de desenvolvedor com reconhecimento de idioma. É usado pelo IntelliJ IDEA, Android Studio, Cursive, para citar alguns. Construído em Java, fornece uma abordagem de plataforma cruzada para a construção de ferramentas para qualquer tipo de linguagem, independentemente de ter como alvo a JVM ou não. Ele está disponível sob a licença Apache 2.0, permitindo a criação de produtos comerciais e não comerciais isentos de royalties, sejam eles IDEs ou plug-ins de pleno direito.

Os principais componentes da plataforma incluem um sistema de arquivos virtual, editor de texto, depurador da estrutura da interface do usuário, test runner. A plataforma também fornece APIs para criar linguagem personalizada e suporte à estrutura: navegação, conclusão de código, inspeções, intenções, refatorações, depurador, executores de teste, janelas de ferramentas, ações, etc., bem como lexers e analisadores gerais.

O JetBrains hospeda um repositório de plugins em que qualquer plug-in pode ser publicado e instalado em qualquer IDE criado na plataforma ([JETBRAINS, 2019](#)).

2.6 Java 8

Todo o projeto APO Digital foi desenvolvido utilizando a atualização 8 do Java, linguagem que é líder de mercado a mais de 10 anos.

O Java é a base para praticamente todos os tipos de aplicações em rede e é o padrão global para o desenvolvimento e distribuição de aplicações móveis e incorporadas, jogos, conteúdo baseado na Web e softwares corporativos. Com mais de 9 milhões de desenvolvedores em todo o mundo, de forma eficiente, o Java permite que você desenvolva, implante e use aplicações e serviços estimulantes

O Java foi testado, refinado, estendido e comprovado por uma comunidade dedicada de desenvolvedores, arquitetos e entusiastas do Java. O Java foi projetado para permitir o desenvolvimento de aplicações portáteis de alto desempenho para a mais ampla variedade possível de plataformas de computação. Ao disponibilizar aplicações entre ambientes heterogêneos, as empresas podem fornecer mais serviços e aumentar a produtividade, a comunicação e a colaboração do usuário final — além de reduzir drasticamente o custo de propriedade das aplicações da empresa e do consumidor. O Java tornou-se inestimável para os desenvolvedores.

o Java 8 é uma das últimas atualizações da linguagem contendo novos recursos que permitem ampliar muito a velocidade e os modos de desenvolvimento.

Aqui há um breve resumo das melhorias incluídas na release 8 do Java:

1. Métodos de Expressão Lambda e Extensão Virtual: O destaque do Java SE 8 é a implementação de expressões Lambda e funcionalidades de suporte para a linguagem

de programação e plataforma Java.

2. API de Data e Hora: Essa nova API permitirá que os desenvolvedores tratem data e hora de maneira mais natural, clara e fácil de entender.
3. Nashhorn JavaScript Engine: Uma nova implementação leve de alto desempenho do motor JavaScript foi integrada ao JDK e está disponível para aplicações Java por meio das APIs existentes.
4. Maior Segurança: A lista manual existente de métodos sensíveis do chamador foi substituída por um mecanismo que identifica com precisão esses métodos e permite que seus chamadores sejam descobertos com confiança ([JAVA.COM, 2019](#)).

3 Sistema APO Digital

O Sistema APO Digital propicia a construção de questionários de avaliação de pós-ocupação em moradias, contendo perguntas relativas às características socioeconômicas do morador, aspectos gerais da habitação, aspectos ambientais e de sustentabilidade, entre outros. Os esforços empreendidos na especificação e na construção do referido sistema se justificam no sentido de buscar possíveis soluções para as demandas de:

- Eficiência e confiabilidade dos resultados da avaliação obtidos por métodos quantitativos. Busca-se um conjunto de métodos avaliativos que forneça resultados cada vez mais fiéis à realidade da habitação.
- Privacidade dos moradores avaliados. Grande parte das dificuldades em avaliar habitações reside no fato do pesquisador ter de lidar com aspectos íntimos e particulares dos moradores.
- Possibilidade de maior interação entre o pesquisador e o morador na avaliação.
- Redução do tempo de avaliação. Técnicas muito extensas podem comprometer a qualidade dos resultados obtidos em avaliações de espaços habitacionais na medida em que o morador se sente cansado e desmotivado em responder à tantas questões.
- Redução dos custos da avaliação. As tradicionais aplicações de questionários e/ou outras técnicas impressas em papel podem acarretar custos altos para a pesquisa.
- Eficiência da tabulação dos resultados da avaliação.

Em consequência ao exame destas questões, buscou-se integrar a utilização de softwares e de meios digitais a fim de suprir as deficiências das análises convencionais, variando os métodos de avaliação. Neste contexto, foram construídos os seguinte componentes:

1. Aplicativos para dispositivos móveis, executados em plataforma Android, com apresentação das questões de diferentes APOs num formato interativo e utilizando técnicas de jogos digitais para criar um ambiente agradável e de imersão no contexto da APO. Um personagem interage com o morador respondente e conduz a apresentação das questões e a coleta dos dados. A confecção destes aplicativos seguiu a metodologia ágil SCRUM, sendo implementados na plataforma Corona SDK/Linguagem Lua ([ZAMMETTI, 2013](#); [IERUSALIMSKY et al., 1993](#)). São eles:

- a) *Meu Apê*: refere-se à uma APO própria para o estudo da forma de morar em edifícios de apartamentos.
 - b) *Como Você Mora?*: refere-se à uma APO ampliada para diferentes tipos de moradia, especificamente casas e edifícios de apartamentos.
2. A aplicação *APOWeb*, projetada em estrutura cliente-servidor, desenvolvida em linguagem Java e executada em plataforma web sendo acessível por qualquer navegador (Chrome, Firefox, Opera, Internet Explorer). A *APOWeb* permite a criação de novas APOs e o gerenciamento de APOs já realizadas, além do processamento de consultas.

Juntamente com os programas que compõem o sistema, foi desenvolvida uma base de dados para manter os registros de diversas APOs. Cada APO registrada na base contém um conjunto de técnicas, normalmente baseadas em questionários, e podem ser respondidas por um conjunto de usuários moradores. Na concepção desta pesquisa, uma APO deve ser aplicada à um conjunto de moradores utilizando-se interfaces gráficas desenhadas para dispositivos móveis. Assim, para cada APO respondida, as respostas são coletadas e armazenadas localmente, para em seguida serem enviadas para a base de dados principal. Para cada APO podem ser gerados relatórios e gráficos analíticos e visuais, como mostra a Figura 1.

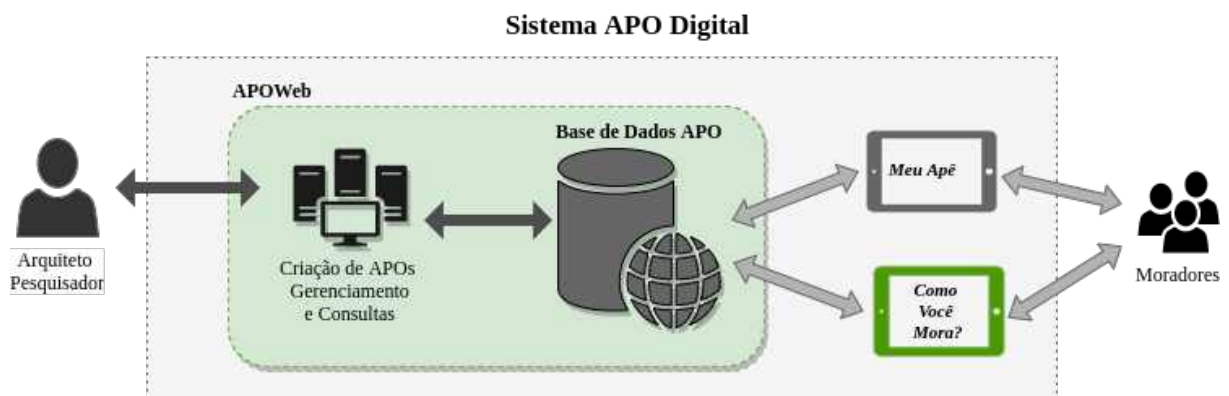


Figura 1 – Comunicação entre componentes do Sistema APO Digital

A *APOWeb* possibilita o gerenciamento das diversas técnicas propostas em APOs. As técnicas são baseadas em questionários e a aplicação permite que diferentes questionários sejam armazenados de acordo com a avaliação que se deseja realizar da pós-ocupação. Desta forma a *APOWeb* é responsável por fornecer as seguintes funcionalidades:

1. Criar e manter as APOs:
 - criar uma nova APO;

- inserir e editar técnicas/métodos de avaliação;
 - inserir e editar conceitos que devem ser avaliados (ventilação, nível de ruído, etc);
 - inserir e editar cômodos ou espaços;
 - inserir e editar categorias que delimitam cenários dentro de uma APO;
 - inserir e editar questões, associadas às suas categorias, com suas possibilidades de respostas; e
 - exibir uma grade contendo as diversas APOs e suas técnicas/métodos de avaliação.
2. Obter e armazenar as respostas dos usuários para as APOs em um repositório único, considerando que tais respostas são provenientes de dispositivos móveis.
 3. Responder à consultas sobre APOs realizadas.
 4. Gerar gráficos para perguntas e consultas específicas.

Cada questão contida numa técnica possui seu enunciado, uma descrição opcional e pode ser de quatro tipos, dependendo do tipo de resposta a ser fornecido:

- texto livre
- escolha única a partir de uma lista de opções
- escolha múltipla numa lista de opções
- seleção de prioridades numa lista de opções

Para as questões cujas respostas são de escolha única, múltipla ou seleção de prioridades deve-se definir a lista de opções a ser apresentada ao usuário morador. Para este fim são criados os grupos de qualificadores, como a lista bastante usual para coletar o nível de satisfação *{ Muito Bom, Bom, Regular, Ruim, Muito Ruim }*. A definição destes grupos permite a reutilização destes conjuntos de respostas para diferentes questões.

Neste trabalho, considera-se a evolução da aplicação *APOWeb*, implementada inicialmente em linguagem Java, segundo o padrão MVC, e fundamentada no arcabouço Spring Boot (GUTIERREZ, 2014), sendo a base de dados principal construída com o MongoDB.

Existem diversos *frameworks* para o desenvolvimento web em plataforma Java e boa parte destes *frameworks* implementam o padrão MVC. Para facilitar a configuração da aplicação web do Sistema APO Digital, utilizou-se o arcabouço Spring Boot¹ que é uma solução da Plataforma Spring² caracterizada pela *convenção sobre configuração*. Em conjunto com o String Boot (Versão 1.4) foram utilizados ainda:

¹ <http://projects.spring.io/spring-boot/>

² <https://spring.io/>

- *NPM*: O uso do NPM (*Node Package Manager*) se deu por dois motivos: (i) é um repositório *on-line* para publicação de projetos de código aberto para o Node.js; (ii) auxilia na instalação de pacotes, gerenciamento de versão e gerenciamento de dependências.
- *Bower*: O Bower é um gerenciador de pacotes *front-end* e foi utilizado para a administração dos pacotes responsáveis por gerenciar o que será consumido pelos usuários (pesquisadores/arquitetos) através do navegador (HTML, CSS, imagens).
- *Gulp.js*: Gulp.js é um kit de ferramentas open-source³ e foi escolhido para automatizar tarefas repetitivas relacionadas ao desenvolvimento *web* como minificação de *scripts*, concatenação, testes unitários, validação de sintaxe e otimização.
- *Maven*: O Maven é uma ferramenta responsável por diversas tarefas como: gerenciamento de dependências, compilação de código fonte, automatização de tarefas, entre outros. O Maven utiliza um arquivo XML (denominado *pom.xml*) para descrever o projeto. Esse arquivo contém dependências, tarefas a serem executadas, entre outras informações referentes ao projeto. Para a construção da *APOWeb* foi utilizada a versão 3.

Finalmente, foram utilizados ainda o Tomcat 8 (servidor *web* interno ao Spring Boot) e o sistema gerenciador de banco de dados MongoDB para manter o base das APOs.

