UNIVERSIDADE ESTADUAL DA PARAÍBA – UEPB
CAMPUS VII – GOVERNADOR ANTÔNIO MARIZ
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS – CCEA
CURSO DE BACHARELADO EM COMPUTAÇÃO

ERIC DA COSTA MUNIZ

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DE APOIO À AUTO-LOCALIZAÇÃO NO CAMPUS VII DA UEPB

PATOS – PB
2018

ERIC DA COSTA MUNIZ

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DE APOIO À AUTO-LOCALIZAÇÃO NO CAMPUS VII DA UEPB

Trabalho de Conclusão de Curso Bacharelado em Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de graduado em Computação.

Área de concentração: Desenvolvimento de Aplicativos Móveis.

Orientador: Prof.º Me. Jefferson Felipe Silva de Lima.

PATOS – PB

2018

M963d Muniz, Eric da Costa.

Desenvolvimento de um aplicativo móvel de apoio à auto-localização no Campus VII da UEPB [manuscrito] / Eric da Costa Muniz. - 2018.

94 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas , 2018.

"Orientação : Prof. Me. Jefferson Felipe Silva de Lima , Coordenação do Curso de Computação - CCEA."

1. QR Code. 2. easYProcess. 3. Desenvolvimento móvel Android. 4. Android OS. 5. Aplicativo móvel. I. Título

21. ed. CDD 005.12

Elaborada por Camile de A. Gomes - CRB - 15/559 BC/UEPB

Eric da Costa Muniz

Desenvolvimento de um aplicativo móvel de apoio à auto-localização no Campus VII da UEPB

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciências da Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Bacharel em Ciências da Computação.

Aprovado em 27/11/2018

BANCA EXAMINADORA

Msc. Jefferson Felipe Silva de Lima
(Orientador)

Esp. Fabio Junior Francisco da Silva
(Examinador)

Msc. Pablo Ribeiro Suarez
(Examinador)

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelos dons da vida, da inteligência e da sabedoria, também pela virtude da paciência e por todas as bênçãos que recebi ao longo desta caminhada.

A minha família, que sempre me impulsionou em cada momento, me dando coragem de prosseguir e acreditando em mim quando achei que não conseguiria.

Aos meus professores e professoras, desde o tempo da licenciatura, por compartilharem seu imenso saber. Em especial ao grande mestre Pablo Ribeiro Suárez, por todas as oportunidades e ensinamentos fornecidos.

Aos meus amigos, tanto os que vieram e ainda estão comigo quanto os que vieram e se foram, pelas experiências que ajudaram a construir meu caráter.

E em especial ao meu orientador e grande mestre Jefferson Felipe Silva de Lima, pela honestidade, paciência, dedicação e apoio infindáveis desde o começo, sem os quais este trabalho não aconteceria.

“Uma viagem de mil milhas começa com um único passo.”

(Lao-Tsé)

RESUMO

Saber localizar-se é uma das primeiras necessidades de qualquer pessoa quando se encontra em um ambiente desconhecido. Universidades também são passíveis deste tipo de empecilho, considerando que estão sempre abrindo as portas para novos membros. Como novos ingressos, que tem seu primeiro acesso ao Campus a cada semestre, existe a necessidade de uma certa familiarização com o ambiente, para poderem localizar suas salas de aula, laboratórios, coordenações, biblioteca, etc. Este trabalho apresenta o processo de desenvolvimento de uma aplicação móvel para o sistema operacional *Android OS* que integre a tecnologia de QR Code e sirva como ferramenta de apoio à auto localização e orientação no Campus VII – Patos da Universidade Estadual da Paraíba (UEPB). Para tanto foi utilizada como guia a metodologia ágil *easYProcess* (YP) para projetar e gerenciar o processo de criação do aplicativo, e tecnologias como a linguagem de programação Kotlin para a criação do código da aplicação. Como resultado, observou-se que a tecnologia do QR Code foi eficaz e eficiente no cumprimento dos requisitos da problemática e através dos diferentes tipos de testes realizados no aplicativo, comprovou-se seu bom desempenho e usabilidade.

Palavras-chave: desenvolvimento móvel *Android*; QR Code; metodologia ágil de desenvolvimento; *easYProcess.*

ABSTRACT

Knowing how to locate yourself is one of the first needs of anyone when you are in an unfamiliar environment. Universities are also liable to this kind of hindrance, considering that they are always opening doors for new members. As new entrants, who have their first access to the Campus every semester, there is a need for a certain familiarization with the environment, to be able to locate their classrooms, laboratories, coordinations, library, etc. This work presents the process of developing a mobile application for the Android OS operating system that integrates the QR Code technology and serves as a tool to support self - location and orientation in Campus VII - Patos of the State University of Paraíba (UEPB). For this, the easYProcess (YP) agile methodology was used as a guide to design and manage the process of creating the application, and technologies such as the Kotlin programming language for creating the application code. As a result, it was observed that the technology of QR Code was effective and efficient in the fulfillment of the requirements of the problem and through the different types of tests carried out in the application, it was verified its good performance and usability.

Keywords: Android mobile development; QR Code; agile development methodology; easYProcess.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo é apresentada uma visão geral do tema proposto, visando contextualizar o problema, objetivo e justificativa deste trabalho.

1.1 Cenário técnico científico

Saber localizar-se é uma das primeiras necessidades do ser humano quando se encontra em um ambiente desconhecido, seja o mesmo um espaço aberto ou fechado. Ao longo da história a humanidade inventou diversas formas de orientar-se em um espaço, passando desde a aplicação de técnicas tal como a orientação pelos astros, até a utilização de instrumentos como a bússola. Em dias atuais, é comum quando tal situação ocorre recorrermos ao auxílio da tecnologia para buscar referências para podermos obter alguma orientação (PENA, 2018).

Universidades também são passíveis de problemas quanto à localização e orientação, considerando que estão sempre abrindo as portas para novos membros (estudantes, professores e membros do setor administrativo). Como novos ingressos, que tem seu primeiro acesso ao Campus a cada semestre, existe a necessidade de uma certa familiarização com o ambiente, para poderem localizar suas salas de aula, laboratórios, coordenações e qualquer outro espaço que o ambiente da universidade venha a oferecer.

1.2 Problemática e proposta de solução

Durante o período acadêmico de 2016.2 no Campus VII - Patos da Universidade Estadual da Paraíba (UEPB), mais precisamente durante a escolha de proposta de projeto para as disciplinas de Laboratório de Engenharia de Software I e Paradigmas de Linguagens de Programação, notou-se que o problema de localização levava estudantes e professores (novatos ou não) a passarem vários minutos para encontrarem salas de aula e a biblioteca do Campus por falta de familiaridade ou alteração da disposição das turmas em cada sala do Campus.

Tal cenário é recorrente no Campus de Patos visto que o mesmo convive com problemas que o impulsionam. Sendo estes: estudantes matriculados efetivamente em um determinado período de seu curso são permitidos cursarem disciplinas de períodos subsequentes, às vezes até mesmo ignorando o sistema de créditos do curso; turmas de primeiro período por vezes são muito numerosas e precisam ser realocadas de uma sala de aula para outra devido à falta de

espaço para os estudantes; o próprio crescimento do Campus, visto que novas localidades são agregadas ao ambiente.

Este contexto levantou o seguinte questionamento: como prover apoio através da tecnologia para o problema recorrente de localização e orientação no Campus VII – Patos da UEPB?

Como proposta de solução para esta questão, foi realizado ainda na oportunidade do semestre 2016.2 dentro das disciplinas de Laboratório de Engenharia de Software I e Paradigmas de Linguagens de Programação o desenvolvimento de um aplicativo para smartphones que pudesse auxiliar os membros do Campus no problema de localização e orientação. Visto que o produto desenvolvido nesta oportunidade apresentou boa funcionalidade e agradou professores e membros da equipe que o desenvolveram, e que havia espaço para melhorias no mesmo, decidiu-se recriar o aplicativo dentro de um trabalho mais elaborado buscando melhorar características como desempenho e usabilidade.

1.3 Justificativa

A criação da versão anterior da aplicação foi realizado através do desenvolvimento móvel nativo para o sistema *Android*. Naquela oportunidade foram utilizadas as tecnologias padrão para tal desenvolvimento, sendo as mesmas: a linguagem XML para montar a interface gráfica do aplicativo, a linguagem de programação Java para criar sua lógica e o banco de dados SQLite para armazenamento de dados. Neste trabalho porém, o conjunto de tecnologias de desenvolvimento foi alterado para: interface gráfica ainda feita em XML, a linguagem de programação Kotlin para a parte lógica, e o banco de dados MongoDB para realizar o armazenamento de dados. Dados estes que serão acessados pelo aplicativo através de um serviço web REST.