³ <http://gulpjs.com/>

4 Desenvolvimento de Serviço Web para comunicação entre componentes do Sistema APO Digital

Com o avanço da utilização das tecnologias móveis, fez-se necessária a definição de uma arquitetura com modelo ágil e eficiente de comunicação entre aplicação web e aplicativos em dispositivos móveis. Para este fim, o desenvolvimento de serviços de comunicação, baseados em troca de informação padronizada, foi proposto para promover o envio e o recebimento de requisições provenientes de diferentes aplicativos, que colaboram na coleta e análise de dados em pesquisas em pós-ocupação.

4.1 Aprimoramento do componente APO Digital

Após a realização de diversas pesquisas de pós-ocupação com coleta de dados utilizando dispositivos móveis, identificou-se a dificuldade de conexão com a internet em alguns locais, situações de instabilidade e lentidão em conexões. Desta forma, optou-se por armazenamento temporário dos dados, em lugar de acesso em tempo real. O armazenamento temporário deve ficar associado à uma sincronização dos dados com a aplicação web no momento da finalização de uma coleta (via morador ao final do questionário), ou a depender da decisão do usuário (pesquisador) em enviar várias aplicações de questionários diferentes, contendo respostas/dados de diversos moradores que foram coletados *offline* e armazenados no dispositivo.

Assim, em consideração ao fato de que o envio das respostas em tempo real poderia causar muitos transtornos se um dos aplicadores fosse em um local em que não se pudesse conectar à internet, optou-se por trabalhar com a opção de salvar as informações localmente e posteriormente quando houvesse a possibilidade de conexão, o aplicativo forneceria a opção de enviar as respostas salvas localmente à base de dados principal, hospedada na APO Digital. As funções para que esta opção pudesse ser utilizada são objeto deste trabalho e estão descritas nesta seção.

As funções de comunicação entre aplicativos móveis e a APO Digital foram desenvolvidas na camada de *back-end* da APO Digital utilizando-se o *framework* Spring Boot. Este *framework* facilita o processo de configuração de propriedades de bibliotecas importadas no projeto, configuração de propriedades de execução do projeto, seja ela local ou em ambientes preparados como contêineres, servidores físicos, ou instâncias na nuvem e o processo de construção de *build* e compilação para geração dos pacotes executáveis da

aplicação, o que tornou muito mais rápida e simples a evolução do projeto em produção.

Todo o processo de comunicação com a base de dados principal está hospedada na APO Digital através de uma instância no MongoDB alocada no servidor físico de endereço IP: 200.131.206.11.

O desenvolvimento deste processo de comunicação com o banco de dados foi implementado usando os *frameworks hibernate* e *jpa*, sendo o *hibernate* responsável pela configuração e conexão com a base de dados principal e o *jpa* responsável pela comunicação entre o código em Java e o banco de dados, permitindo por exemplo a conversão de registros do banco de dados em objetos Java.

As tabelas ou entidades de banco de dados foram definidas como interfaces repository que através do *hibernate* proporciona que: sempre que o projeto é executado ocorre a criação das tabelas definidas na interface no banco de dados do MongoDB, e a biblioteca do Spring *MongoRepository* possui o objeto que pode ser estendido para criação das tabelas. Sempre que se cria uma entidade através dessa extensão cria-se um *bean* que é um componente completo gerenciado pelo spring. Desta forma, deve-se considerar apenas com o nome da tabela, pois o jpa fará todo o resto. Após a criação do bean através da biblioteca *MongoRepository* do spring, é possível implementar métodos que possuem função de busca no banco de dados sem ser necessário utilizar a linguagem nativa do MongoDB.

A figura 2 mostra o exemplo de uma criação de uma tabela utilizando o *MongoRepository* e exemplo de métodos com função de busca.

```
package br.ufu.faued.repository;

import br.ufu.faued.domain.MobileResponse;
import org.springframework.data.mongodb.repository.MongoRepository;
import java.util.List;

public interface MobileRepository extends MongoRepository<MobileResponse, String> {

    MobileResponse findTopByOrderByIdDesc();

    List<MobileResponse> findByDwellerId(Integer dwellerId);
}
```

Figura 2 – Exemplo de classe que cria tabela no MongoDB

Como informado no parágrafo anterior o banco de dados utilizado na APO Digital é o MongoDB, que é um banco de dados não relacional que usa o recurso de registro dos dados em índices e arquivos. A escolha do MongoDB se deu pela rapidez e robustez na manipulação dos dados, e esta vantagem viabiliza o uso de bancos de dados não relacionais, que são indicados para aplicações que manipulam grandes volumes de dados e que

necessitam de rapidez para retorno dos dados. Neste sentido, a APO Digital tem como finalidade oferecer uma plataforma digital para a construção de pesquisas de pós-ocupação, que poderá ser utilizada por diversos pesquisadores e instituições como o IPEA ([VILLA et al., 2015](#)).

A aplicação está dividida em seis pacotes que compõem toda a estrutura do projeto:

1. O pacote *domain* possui todos os objetos do projeto bem como seus atributos. Cada objeto é composto pela classe, atributos e métodos *get* e *set* utilizados para a manipulação dos objetos em outras classes.
2. O pacote *config* possui as configurações para comunicação web no projeto, essas configurações possuem headers constantes enviados nas requisições e classe para liberação do Cross-Origin Resource Sharing (Compartilhamento de recursos com origens diferentes) OU CORS, trata-se de um mecanismo que utiliza cabeçalhos Http para informar a um navegador que permita que um aplicativo web possa acessar recursos da aplicação, assim com o pacote desenvolvido e possível fornecer permissão web no servidor para que o front possa efetuar as requisições.
3. O pacote *exception* possui as classes com os tratamentos de exceções que seriam erros mapeados na aplicações. Estes tratamentos podem ser usados pelos métodos do pacotes *service* e pelos métodos de requisição do pacote *rest*.
4. O pacote *repository* possui a instanciação das entidades no banco de dados bem como as consultas pré-escritas utilizando as classes criadas pelo JPA.
5. O pacote *rest* possui as URLs dos serviços que serão chamados pelo front. Estes serviços são *beans* criados para a camada de controle, todas essas URLs são mapeadas em métodos que são chamados pelo front através de requisições no padrão REST.
6. O pacote *service* possui todos os métodos com a lógica de serviço das aplicações e formam a camada *service*. Os serviços são chamados pelos métodos de requisição *rest* e implementam os serviços de criação, leitura, alteração e exclusão dos dados da aplicação.
7. O pacote *util* possui métodos de conversões de datas, separadores de strings, manipuladores de strings a fim de tratar palavras com acentos e metodos de compressão e descompressão de string afim de executar transições entre as aplicações.

4.2 Infraestrutura de ambiente em produção

Atualmente o projeto possui a camada *front-end* da aplicação Web acoplada ao mesmo projeto do *back-end*. A camada *front-end* foi implementada utilizando-se o *fra-*

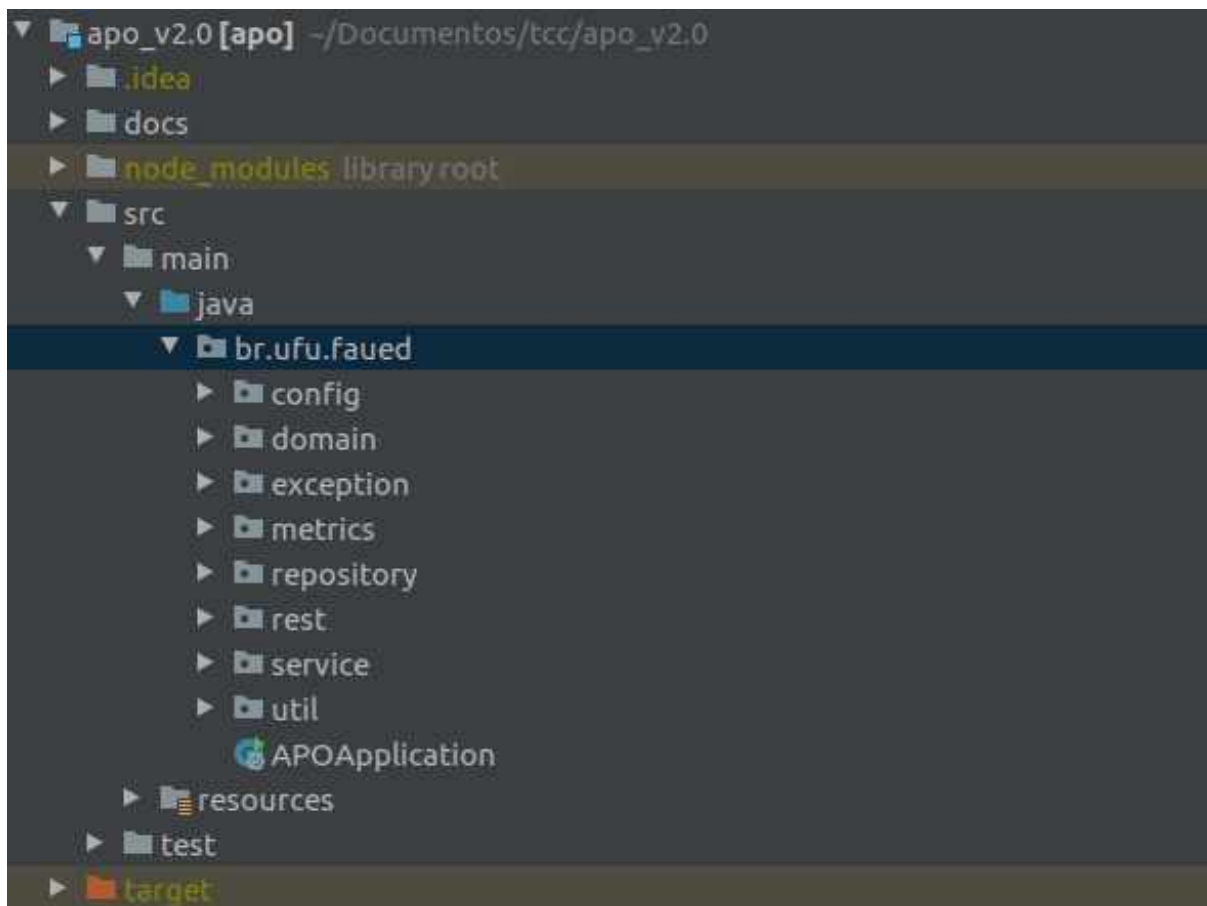


Figura 3 – Pacotes da aplicação APO

mework Angular e as linguagens HTML, CSS e JavaScript. As requisições são realizadas a partir do *front-end* em direção ao *back-end* através do uso de Rest client, que são requisições no padrão do protocolo HTTP de rede.

As partes *front-end* Web e *back-end* estão atualmente em um servidor físico alocado no CTI(Centro de Tecnologia e informação) localizado na Universidade Federal de Uberlândia, o servidor está acoplado em uma máquina de 32 bits com sistema operacional Ubuntu 16.04.6 LTS com memória de 20.44GB.

O processo de deploy da aplicação é feito através de conexão ssh com servidor, para isto é necessário ter o SSH instalado na máquina e utilizar o terminal para os seguintes passos abaixo:

1. Geração de Build da aplicação *back-end* através do comando no Maven *mvn clean install*, este comando basicamente limpa os arquivos da pasta target, compila todos os arquivos java, testa os arquivos, gera arquivos compilados .class e salva novamente na pasta target, Após o build da aplicação será gerado um arquivo .jar dentro da pasta target no projeto, executável gerado a partir das classe compiladas da pasta

target.

2. O arquivo .jar deverá ser enviado ao servidor utilizando o comando 3 da Figura 4, por padrão o arquivo deve ser enviado para a pasta apo onde foi configurado a infraestrutura de logs para a aplicação.
3. Depois de enviar o arquivo .jar ao servidor será necessário estabelecer conexão com o servidor *ssh* utilizando o comando 1 da Figura 4.
4. Após estabelecer uma conexão *ssh* com o servidor será necessário encerrar o processo que estava rodando anteriormente caso o jar esteja em execução, o processo roda na porta 8080 desta forma pode-se utilizar comando 4 da Figura 4 para localizar o processo que está rodando e posteriormente utilizar comando 5 da Figura 4 para encerrar o processo(o PID refere-se ao id do processo rodando na porta 8080 retornado no comando 4).
5. Depois de encerrado o processo deve-se rodar o novo arquivo jar utilizando o comando 7 da Figura 4.
6. Após a execução do arquivo aguardar cerca de 1 minuto e verificar se aplicação está rodando no servidor através da URL contida no item 1.

```
1 - ssh madriana@200.131.206.11 -p 1026
2 - rm -rf apo.jar
3 - scp -P 1026 /<caminho>/apo_v2.0/target/apo.jar madriana@200.131.206.11:apo
4 - fuser -n tcp 8080
5 - kill -9 PID
7 - nohup java -jar apo.jar &
```

Figura 4 – Comandos Servidor da Aplicação

Caso ocorram erros na aplicação pode ser necessário realizar uma verificação logs de erros no servidor, para isso em diversas partes da implementação do código foi utilizado o *framework log4j* do java 8 que ao ser chamada no código insere mensagens de log especificadas no servidor de execução.

Na infraestrutura do servidor foi configurado o salvamento dos logs da aplicação no arquivo nohup.out localizado na pasta apo do servidor, assim facilita-se a busca de possíveis erros que ocorram na aplicação enquanto ela estiver em execução no servidor.

Qualquer execução da aplicação salva o *stackTrace*(pilha de execução) no arquivo *nohup.out*, o que é de extrema importância para acompanhamento da aplicação.

4.3 Comunicação com os aplicativos via REST

Foi implementada nesta etapa a comunicação entre aplicativos Móvel e aplicação *back-end* externa, sendo esta comunicação totalmente apartada, isto é, aplicações *back-end* e *front-end* Móvel em repositórios diferentes e utilizando o padrão REST para comunicação entre si.

O padrão *REST* é um dos padrões mais avançados de comunicação Web atualmente, este padrão é considerado uma arquitetura de software que define uma junção de restrições e regras a serem usadas para a criação de *Web Services* ou Serviços Web, os *Web Services* que estão em conformidade com o estilo de arquitetura *REST* são denominados serviços *RESTful*, Serviços *RESTful* ligam-se e operam exibições textuais de recursos web usando um conjunto de uniforme e já definido de execuções *sem estado* ou requisições independentes.

Muito importante citar que o padrão *REST* foi utilizado pela primeira vez na *World Wide Web* com documentos ou arquivos identificados em suas URLs, tornando-se assim o principal *case* de sucesso e referência para utilização em diversas tecnologias do desenvolvimento de software.

No pacote Rest desenvolveu-se classes com os métodos que formaram a API e seus recursos, onde classes são componentes *RestController* que definem o bean para a requisições de recursos da API.

A notação identificada na classe pelo anotation *RestController*, escrito antes da criação da classe, define todas as URLs das APIs. Estas URLs são formadas por uma junção entre uma string e um método Rest, a String da URL precisa do serviço da aplicação e do endereço onde a aplicação é executada, essa junção forma uma espécie de endereço do recurso, onde aplicações móveis, webs ou ferramentas para testes em serviços *RESTful* podem efetuar requisições a estes recursos.

Em uma classe que define a API pode possuir diversos recursos, a classe com Componente *RestController* pode possuir uma parte da URL comum a todos os outros recursos, esta pode ser definida na anotation *RequestMapping* antes do inicio da classe conforme a figura 5. As APIs definidas na classe do componente *RestController* se comunicam com outros componentes para manipulação e busca dos dados e estes precisam ser passados no componente *RestController* através do anotation *Autowired* que libera as dependências das classes utilizadas, existem inúmeros métodos REST, os principais utilizados na APO Digital são *GET*, *POST*, *PUT* e *DELETE* e possuem as características

```
package br.ufu.faued.rest;

import br.ufu.faued.domain.DwellerIdGenerateResponse;
import br.ufu.faued.domain.DwellerIdResponse;
import br.ufu.faued.domain.MobileResponse;
import br.ufu.faued.repository.DwellerIdResgisterRepository;
import br.ufu.faued.repository.MobileRepository;
import br.ufu.faued.service.MobileService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;
import java.util.UUID;

@RestController
@RequestMapping("api/mobile")
public class MobileController {

    @Autowired
    MobileRepository mobileRepository;

    @Autowired
    MobileService mobileService;

    @Autowired
    DwellerIdResgisterRepository dwellerIdResgisterRepository;
```

Figura 5 – classe MobileController do pacote Rest

abaixo.

1. *GET* - Refere-se a Requisições de busca de informações já criadas anteriormente por um recurso na base de dados
2. *POST* - Refere-se a Requisições de criação de recursos com informações novas na base de dados
3. *PUT* - Refere-se a Requisições de alteração de recursos com informações já salvas na base de dados
4. *DELETE* - Refere-se a Requisições para exclusão de um recurso com informações salvas na base de dados

Para qualquer Requisição Rest, como se trata de uma requisição utilizando o protocolo de rede HTTP, há sempre a resposta com um conjunto de informações. Entre essas informações está o status que é representando por um código numérico.

Existem inúmeros códigos definidos no protocolo HTTP, os mais utilizados no Back-end do APO Digital são HTTP 200, 400, 404 e 500, a figura 10 mostra um exemplo de requisição onde é retornado o status HTTP 200, o retorno desses status já são pré-configurados no Spring Boot juntamente com o padrão Rest e podem ser alterados na aplicação de acordo com a necessidade de cada recurso. As características dos códigos retornados na aplicação APO são descritos abaixo.

1. HTTP 200 - caso ocorra uma resposta de sucesso.
2. HTTP 404 - quando a requisição de um recurso não é encontrada
3. HTTP 400 - caso seja uma requisição por parâmetros inválidos
4. HTTP 500 - caso seja um erro inesperado pela aplicação, causados por falhas no desenvolvimento ou recebimento de parâmetros inesperados na requisição.

A figura 6 mostra uma requisição onde o recurso não foi encontrado pois não há na classe controller um recuso com a URL `"/api/mobile/sav"`.

A figura 7 mostra uma requisição com parâmetros inválidos pois não foi passado o corpo da requisição que é exigido pelo recurso.

A figura 8 mostra um exemplo de um Recurso da *API MobileController* com URL: `"api/mobile/getDwellerId/dwellerId"` e método *GET*, este recurso retorna um *JSON* com os dados da tabela *mobileResponse* para o morador passado como parâmetro entre as chaves na URL.

A figura 9 mostra um exemplo de Recurso da *API MobileController* com URL: `"api/controller/save"` e método *POST*, este recurso salva dos os dados de uma resposta, passando no corpo da requisição, conforme visto a URL é definido dentro de uma *annotation* java que já indica o método a ser utilizado após o simbolo arroba.

A figura 10 mostra um exemplo de requisição do tipo *POST*

4.4 Estrutura de comunicação entre aplicativos móveis e APO Digital

A Plataforma APO foi atualizada com uma estrutura que propicia a comunicação dos aplicativos com a aplicação web, considerando-se a possível ausência de conexão à internet. Para esta situação foi criada uma tabela de transição denominada *MobileResponse*.

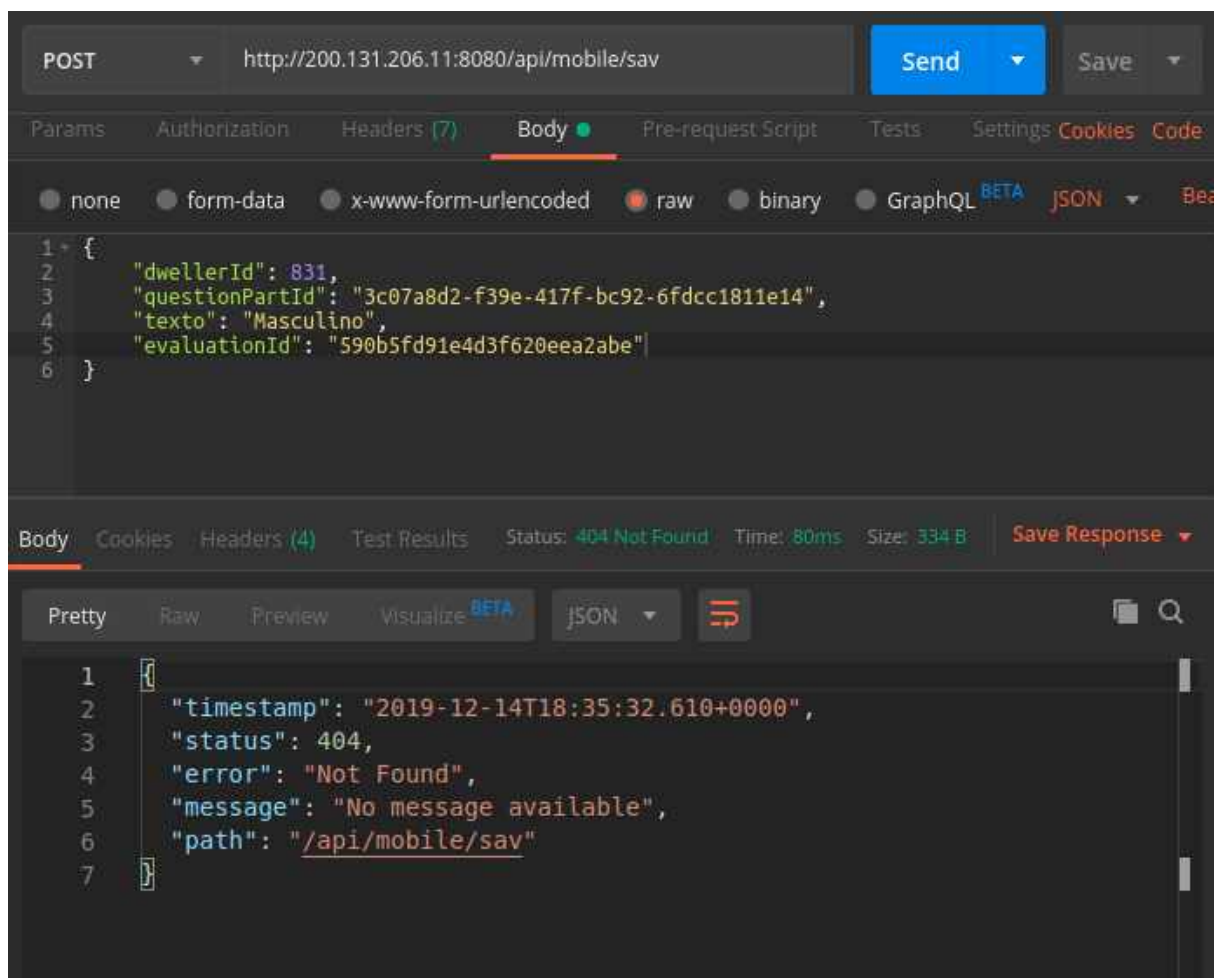


Figura 6 – Requisição postMan com erro 404

Para esta tabela serão enviadas as informações de respostas dos questionários respondidos pelos moradores através dos serviços criados: (i) para o salvamento de cada resposta, (ii) para salvamento de lista de respostas suportando qualquer quantidade de respostas e (iii) para limpeza da base *MobileResponse*. Foi criada uma rotina que roda todos os dias à 01h00min da manhã, que consome todos os dados na tabela *MobileResponse* naquele momento e, após consumir, inicia o processamento para incluir todas as informações das respostas dos moradores na tabela principal de respostas, a tabela *EvaluationAnswers*.

Foi criado um algoritmo para o processamento das respostas recebidas e armazenadas na tabela *MobileResponse*, que faz o controle temporário, uma vez que as repostas dos moradores podem ser enviadas incompletas em tempo real. Assim, esse algoritmo define a montagem de objetos que correspondem às respostas de um questionário para cada morador, de acordo com a hierarquia definida na tabela *EvaluationAnswers*. Todas as respostas de um morador são convertidas uma instância de árvore de respostas e, após a montagem de todos os objetos, o algoritmo salva as informações de todos os questionários que foram efetuados nos diversos aplicativos em dispositivos móveis. As respostas orga-

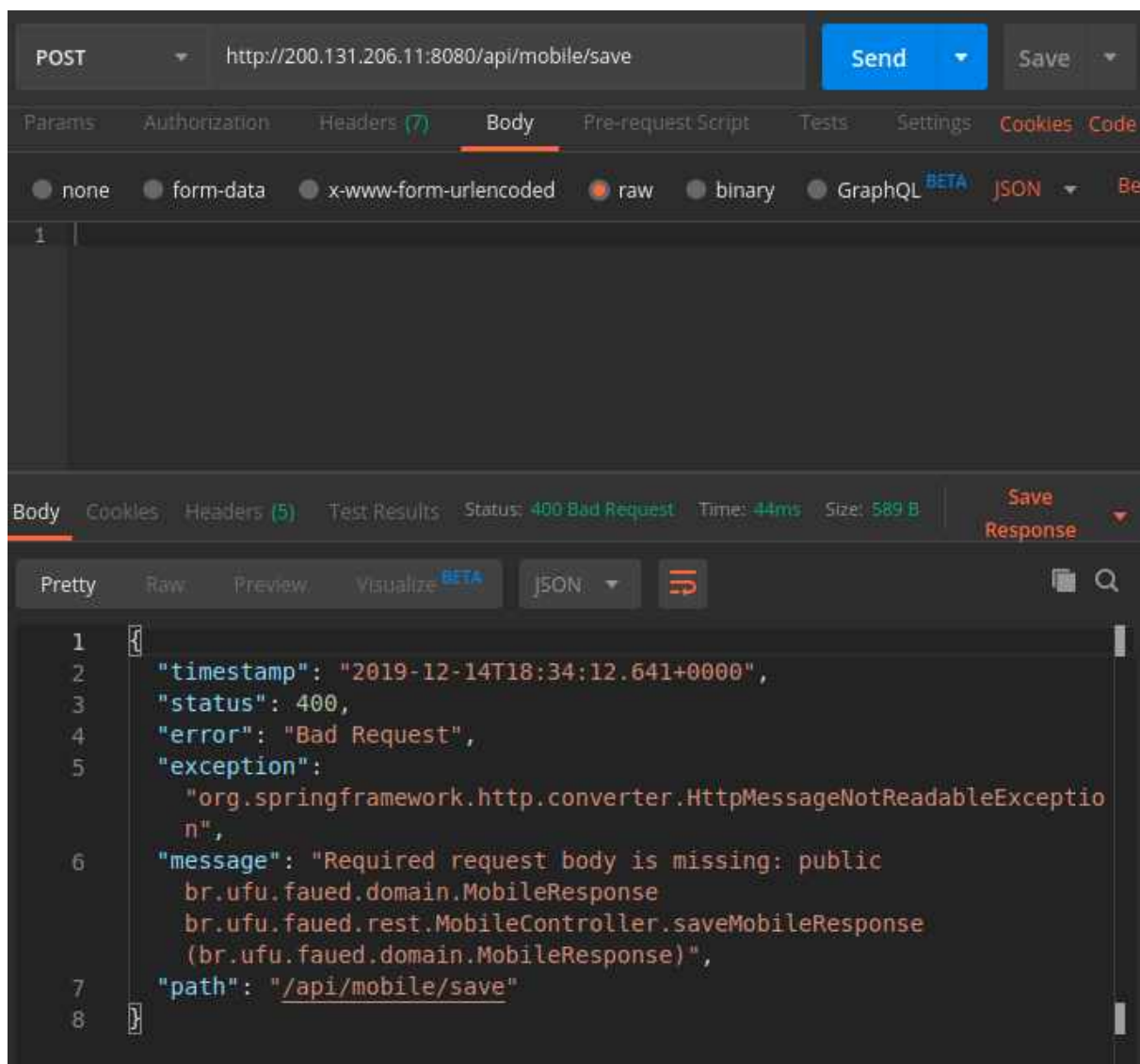


Figura 7 – Requisição postMan com erro 400

```
@GetMapping(value = "/getDwellerbyId/{dwellerId}")
public List<MobileResponse> getDwellerbyId(@PathVariable(value = "dwellerId") String dwellerId){
    return mobileRepository.findByDwellerId(Integer.parseInt(dwellerId));
}
```

Figura 8 – Requisição com método GET APO Digital

nizadas na tabela *EvaluationAnswers* possibilitam a confecção de relatórios e estruturas gráficas de visualização das respostas de todos os moradores. Diversas análises na área da pós-ocupação em moradias podem ser realizadas sobre os dados coletados, analisados e tabulados em relatórios e gráficos, considerando os conceitos abordados nos questionários como: características socioeconômicas do morador, aspectos gerais da habitação, aspectos

```
@PostMapping(value = "/save")
public MobileResponse saveMobileResponse(@Valid @RequestBody MobileResponse mobileResponse){
    mobileResponse.setId(UUID.randomUUID().toString());
    return mobileRepository.save(mobileResponse);
}
```

Figura 9 – Requisição com método POST APO Digital

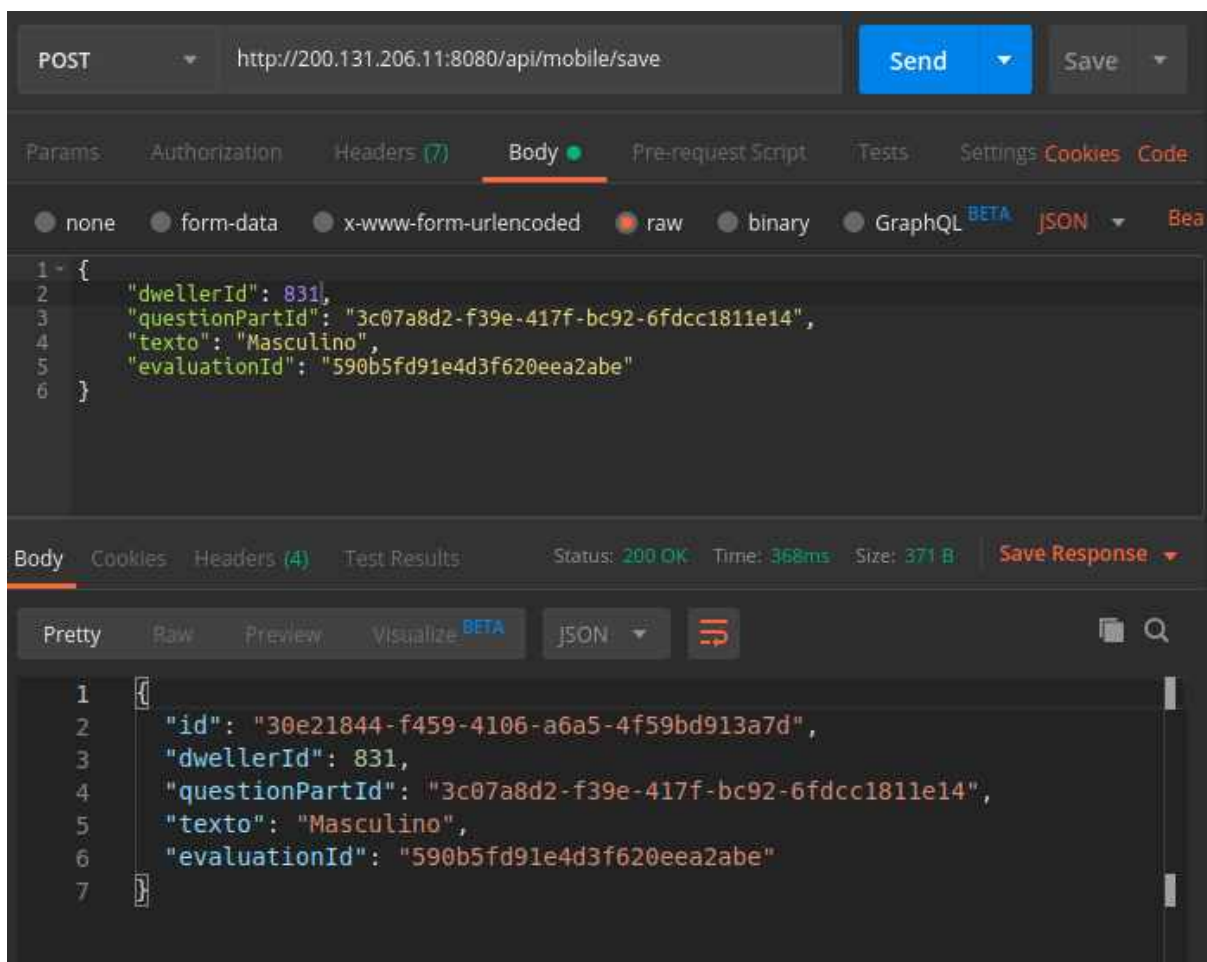


Figura 10 – Requisição com método POST APO Digital

ambientais, entre outros. O algoritmo que processa as respostas coletadas está definido na Figura 3.

A Figura 11 mostra especificamente o método *createObjectfromMobileResponse* que implementa o algoritmo de criação de um objeto para cada questionário respondido. O objetivo é buscar todos os dados (respostas) de um determinado morador na tabela *MobileResponse* e criar um novo objeto de tipo *EvaluationAnswer*. Após a instanciação do novo objeto, este é atualizado e preenchido com as respostas coletadas e armazenadas na estrutura temporária. Os passos que foram definidos neste algoritmo são os seguintes:

```

public EvaluationAnswer createObjectfromMobileResponse(String dwellerId) {
    List<MobileResponse> mobileResponseList = mobileRepository.findByDwellerId(Integer.parseInt(dwellerId));
    EvaluationAnswer evaluationAnswer = new EvaluationAnswer();
    evaluationAnswer.setId(UUID.randomUUID().toString());
    evaluationAnswer.setDwellerId(dwellerId);
    evaluationAnswer.setEvaluationId(mobileResponseList.get(0).getEvaluationId());
    List<Answer> answerList = new ArrayList<>();
    List<AnswerPart> answerPartList = new ArrayList<>();
    mobileResponseList.forEach(t -> {
        AnswerPart answerPart = new AnswerPart();
        answerPart.setQuestionPartId(t.getQuestionPartId());
        answerPart.setText(t.getText());
        answerPartList.add(answerPart);
    });
    List<String> questionsFromEvaluation = findAllQuestionsOnEvaluation(mobileResponseList.get(0)
        .getEvaluationId());
    questionsFromEvaluation.forEach(t -> {
        Answer answer = new Answer();
        answer.setQuestionId(t);
        List<AnswerPart> answerParts = new ArrayList<>();
        answerPartList.forEach(l -> {
            if (findQuestionIdWithQuestionsParts(l.getQuestionPartId(), mobileResponseList.get(0)
                .getEvaluationId()).equals(answer.getQuestionId())) {
                answerParts.add(l);
            }
        });
        answer.setAnswerParts(answerParts);
        answerList.add(answer);
    });
    evaluationAnswer.setAnswers(answerList);

    return evaluationAnswer;
}

```

Figura 11 – Algoritmo para montagem dos Objetos com as respostas dos Questionários

1. criar um id único para o conjunto de respostas de um morador ao questionário.
2. atribuir o id do morador recebido como argumento de entrada do método.
3. atribuir o id do questionário a partir da primeira resposta do um morador.
4. instanciar uma nova lista nomeada *mobileResponseList* com as respostas do morador.
5. instanciar uma nova lista com nome *answerPartList* para todas as sub-respostas para perguntas do item 4.
6. a partir da lista *mobileResponseList* montada na busca de todos os ids de moradores na tabela *MobileResponse* inicia-se um processo de percorrer todos dos dados, foi utilizado para o método *forEach* da biblioteca *java lang*, este método faz a implementação do for de forma que se pode utilizar o lambda para percorrer uma lista, a utilização do lambda com *forEach* diminui muito o volume de código no projeto e facilita a interpretação do mesmo, no primeiro for para cada *questionPartId* é criado uma novo objeto juntando o id da *questionPart* com o texto correspondente a resposta, depois é feita uma busca a partir do id do questionário para retornar todos

os ids da perguntas deste questionário, depois todas a perguntas são percorridas por um novo for, para cada pergunta é criado um novo objeto resposta que é formado de uma lista com as questionPartids da pergunta, além disso para cada pergunta é criado um novo laço for que analisa se a questionPartId é equivalente a ao id da pergunta, se equivalente ele insere na lista resposta.

```
public List<EvaluationAnswer> saveAllTableMobileResponseAndClean(){
    List<Integer> dwellerId = new ArrayList<>();
    List<EvaluationAnswer> evaluationAnswers = new ArrayList<>();
    List<MobileResponse> mobileResponseList = mobileRepository.findAll();
    mobileResponseList.forEach(o -> {
        if(!dwellerId.contains(o.getDwellerId())){
            dwellerId.add(o.getDwellerId());
        }
    });
    dwellerId.forEach(t -> {
        evaluationAnswers.add(createObjectfromMobileResponse(Integer.toString(t)));
    });
    evaluationAnswerRepository.save(evaluationAnswers);
    log.info("Job Evaluation saved answers");
    mobileRepository.deleteAll();
    log.info("Job Evaluation cleaned table Mobile Response");
    return evaluationAnswers;
}
```

Figura 12 – Algoritmo para montagem dos Objetos com as respostas dos Questionários

A Figura 12 mostra o método que implementa a criação de todos os objetos de respostas de questionários salvas na Tabela *MobileResponse* e convertidas para a Tabela *EvaluationAnswers*.

Inicialmente foi instanciada uma lista de inteiros nomeada de *dwellerId* com o objetivo de salvar o id de todos os moradores, depois foi instanciada uma lista que receberá uma lista com todas as respostas de moradores já objeto do tipo esperado pela tabela *EvaluationAnswer*, posteriormente através do método *findAll* implementado na classe *mobileRepository* é feita a busca de todas as respostas salvas na tabela *MobileResponse*, as informações da tabela são montadas em uma lista nomeada *mobileResponseList* com objetos do tipo *MobileResponse*.

Após o salvamento de todas as respostas na lista *mobileResponseList* é iniciado a execução de um laço tipo *for* através do *forEach* do java 8, para percorrer toda a lista, para cada uma das resposta é verificado se o id do morador está contido na lista *dwellerId* instanciada no inicio do método, o laço *for* possui a seguinte verificação através de um laço para cada resposta da lista *mobileResponseList*, caso o id do morador da resposta esteja

na lista de ids de moradores instanciada no início do método ele insere o id do morador na lista, caso não esteja na lista ele segue execução do laço para a próxima resposta até terminar a comparação com todas as respostas, o objetivo desse laço é montar a lista com todos os moradores que responderam alguma pergunta de algum questionário que esteja com a resposta salva na tabela *MobileResponse*.

Após obter todos os moradores com respostas na tabela *MobileResponse*, é criado um novo laço que percorre todos os ids dos moradores, para cada id de cada morador é chamado o método *createObjectfromMobileResponse* que cria o objeto para montar a árvore com respostas completas e atributos esperados na tabela *EvaluationAnswer* e adiciona na lista *evaluationsAnswers* instanciada no início do método..

Após a criação de todas as árvores com as respostas de todos os moradores, a lista *evaluationsAnswers* é salva na tabela *EvaluationAnswers* do Mongo, após o salvamento o logger salva no servidor a mensagem "Job Evaluation saved answers", após esta mensagem todos os dados da tabela *MobileResponse* são deletados e o logger envia para o servidor a mensagem "Job Evaluation cleaned table Mobile Response".

```
@Scheduled(cron = "0 0 1 * * *")
public void runJob(){
    log.info("Job Evaluation Answer Start Run");
    try {
        evaluationService.saveAllTableMobileResponseAndClean();
    } catch (Exception e){
        log.error("Not possible save answers",e);
    }
    log.info("Job Evaluation Answer End Run");
}
```

Figura 13 – Algoritmo de rotina para montagem dos Objetos com as respostas dos Questionários

A figura 13 mostra o método que inicia a rotina de atualização todos os dias a 01:00hs da manhã, nela foi utilizado um anotation criado a partir do java 8, o *@Scheduled* que implementa uma rotina de execução de acordo com o tempo informado no parâmetro cron, no parâmetro podemos implementar qualquer tempo de execução, quando o chegar na hora informada no cron, o método inicia sua execução, este método chama outro método que é responsável pelo salvamento de todas as respostas dos questionários respondidos e salvos na tabela *MobileResponse*, o método que inicia a rotina de atualização lança o log "Job Evaluation Answer Start Run" no servidor e assim que inicia sua execução, foi implementado o método *saveAllTableMobileResponseAndClean* que tem

o objetivo de chamar os métodos de criação dos questionários e posteriormente após a montagem das arvores com todas as respostas dos questionários da APO Digital, limpar todos os dados da tabela *MobileResponse*, a limpeza dos dados da tabela é feita apenas no final da execução do método, o objetivo dessa estratégia é garantir que todos os dados estejam salvos antes da exclusão dos dados da tabela *MobileResponse* do Mongo, assim para manter a segurança dos dados dos questionários respondidos moradores, este método foi implementado dentro de um recurso do java nomeado *"try catch"*. Esse recurso permite iniciar a execução de método e executar tratativas diferentes em caso de falha ou sucesso na execução do método.

Caso ocorra sucesso na execução do método *saveAllTableMobileResponseAndClean*, a rotina termina a execução e gera um *log* no servidor com mensagem *"Job Evaluation Answer End Run"* indicando o fim da execução da rotina.

Caso ocorra alguma falha na execução do método *saveAllTableMobileResponseAndClean* seja ela por recursos da aplicação ou dados incorretos, como o método foi definido dentro de um *try catch*, a execução é cancelada e nenhuma informação é modificada no banco de dados sejam elas inserção dos dados na Tabela *EvaluationAnswer* ou exclusão dos dados na tabela *MobileResponse*, assim é lançada uma exceção com a mensagem *"Not possible save answers"*, a exceção também gera um *log* no servidor da aplicação com a mesma mensagem, assim será possível identificar quando conectado ao servidor, o exato momento em que ocorreu a falha e a exceção foi lançada, da mesma forma ocorre o fim da execução da Rotina e o método *logger* lança a mensagem *"Job Evaluation Answer End Run"* mas sem nenhuma alteração dos dados desta vez.

```
@PostMapping(value = "/save")
public MobileResponse saveMobileResponse(@Valid @RequestBody MobileResponse mobileResponse){
    mobileResponse.setId(UUID.randomUUID().toString());
    return mobileRepository.save(mobileResponse);
}

@GetMapping(value = "/save/other")
public MobileResponse saveMobileResponseGet(@Valid @RequestBody MobileResponse mobileResponse){
    return mobileRepository.save(mobileResponse);
}

@PostMapping(value = "/saveListResponse")
public List<MobileResponse> saveListResponse(@Valid @RequestBody List<MobileResponse> mobileResponseList){
    mobileResponseList.forEach(mobileResponse -> mobileResponse.setId(UUID.randomUUID().toString()));
    return mobileRepository.save(mobileResponseList);
}

@DeleteMapping(value = "deleteAll")
public void deleteAll() { mobileRepository.deleteAll(); }
```

Figura 14 – Requisições controller do aplicativo

A figura 14 mostra os recursos criados para a API de comunicação entre o aplicativo

e a aplicação back-end, o objetivo da criação dessa API partiu-se dos casos onde as condições de conexão a internet são extrema-mentes ruins, pois essa API faz comunicação com uma tabela considerada provisória para os dados de resposta dos questionários, abaixo segue detalhamento dos recursos implementados nesta API.

1. o recurso com URL *"/save"* com método *REST* do tipo *POST* salva uma resposta na tabela *MobileResponse*, nesta resposta o id dos dados é criado no momento da requisição sendo ele um atribuído representado por um código hash único.
2. o recurso com URL *"/save/other"* com método *REST* do tipo *GET* salva uma resposta simples na tabela *MobileResponse*, (serviço descontinuado)
3. o recurso com URL *"/saveListResponse"* com método *REST* do tipo *POST* salva uma lista de respostas, para cada resposta é atribuído um id hash único.
4. o recurso com URL *"deleteAll"* com método *REST* do tipo *DELETE* deleta todos os dados da tabela *MobileResponse*

```
@PostMapping(value = "/postdwellerId")
public DwellerIdGenerateResponse postDwellerIdResponse(){
    DwellerIdGenerateResponse dwellerIdGenerateResponse = mobileService.autoGenerateDwellerId();

    return dwellerIdRegisterRepository.save(dwellerIdGenerateResponse);
}
```

Figura 15 – Requisição utilizado pelo aplicativo móvel para gerar o id do morador

A figura 15 mostra a requisição do serviço de geração de ids para cada morador, o serviço com requisição *REST* do tipo *POST* incrementa um id na tabela *dwellerIdRegister* toda vez que for acionado. neste serviço inicialmente é gerado um id incrementado da tabela *DwellerIdRegister* e posteriormente esse id é salvo utilizando o método *save* implementado no *JPA* da classe *dwellerIdRegisterRepository*.

A figura 16 mostra o algoritmo implementado na camada service para geração dos ids dos moradores, o algoritmo é um método que sempre que chamado, instancia um objeto do tipo *DwellerIdGenerateResponse*, trata-se uma resposta para o controller que indica o id a ser utilizado pelo aplicativo, o id é do tipo Long nomeado de *DwellerIdGenerate* e é incrementado sequencialmente de forma ascendente a partir do 1.

O método verifica a quantidade de dados na tabela através do primeiro laço, se a quantidade for igual a 0, a variável *DwellerIdGenerate* é setada com valor 1, caso contrário segue sua execução.

No segundo laço, se a quantidade for maior ou igual a 1, a variável *DwellerIdGenerate* é setada com o retorno da quantidade somada a 1.

```
@Override
public DwellerIdGenerateResponse autoGenerateDwellerId(){

    DwellerIdGenerateResponse dwellerIdGenerateResponse = new DwellerIdGenerateResponse();

    if(dwellerIdResgisterRepository.count() == 0){
        dwellerIdGenerateResponse.setDwellerIdGenerate(1L);
    }

    if(dwellerIdResgisterRepository.count() >= 1){
        dwellerIdGenerateResponse.setDwellerIdGenerate(Long.valueOf(dwellerIdResgisterRepository.count() + 1));
    }

    return dwellerIdGenerateResponse;
}
```

Figura 16 – Algoritmo para geração do id do morador

Ao final do método é retornado o objeto *DwellerIdGenerateResponse* instanciado no início com o atributo *DwellerIdGenerate* atualizado.

Os aplicativos chamarão este recurso sempre que um morador começar a responder um questionário, com o objetivo de identifica-lo no banco de dados do APO Digital.

4.5 Padrões de Projeto APO Digital

Alguns padrões de projetos foram utilizados durante o desenvolvimento do projeto com o objetivo de otimizar a implementação, melhorar a reutilização das soluções e facilitar a compreensão do código fonte.

Toda a implementação foi baseada no modelo de programação para interfaces.

O padrão *iterator* foi utilizado diversas vezes no desenvolvimento, como descrito pela (WIKIPEDIA, 2019) "um iterador se refere tanto ao objeto que permite ao programador percorrer uma coleção de elementos quanto ao objeto acessar os elementos de uma coleção", este padrão foi bastante utilizado na manipulação das respostas dos questionários aplicados.

O padrão *mediator* foi utilizado durante as transformações dos objetos da tabela *MobileResponse* para a tabela *EvaluationAnswer*, onde métodos do componente service intermediavam as manipulações no componente controller, o padrão mediator, como descrito pela (WIKIPEDIA, 2019), "é um padrão de projeto usado frequentemente quando deseja-se encapsular como os objetos interagem, ou seja, a comunicação entre os objetos é estabelecida através do Mediator."

O padrão *adpater* foi utilizado ao modificar as respostas da tabela *MobileResponse* para a tabela *EvaluationAnswer* do Mongo, o padrão adpater converte a interface de uma classe para outra interface que o cliente espera encontrar, "traduzindo" solicitações

do formato requerido pelo usuário para o formato compatível com o a classe adaptada e as redirecionando. Assim, o Adaptador concede que classes com que interfaces incompatíveis trabalhem juntas.

O Prototype como descrito pela ([WIKIPEDIA, 2019](#)) "define-se por um padrão criacional que permite a geração de novos objetos a partir de um modelo original ou protótipo que é clonado.", este foi utilizado na criação dos modelos de objetos para a criação dos questionários, cada tabela possui um modelo de dados e por sua vez foi criado um objeto referenciado.

5 Conclusão

O desenvolvimento deste projeto possibilitou um grande avanço na aplicação de questionários sobre a moradia e habitação pois a construção de uma aplicação que pode se integrar com diversos aplicativos simultaneamente e utilizando exigências mínimas de recursos de conexão permitirá um alcance muito maior tanto em extensão territorial quanto em volume de pesquisas e estudos sobre as condições de moradores em Uberlândia, podendo chegar a diversos lugares do país.

Além disso, com o desenvolvimento pensado na generalização de construção de questionários, será possível utilizar a aplicação não só para questionários desenvolvidos para pesquisas de habitação, como para questionários de diversos segmentos e áreas de pesquisas das universidades Federais do Brasil.

Dada a importância deste assunto, torna-se necessário a continuação do desenvolvimento da APO Digital, visando sempre melhorar e agilizar os procedimentos de aplicações de questionários visto que estes estudos trazem à tona as condições de moradias de diversos lugares e traçam a realidade destes permitindo o compartilhamento desses dados com as principais fontes do desenvolvimento urbano e social no Brasil.

As perspectivas de continuação deste trabalho são enormes e indicam que cada vez mais precisamos de uma integração entre profissionais de TI e os profissionais de arquitetura pois esta junção gera benefícios enormes para os campos de estudos abrangidos pelas duas áreas.

Para trabalhos futuros do APO Digital é extremamente importante pensar em novas formas de integração com tecnologias inovadoras, assim como nesta parte foi implementada a comunicação entre aplicação *back-end* e aplicativos móveis, novas tecnologias e ferramentas surgirão e é sempre importante utilizá-las da melhor forma possível para que cada vez mais se consiga um avanço na estruturação do planejamento urbano e habitacional.

Referências

- AUSTIN, D. et al. Web services architecture requirements. w3c working draft. *WWW Consortium*, v. 14, 2002. Citado na página 10.
- BECKER, F. Post-occupancy evaluation: Research paradigm or diagnostic tool. In: *Building evaluation*. [S.l.]: Springer, 1989. p. 127–134. Citado na página 6.
- COATES, P.; ARAYICI, Y.; OZTURK, Z. New concepts of post occupancy evaluation (POE) utilizing BIM benchmarking techniques and sensing devices. In: *Sustainability in Energy and Buildings*. [S.l.]: Springer, 2012. p. 319–329. Citado na página 6.
- COUNCIL, N.; ENVIRONMENT, B.; COUNCIL, F. *Learning from Our Buildings: A State-of-the-Practice Summary of Post-Occupancy Evaluation*. National Academies Press, 2002. (Compass series). ISBN 9780309076111. Disponível em: <<https://books.google.com.br/books?id=skhwiYaAw4kC>>. Citado na página 6.
- CRUZ, S. M. S. da. *Serviços Web – Uma breve introdução (Parte I)*. 2019. Disponível em: <<http://www.nce.ufrj.br/conceito/artigos/2005/01p1-1.htm>>. Acesso em: Dezembro 2019. Citado na página 10.
- GUTIERREZ, F. Spring boot, simplifying everything. In: *Introducing Spring Framework*. New York: Springer, 2014. p. 263–276. Citado na página 17.
- HADJRI, K.; CROZIER, C. Post-occupancy evaluation: purpose, benefits and barriers. *Facilities*, v. 27, n. 1/2, p. 21–33, 2009. Disponível em: <<http://dx.doi.org/10.1108/02632770910923063>>. Citado na página 6.
- IERUSALIMSKY, R. *Programming in Lua, Second Edition*. Rio de Janeiro: Lua.Org, 2006. ISBN 8590379825. Citado na página 7.
- IERUSALIMSKY, R. et al. Lua: uma linguagem para customização de aplicações. In: *VII Simpósio Brasileiro de Engenharia de Software - Caderno de Ferramentas*. Rio de Janeiro: SBC, 1993. p. 55. Citado na página 15.
- JAVA.COM. *Obtenha Informações sobre a Tecnologia Java*. 2019. Disponível em: <https://www.java.com/pt_BR/about/>. Acesso em: 10 dez. 2019. Citado na página 14.
- JETBRAINS, I. *IntelliJ Platform*. 2019. Disponível em: <<https://www.jetbrains.com/opensource/idea/>>. Acesso em: 10 dez. 2019. Citado na página 13.
- MAVEN.ORG. *Welcome to Apache Maven*. 2019. Disponível em: <<https://maven.apache.org/>>. Acesso em: 10 dez. 2019. Citado na página 12.
- MEDIA, A. D. *O que é NoSQL? - Java Magazine 87*. 2011. Disponível em: <<https://www.devmedia.com.br/o-que-e-nosql-java-magazine-87/19015>>. Acesso em: 10 dez. 2019. Citado na página 11.

PREISER, W.; WHITE, E.; RABINOWITZ, H. *Post-Occupancy Evaluation (Routledge Revivals)*. Routledge, 2016. (Routledge Revivals). ISBN 9781138888326. Disponível em: <<https://books.google.com.br/books?id=mn9NvgAACAAJ>>. Citado na página 6.

ROMERO, M. A.; ORNSTEIN, S. W. Avaliação pós-ocupação: métodos e técnicas aplicados à habitação social. In: *Avaliação pós-ocupação: métodos e técnicas aplicados à habitação social*. Porto Alegre: ANTAC, 2003. Citado na página 6.

SPRINGIO, S. P. *Spring Boot by Pivotal*. 2019. A. Disponível em: <<https://spring.io/projects/spring-boot>>. Acesso em: 10 dez. 2019. Citado na página 11.

VILLA, S. B. et al. *Desenvolvimento de Metodologia de Avaliação Pós-Ocupação em Unidades Habitacionais na Cidade de Uberlândia, Pertencentes à Primeira Fase do MCMV, Enfocando Aspectos Funcionais, Comportamentais e Ambientais*. Rio de Janeiro, 2015. Disponível em: <http://repositorio.ipea.gov.br/bitstream/11058/7477/1/RP_Desenvolvimento_2015.pdf>. Citado na página 21.

WIKIPEDIA. *Padrão de projeto de software*. 2019. A. Disponível em: <<https://pt.wikipedia.org/wiki/>>. Acesso em: 10 dez. 2019. Citado 2 vezes nas páginas 35 e 36.

ZAMMETTI, F. *Learn Corona SDK game development*. [S.l.]: Apress, 2013. Citado 2 vezes nas páginas 7 e 15.

ANEXO A – Questionário Como Você Mora?

Este anexo apresenta o questionário completo da APO **Como Você Mora?** que pretende a aferição da qualidade em habitação, considerando casas e apartamentos. A avaliação envolve tanto a qualidade técnica da moradia quanto a qualidade subjetiva que, em termos mais amplos, considera o atendimento à satisfação das necessidades sociais, do bem estar e da qualidade de vida do ser humano em relação à habitação.

São consideradas as perspectivas¹:

- (i) arquitetônica e urbanística: dimensões espaço-funcional, sociocultural e estética (adequação da imagem e volumetria da habitação aos significados que, numa determinada cultura, lhe são habitualmente associados);
- (ii) ambiental – adequação em termos de conforto ambiental (visual, acústico, higrotérmico, lumínico);
- (iii) construtiva – coerência das soluções construtivas (qualidade dos materiais e acabamentos, dos elementos e processos de construção);
- (iv) social – adequação às necessidades, aspirações, preferências, percepções, valores e sentimentos dos moradores; e
- (v) do processo – adequação do modo como os moradores adquirem as habitações (participação na fase de concepção, assistência por alguma instituição de apoio durante as diversas fases do processo).

O questionário base para o **Como Você Mora?** foi desenvolvido por alunos do curso de Bacharelado em Arquitetura e Urbanismo da Universidade Federal de Uberlândia sob a orientação dos professores: Simone Barbosa Villa, Fernando Garrefa, Rita de Cássia Pereira Saramago e Viviane dos Guimarães Alvim Nunes. São eles (em ordem alfabética):

¹ PEDRO, J. A. C. B. O. Definição e avaliação da qualidade arquitetônica habitacional. In: NUTAU, 1, 2002, São Paulo. Anais... São Paulo: NUTAU, 2002, p.95-111.

Ana Carolina Carleto Silva
Daniel Rodrigues Pascoal
Gabriela Barros Portilho
Lorena Spirandeli Marques
Luanny da Silva Souza
Paloma Ribeiro de Souza
Pedro Henrique Viana Silva
Thais Castro Alves de Ávila
Thiago Rodrigues

As Figuras [17](#), [18](#), [19](#), [20](#) incorporam neste documento o referido questionário.

Universidade Federal de Uberlândia - Faculdade de Arquitetura e Urbanismo e Design
[MORA] PESQUISA EM HABITAÇÃO

QUESTIONÁRIO - APO DIGITAL - GAME COMO VOCÊ MORA

TERMO DE COMPROMISSO

Você aceita responder esse questionário? Antes de continuar, por favor, leia e reconheça os termos seguintes: "Eu entendo que a minha participação nesta pesquisa é voluntária. Também entendo que todas as informações serão tratadas com a máxima confidencialidade, que o meu anonimato será respeitado em todos os momentos e que todas as informações coletadas serão utilizadas exclusivamente para fins acadêmicos científicos".

ACEITO () NÃO ACEITO ()

Identificação para organização da pesquisa (número do apartamento): _____ Bloco: _____ Data / / Horário: _____

1. SOBRE VOCÊ/MORADORES, FUNCIONÁRIOS E RENDA

- a. Qual é o seu gênero? Feminino () Masculino ()
b. Quantos anos você tem? () menos de 15 () de 15 a 20 () de 21 a 30 () de 31 a 40 () de 41 a 50 () de 51 a 60 () de 61 a 70 () mais de 70
c. Até que ano da escola você cursou?

() nunca fui à escola	() 4º ano fundamental incompleto	() 7º ano fundamental completo	() 2º ano médio incompleto
() 1º ano fundamental incompleto	() 4º ano fundamental completo	() 8º ano fundamental incompleto	() 2º ano médio completo
() 1º ano fundamental completo	() 5º ano fundamental incompleto	() 8º ano fundamental completo	() 3º ano médio incompleto
() 2º ano fundamental incompleto	() 5º ano fundamental completo	() 9º ano fundamental incompleto	() 3º ano médio completo
() 2º ano fundamental completo	() 6º ano fundamental incompleto	() 9º ano fundamental completo	() ensino superior incompleto
() 3º ano fundamental incompleto	() 6º ano fundamental completo	() 1º ano médio incompleto	() ensino superior completo
() 3º ano fundamental completo	() 7º ano fundamental incompleto	() 1º ano médio completo	

- d. Agora vamos falar sobre os moradores. Selecione quem mora com você e clique em avançar. Se você mora só, apenas clique em avançar.

() Esposa/Parceira	() Filha	() Mãe	() Avô	() Irmão / cunhado	() Animal de estimação	() Outro parente
() Marido /Parceiro	() Filho	() Pai	() Avó	() Irmã / cunhada	() Amigo (a)/ Colega	

- e. Trabalha algum funcionário em sua moradia? () Sim () Não.

- f. Algum funcionário mora em sua moradia? () Sim, regularmente () Sim, eventualmente () Não

- g. Qual sua renda familiar em reais? Lembre-se que é a renda de todos os moradores somada (opcional)

Até R\$ 500	Até R\$ 2.000	Até R\$ 3.500	Até R\$ 5.000	Até R\$ 6.500	Até R\$ 8.000	Até R\$ 9.500
Até R\$ 1.000	Até R\$ 2.500	Até R\$ 4.000	Até R\$ 5.500	Até R\$ 7.000	Até R\$ 8.500	Até R\$ 10.000
Até R\$ 1.500	Até R\$ 3.000	Até R\$ 4.500	Até R\$ 6.000	Até R\$ 7.500	Até R\$ 9.000	Mais de R\$ 10.000

- h. Qual é a sua atividade principal? _____

2. MORADIA ANTERIOR

- a. Você sempre morou no mesmo local ou já morou em outro local antes? () já morei em outro local () sempre morei no mesmo local

- a.1. Você residia em condomínio? () Sim () Não

- b. Qual era o tipo da sua moradia anterior? () Casa () Apartamento

- b.1. Que tipo de apartamento era? () Convencional () Quinete () Loft () Duplex () Triplex () Cobertura

- b.2. Que tipo de casa era? () Térrea () Sobrado () Geminada () Casa de Vila () Casa/cômodo de fundo

- c. Qual era o estado de aquisição? () Alugado () Empréstado () Financiado () Próprio

- d. Qual era seu grau de satisfação geral em relação a esta moradia anterior? () Muito bom () Bom () Regular () Ruim () Muito ruim

- e. Comparando com sua moradia atual, qual era o tamanho? () Muito bom () Bom () Regular () Ruim () Muito ruim

- f. Como era o padrão de acabamento? () Muito bom () Bom () Regular () Ruim () Muito ruim

- g. Como era a localização? () Muito bom () Bom () Regular () Ruim () Muito ruim

- h. E o gasto com a moradia por mês? () Muito bom () Bom () Regular () Ruim () Muito ruim

- i. Qual o tempo de permanência na(s) moradia (s) anterior (es)? () 0 a 1 ano() 1 a 2 anos() 2 a 3 anos() 3 a 4 anos() 4 a 5 anos() 5 anos ou mais

3. ENTORNO (definindo moradia atual)

- a. Muito bem. Agora vamos falar sobre sua moradia atual. Em qual estado você mora?

() Acre	() Distrito Federal	() Minas Gerais	() Rio de Janeiro	() São Paulo
() Alagoas	() Espírito Santo	() Pará	() Rio Grande do Norte	() Sergipe
() Amapá	() Goiás	() Paraíba	() Rio Grande do Sul	() Tocantins
() Amazonas	() Maranhão	() Paraná	() Rondônia	
() Bahia	() Mato Grosso	() Pernambuco	() Roraima	
() Ceará	() Mato Grosso do Sul	() Piauí	() Santa Catarina	

Agora digite:

b. cidade? _____

c. bairro. _____

d. seu CEP: _____

e. seu apartamento/bloco: _____

f. Desde quando mora ? _____

- g. Você se identifica com seu bairro? () Plenamente () Muito () Médio () Pouco () Ligeiramente

- h. Indique aqui quais dos equipamentos urbanos abaixo estão presentes no seu bairro:

() Posto de saúde	() Escola	Equipamento cultural (teatro, biblioteca, cinema, etc.)
() Lazer (clube, praça, parque, playground, bar, restaurante, cinema, etc.)	() Equipamentos para esporte (clube, ginásio poliesportivo, quadra, ciclovia, pista de skate, etc)	() Segurança (posto policial)

- i. Em relação à presença de comércio, seu bairro é: () Muito bem servido () Bem servido () Regular () Mal servido () Muito mal servido

- j. De qual tipo de estabelecimento comercial/serviço você sente falta no seu bairro? _____

- k. Indique quais dos espaços públicos abaixo estão presentes no seu bairro:

() Praça	() Parque	() Academia ao ar livre	() Espaço esportivo (campo de futebol, quadra, etc)
-----------	------------	--------------------------	--

- j.1. Qual sua satisfação dos espaços públicos em relação ao acesso () Muito bom () Bom () Regular () Ruim () Muito ruim

- j.2. A quantidade dos equipamentos: () Muito bom () Bom () Regular () Ruim () Muito ruim

- j.3. A qualidade dos equipamentos: () Muito bom () Bom () Regular () Ruim () Muito ruim

- j.4. A segurança: () Muito bom () Bom () Regular () Ruim () Muito ruim

- k. Em relação ao seu bairro, como é a agradabilidade? () Muito bom () Bom () Regular () Ruim () Muito ruim

- l. E as relações de convívio? () Muito bom () Bom () Regular () Ruim () Muito ruim

- m. Como é a acessibilidade no seu bairro (em relação às calçadas)? () Muito bom () Bom () Regular () Ruim () Muito ruim

- n. Existe alguma organização de moradores (comunitária) em seu bairro? () Sim () Não () Não Sei

- o. Indique qual: _____

- o. Você se sente integrado à cidade onde mora? () Sim () Não

4. CONJUNTO

- a. A sua moradia em que você reside pertence a um condomínio? () Sim () Não

SE SIM

Figura 17 – Questionário Casas e Apartamentos - Folha 1

Universidade Federal de Uberlândia - Faculdade de Arquitetura e Urbanismo e Design
[MORA] PESQUISA EM HABITAÇÃO

b. Quantos blocos existem no seu condomínio? () 1 () 2 () 3 () 4 () 5 () 6 () 7 () 8 () 9 () 10 () 11 () 12 ou mais

c. Selecione os equipamentos de uso coletivo que existem no seu condomínio ou edifício e indique sua quantidade.

() Piscina adulto	() Playground/ Brinquedoteca	() Academia	() Quadra	() Espaço gourmet	() Salão de festas
() Área para churrasco	() Piscina infantil	() Sauna	() Salão de jogos	() Quiosques	() Outro

d. Que tipologia de habitação você reside? () Casa () Apartamento

SE A PERGUNTA a* FOR NÃO

d. Que tipologia de habitação você reside? () Casa () Apartamento

5. AVALIAR MORADIA / Edifício

a. Você escolheu *viver nessa moradia* ou não teve escolha? () Escolhi () Não tive escolha

a.1. Diga-me, por que você escolheu *viver nessa moradia*? Escolha por ordem de prioridade.

() Segurança	() Qualidade	() Status social	() Localização	() Vista privilegiada	() Investimento	() Outro motivo
---------------	---------------	-------------------	-----------------	------------------------	------------------	------------------

b. Você ou sua família contribuíram no processo de projeto *da moradia*? () Sim () Não

c.1. Agora me diga, quais os pontos negativos de morar em apartamentos? Escolha por ordem de prioridade

() Falta de privacidade	() Espaço reduzido	() Limitação à reformas	() Outro motivo
() Vida social controlada	() Convívio social coletivo	() Ausência de jardim	

c.2. Agora me diga, quais os pontos negativos de morar em casa?

() Falta de segurança em relação à rua	() Espaço reduzido	() Limitação à reformas (sistema construtivo)	() Outro motivo
() Convivência com vizinhos	() Falta de privacidade em relação aos vizinhos	() Vida social controlada pelas regras do condomínio	

d. Quantos andares tem em média cada edifício? () 1 () 2 () 3 () 4 () 5 () 6 () 7 () 8 () 9 () 10 () 11 () 12 ou mais

e. Qual o número de apartamentos por andar do seu edifício? () 1 () 2 () 3 () 4 () 5 () 6 () 7 () 8 () 9 () 10 () 11 () 12 ou mais

f. Agora classifique os seguintes aspectos da *sua moradia*, de acordo com seu nível de satisfação:

	Muito ruim	Ruim	Regular	Bom	Muito bom(a)
Acessibilidade no edifício					
Qualidade da construção					
Aparência do edifício					
Limpeza e conservação dos espaços					
Relação custo-benefício					
Segurança em relação à rua					
Privacidade em relação à rua					
Nível de convivência entre os vizinhos					

g. (para apartamentos) Agora diga-me seu nível de satisfação em relação a:

	Muito ruim	Ruim	Regular	Bom	Muito bom	Não há
Vagas de estacionamento						
Hall de entrada do edifício						
Elevadores						
Escadas						
Hall dos apartamentos						

h. Existem áreas verdes em sua moradia? () Sim () Não

i. Você sente falta dessas áreas? () Sim () Não

j. Existe alguma delimitação/barreira entre a sua moradia e a vizinhança? () Sim () Não

k.1. Indique qual: () Muro () Grades () Vegetação

k.2. Você se sente bem em sua moradia? () Muito bem () Bem () Regular () Ruim () Muito ruim

l. Qual a sua satisfação em relação ao tamanho do seu lote? () Muito ruim () Ruim () Regular () Bom () Muito bom

l.1. Qual a sua satisfação em relação à arborização? () Muito ruim () Ruim () Regular () Bom () Muito bom

l.2. Qual a sua satisfação em relação à segurança? () Muito ruim () Ruim () Regular () Bom () Muito bom

l.3. Qual a sua satisfação em relação ao o fluxo de veículos? () Muito ruim () Ruim () Regular () Bom () Muito bom

6. AVALIAR UNIDADE

a. Qual é o estado de aquisição *da sua moradia*? () Alugado () Empréstado () Financiado () Próprio

b.1. Qual o tipo do seu apartamento? () Convencional () Quinete () Loft () Duplex () Triplex () Cobertura

b.2. Qual o tipo da sua casa? () Casa térrea () Casa assobradada () Casa geminada () Casa de vila () Casa/cômodo e fundo

c. O que a sua moradia é para você? Escolha por ordem de importância.

() Um local onde estão as pessoas que me interessam	() Um investimento material	() Um local onde apenas durmo
() Um local no qual consigo realizar minhas atividades	() Um local onde estão meus pertences	() Um local com o qual me identifico
() Um local onde passo a maior parte do tempo	() Um local seguro	() Outro

d. Você se adaptou à sua moradia e se identifica com ela? () Plenamente () Muito () Médio () Pouco () Ligeiramente

e. Na sua atual moradia, foi possível acomodar os seus antigos móveis e eletrodomésticos? () Sim () Não

e.1. SE SIM

Móveis grandes	Eletrodomésticos pequenos	() Móveis novos
Móveis pequenos	Móveis adequados	() Eletrodomésticos novos
Eletrodomésticos grandes	Eletrodomésticos adequados	

e.2 SE NÃO

Eletrodomésticos grandes	Móveis pequenos	Móveis novos
Eletrodomésticos pequenos	Portas muito estreitas	Excesso de eletrodomésticos
Móveis grandes	Posição ruim das portas e janelas	

f. Você utiliza a sua moradia para adquirir alguma renda extra? () Sim () Não

f.1. Que atividade você realiza?

() Vendas de produtos em catálogos	() Cuidas de pessoas extra familiares	() Lavar e passar roupas para fora
() Vendas de produtos	() Cozinhar para fora	() Trabalhos manuais (costura, artesanato)
() Trabalhos via internet	() Outro	

f.2. Quantos moradores utilizam a moradia como fonte de renda? () 1 () 2 () 3 ou mais

f.3. Em casos de utilização da moradia como fonte de renda, o espaço é utilizado: () Apenas para produção () Local de venda/ Recepção de clientes () Ambos

g. Agora avalie a sua moradia. Para cada característica, diga-me seu nível de satisfação.

	Muito ruim	Ruim	Regular	Bom	Muito bom
Tamanho					
Divisão dos cômodos					

Figura 18 – Questionário Casas e Apartamentos - Folha 2

Universidade Federal de Uberlândia - Faculdade de Arquitetura e Urbanismo e Design
[MORA] PESQUISA EM HABITAÇÃO

Qualidade dos materiais de construção					
Facilidade de manutenção e limpeza					
Adaptação às necessidades da família					
Privacidade de todos os membros da casa					

h. A tecnologia, em especial, a internet, influencia nas relações pessoais dentro da moradia? () Plenamente () Muito () Médio () Pouco () Ligeiramente

i. Qual a área útil aproximada da sua moradia?

() 30	() 35	() 40	() 45	() 50	() 55	() 60	() 65	() 70	() 75	() 80	() 85	() 90	() 95
() 100	() 105	() 110	() 115	() 120	() maior que 120m2								

j. Muito bem. Agora selecione os cômodos ou espaços que existem em sua moradia e sua quantidade. Se não houver nenhum dos que eu mostrar, é só avançar.

Cômodo	Nº	Tipo
() Varanda		() Aberta () Fechada
() Cozinha		() Simples () Integrada à área de serviço () Integrada à sala de jantar () Integrada às salas de estar e jantar
() Banheiro		() Banheiro social () Banheiro de serviço () Lavabo
() Área de serviço		() Simples () Integrada à cozinha
() Escritório		() Simples () Integrado à sala
() Sala de jantar		() Simples () Integrada à sala () Integrada à cozinha () Integrada à sala e à cozinha
() Dormitório casal		() Simples () Simples com closet () Suíte () Suíte com closet
() Sala de estar		() Simples () Integrada à sala de jantar () Integrada à cozinha () Integrada à sala e cozinha
() Dormitório solteiro		() Simples () Simples com closet () Suíte () Suíte com closet

k. Agora avalie os cômodos da sua moradia. Para cada característica, diga-me seu nível de satisfação.

COZINHA	Muito ruim	Ruim	Regular	Bom	Muito bom
Tamanho					
Facilidade de mobiliar					
Ventilação natural					
Iluminação natural					
Temperatura					
Nível de ruído					

l. Quais ALTERAÇÕES OU REFORMAS foram feitas na cozinha? Se não houver nenhuma, apenas avança.

() Troca de acabamento	() Colocação de gesso	() Colocação de armário	() Remoção ou acréscimo de parede
() Troca de portas ou janelas	() Pintura	() Nenhuma	

m.1. Com relação à decoração, a sua moradia permite fazer alterações para atender melhor sua necessidade ou o seu gosto? () Sim () Não

m.1.1. Se sim, que tipo de alteração? Para cada um, diga-me o nível de dificuldade:

	Muito fácil	Fácil	Regular	Difícil	Muito difícil
Variar posição dos móveis					
Variar posição dos eletrodomésticos					
Inserir flores e plantas naturais					
Inserir objetos pessoais decorativos ou de recordações					
Pintar e/ou usar papel de parede					
Abrir vãos e/ou demurrar paredes					
Colocar forro de gesso					
Acrescentar tomadas e/ou iluminação					

m.1.2. Se não, por quê?

() Limitações estruturais	() Disposição ruim de portas	() Poucas paredes livres	() Limitações financeiras
() Limitações de área	() Disposição ruim de janelas	() Ausência de sacada	() outros

m.2. Você já realizou alguma das alterações listadas na questão anterior? () Sim () Não

m.2.1. Se sim, quais? Para cada característica, diga-me seu nível de dificuldade:

Variar posição dos móveis	Variar posição dos eletrodomésticos	Inserir flores e plantas naturais	Inserir objetos pessoais decorativos ou de recordações
Pintar e/ou usar papel de parede	Colocar forro de gesso	Acrescentar tomadas e/ou iluminação	

l.2.2. Se não, por quê?

Limitações financeiras	Falta de assistência profissional	Não tenho esses objetos
Já estou satisfeita (o)	Não gosto desses objetos	Vou me mudar em breve

n. Os cômodos da sua moradia conseguem se alterar para usos diferentes?

() Muito ruim () Ruim () Regular () Boa () Muito boa

o. Em que período do dia seu apartamento recebe mais sol? () Manhã () Tarde

o.1. Qual cômodo da sua moradia recebe mais sol durante a manhã? () Quarto () Sala de estar () Sala de jantar () Cozinha () Banheiro () Lavanderia

o.2. Qual cômodo da sua moradia recebe mais sol durante a tarde? () Quarto () Sala de estar () Sala de jantar () Cozinha () Banheiro () Lavanderia

p. Além das atividades convencionais, o que você faz nos seguintes cômodos?

SALA DE ESTAR	SALA DE JANTAR	VARANDA	COZINHA
() Almoçar	() Cochilar	() Estender a roupa	() Estender a roupa
() Jantar	() Exercitar-se	() Passar roupa	() Passar roupa
() Exercitar-se	() Tocar instrumentos	() Exercitar-se	() Exercitar-se
() Tocar instrumento	() Jogar	() Dançar	() Tocar instrumento
() Trabalhar	() Dançar	() Almoçar	() Dançar
() Dançar	() Passar roupa	() Jantar	() Assistir TV
() Passar roupa	() Armazenar coisas	() Refeição rápida	() Usar o computador/celular
() Armazenar coisas	() Dormir	() Ler/Estudar	() Fazer trabalhos manuais
() Dormir	() Fazer trabalhos manuais	() Tocar instrumento	() Trabalhar/estudar
() Fazer trabalhos manuais	() Trabalhar/estudar	() Receber convidados	() Receber convidados
() Ler/estudar	() Usar computador/celular	() Armazenar coisas	() Armazenar coisas
() Outros	() Outros	() Outros	() Outros

ÁREA DE SERVIÇO	DORMITÓRIO CASAL	DORMITÓRIO SOLTEIRO	BANHEIRO
() Exercitar-se	() Refeição rápida	() Refeição rápida	() Exercitar-se
() Tocar instrumento	() Exercitar-se	() Exercitar-se	() Tocar instrumento
() Assistir TV	() Tocar instrumento	() Tocar instrumento	() Jogar
() Usar computador/celular	() Jogar	() Jogar	() Dançar
() Fazer trabalhos manuais	() Dançar	() Dançar	() Lavar roupa
() Trabalhar/estudar	() Passar roupa	() Passar roupa	() Passar roupa
() Navegar na internet	() Armazenar coisas	() Armazenar coisas	() Armazenar coisas
() Armazenar coisas	() Usar computador/celular	() Usar computador/celular	() Cochilar

Figura 19 – Questionário Casas e Apartamentos - Folha 3

Universidade Federal de Uberlândia - Faculdade de Arquitetura e Urbanismo e Design
[MORA] PESQUISA EM HABITAÇÃO

☐ Refeição rápida	☐ Fazer trabalhos manuais	☐ Fazer trabalhos manuais	☐ Fazer trabalhos manuais
☐ Lavar louça	☐ Trabalhar	☐ Trabalhar	☐ Ler/estudar
☐ Dormir	☐ Ler/estudar	☐ Ler/estudar	☐ Usar computador/celular
☐ Outros	☐ Outros	☐ Outros	☐ Outros

7. HÁBITOS SUSTENTÁVEIS

a. Você economiza água? ☐ Sim ☐ Não

Se sim: ☐ Para reduzir o preço das contas ☐ Por medo de faltar água em períodos de seca ☐ Para causar menos problemas ao meio ambiente ☐ Outros

Se não: ☐ Não vejo necessidade ☐ Porque o custo é baixo ☐ Pois acho que é um recurso abundante na natureza ☐ Outros

b. O que você faz para economizar água? (Se necessário, marque mais de uma opção)

☐ Escovo os dentes com a torneira fechada	☐ Utilizo a máquina de lavar com a capacidade máxima
☐ Ensabo o louça com a torneira fechada	☐ Reutilizo a água da máquina de lavar para limpeza do apartamento
☐ Tomo banhos rápidos (5 a 10 minutos)	☐ Posso aparelhos que economizam água (como descarga de duplo acionamento, por exemplo)
☐ Outros	☐ Outros

b.1. (Se marcar que possui aparelhos que economizam água) Quais tipos de equipamentos economizadores de água você possui?

☐ Máquina de lavar com reuso	☐ Descarga com duplo acionamento	☐ Torneira econômica
☐ Caixa d'água para reuso	☐ Vaso com caixa acoplada	☐ Outros

b.2. (Se não marcar que possui aparelhos que economizam água) Por qual motivo você não faz uso de equipamentos economizadores de água?

☐ Desconheço	☐ Instalação depende de reformas	☐ Não vejo necessidade
☐ Custo alto	☐ Difícil manutenção	

c. Você economiza energia elétrica? ☐ Sim ☐ Não

Se sim: ☐ Para reduzir o preço das contas ☐ Por medo de faltar energia em períodos de seca ☐ Para causar menos problemas ao meio ambiente ☐ Outros

Se não: ☐ Não vejo necessidade ☐ Porque o custo é baixo ☐ Pois acho que é um recurso abundante na natureza ☐ Outros

d. O que você faz para economizar energia elétrica? Pode marcar mais de uma opção.

☐ Apago as luzes ao sair do ambiente	☐ Compra eletrodomésticos com selo A de economia de energia	☐ Outros
☐ Desligo os aparelhos que não estão em uso	☐ Utilizo lâmpadas econômicas (como fluorescentes e LED)	

e. Você sabe qual é a diferença entre as bandeiras tarifárias na conta de energia elétrica? ☐ Sim ☐ Já ouvi falar, mas não sei o significado

☐ Desconheço

f.1. Na sua habitação, há alguma equipamento que faz proveito da energia solar? ☐ Placas solares ☐ Painéis fotovoltaicos ☐ Ambos os sistemas ☐ Nenhum dos dois sistemas

f.1.1. Se não ou apenas um dos sistemas: Por qual motivo você não faz uso desse tipo de equipamento (painéis e placas)?

☐ Custo alto	☐ Não vejo vantagem nesse investimento	☐ Desconheço
☐ Instalação depende de reformas	☐ Difícil manutenção	☐ Outros

g. Você separa o lixo entre reciclável e comum? ☐ Sim ☐ Não

Se sim: ☐ Para causar menos problemas ao meio ambiente ☐ Pois há coleta seletiva no meu bairro ☐ Para doar aos catadores de lixo ☐ Outros

Se não: ☐ Pois não há coleta seletiva no meu bairro ☐ Não vejo necessidade ☐ Pois acho difícil separar no dia-a-dia ☐ Pois não sei que tipo de lixo é reciclável ☐ Outros

h. Você tem conhecimento sobre alguma cooperativa de reciclagem em sua cidade? ☐ Sim ☐ Não

h.1. Você sabe para onde o lixo reciclável da sua cidade vai? ☐ Sim ☐ Não

h.2. Você conhece algum Ecoponto próximo da sua residência? ☐ Sim ☐ Não

i. Você separa o óleo dos outros tipos de resíduos? ☐ Sim ☐ Não

Se sim: ☐ Para causar menos problemas ao meio ambiente ☐ Pois reutilizo o óleo para outros fins ☐ Para levar até os locais que coletam óleo ☐ Outros

Se não: ☐ Não vejo necessidade ☐ Pois acho difícil separar no dia-a-dia ☐ Não sei onde posso levar o óleo separado ☐ Outros

j. O que você faz com resíduos de construção civil?

☐ Jogo em algum terreno vazio ☐ Encaminho para um ecoponto ☐ Jogo em algum aterro sanitário ☐ Não faço nada

k. Como você realiza o descarte de baterias e equipamentos eletrônicos?

☐ Junto com o lixo doméstico ☐ Separo, mas deixo na lixeira ☐ Deixo em algum ponto de coleta específico

l. Como você realiza o descarte de medicamentos vencidos?

☐ Junto com o lixo doméstico ☐ Separo, mas deixo na lixeira ☐ Deixo em algum ponto de coleta específico

m. Você consome alimentos, aqueles produzidos sem substâncias químicas (como agrotóxicos ou hormônios)? ☐ Sim ☐ Não ☐ Em parte

Se sim: ☐ Preocupação com a sua saúde ☐ Seu cultivo traz menores problemas ao meio ambiente ☐ São mais saborosos

Se não: ☐ Preços inacessíveis ☐ Dificuldade de encontrar no mercado ☐ Acredito que não contribui em nada utilizá-lo

Se em parte: ☐ Preocupação com a sua saúde ☐ Seu cultivo traz menores danos ao meio ambiente ☐ São mais saborosos ☐ Preços inacessíveis

☐ Dificuldade de encontrar no mercado

n. Quando você vai ao supermercado, como transporta as mercadorias compradas? ☐ Sacola reutilizável ☐ Caixa de papelão ☐ Sacola plástica ☐ Outros

o. Você possui plantas na sua moradia? ☐ Sim ☐ Não

Se sim: ☐ Como decoração, pois acho bonito ☐ Pois gosto de cuidar ☐ Para consumo próprio ☐ Para melhorar a qualidade do ar ☐ Outros

Se não: ☐ Não tenho tempo para cuidar ☐ Não sei como cuidar ☐ Não acho necessário ☐ Não tenho espaço ☐ Outros

p. Quais tipos de plantas você possui? ☐ Decorativa ☐ Hortaliça/ Comestível ☐ Medicinal

q. Qual meio de transporte você utiliza nos casos abaixo?

	A pé	Bicicleta	Carro ou moto	Transporte coletivo
No dia-a-dia, até cinco quarteirões				
No dia-a-dia, acima de cinco quarteirões				
No dia-a-dia, acima de cinco quarteirões				
Mais utilizado no geral				

r. Em relação ao meio de transporte que você mais utiliza, explique sua escolha indicando se necessário mais de uma resposta.

☐ Rapidez de locomoção	☐ Consigo ir a todos os lugares	☐ Divido com outras pessoas
☐ Custo é mais acessível	☐ Preocupação com sua idade	☐ Necessidade física
☐ Causa menos problemas ao meio ambiente	☐ Outros	

Figura 20 – Questionário Casas e Apartamentos - Folha 4