Buscou-se integrar ao aplicativo a tecnologia de QR Code. Para tanto, seria implementado um scanner de QR Code que ao ler os códigos posicionado em uma sala do Campus VII, apresentasse ao usuário os horários de utilização daquela sala. O scanner poderia ainda, indicar através da leitura dos códigos, em qual direção se encontra uma sala que o usuário desejasse chegar.

Para auxiliar no processo de desenvolvimento de software fez-se o uso de uma metodologia ágil de desenvolvimento. Sendo escolhida a *easYProcess* (YP), uma metodologia

de engenharia de software para desenvolvimento de sistemas criado em 2003 na Universidade Federal de Campina Grande (UFCG).

1.4 Objetivos

Esta seção versa sobre a apresentação dos objetivos geral e específicos que compõem este trabalho.

1.4.1 Objetivo Geral

Desenvolver um aplicativo móvel que funcione como ferramenta de apoio à auto localização e orientação para estudantes e professores no Campus VII da UEPB.

1.4.2 Objetivos Específicos

- Mostrar a evolução das tecnologias para localização e orientação;
- Utilizar práticas de desenvolvimento ágil para projetar e implementar o sistema;
- Apresentar as tecnologias de desenvolvimento utilizadas;
- Utilização da tecnologia de QR Code para facilitar a auto localização/identificação de espaços do Campus VII.

1.5 Metodologia

Para o desenvolvimento deste trabalho foram realizadas as seguintes etapas:

- No **Levantamento de Material Bibliográfico** foram pesquisados e estudados artigos, livros e matérias em *websites* que abordassem os conteúdos relevantes para o trabalho com o intuito de obter mais conhecimento sobre o que seria feito no mesmo;
- Na **Definição de Metodologia de Desenvolvimento** foram analisadas as metodologias de desenvolvimento ágil e selecionada uma que melhor se adequasse ao trabalho;
- Na **Definição das Tecnologia de Desenvolvimento** foram reanalisadas as tecnologias utilizadas na versão anterior do aplicativo e outras opções mais atuais de tecnologias para realizar o desenvolvimento, sendo selecionadas as que melhor se adequaram à proposta do trabalho;

- Na **Revisão dos Requisitos da Aplicação** foram revisados os requisitos levantados para o desenvolvimento da versão anterior da aplicação, bem como os artefatos gerados durante o processo de projeto e desenvolvimento da versão em busca de como melhorar os pontos levantados no desenvolvimento que ocorreria neste trabalho;
- Na **Execução da YP** foram seguidas as etapas da metodologia para o projeto e implementação do novo aplicativo, sendo gerados todos os artefatos requeridos pela YP.
- Na **Apresentação dos Resultados** foram apresentados os resultados do processo de desenvolvimento através dos artefatos produzidos dentro da execução da YP, bem como resultados de diferentes tipos de testes realizados no aplicativo para comprovar que o mesmo possui bom funcionamento e cumpre com os requisitos levantados para atacar a problemática em questão.

1.6 Estrutura do trabalho

Este trabalho está organizado da seguinte forma: no Capítulo 1, apresenta uma visão geral desta investigação com relação a contextualização do problema, objetivos e metodologia; no Capítulo 2, são apresentados os temas relacionados à pesquisa; no Capítulo 3, a metodologia de produção do trabalho; no Capítulo 4, são apresentados os resultados do processo de produção do aplicativo, bem como resultados de testes realizados sobre o mesmo; no Capítulo 5, são apresentadas as conclusões; no Capítulo 6, as referências utilizadas no decorrer da pesquisa e por fim são apresentados os Apêndices do trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 Orientação e Localização

Saber localizar-se em um espaço é uma necessidade cotidiana do ser humano. Ter noção de onde se está é obrigatório para poder orientar-se e dirigir-se até o local que se deseja chegar. Sempre que é preciso ir a um determinado lugar pela primeira vez, recorre-se a referências como um endereço ou qual direção seguir. Esse processo cotidiano de locomoção é formado por dois pontos: localização - o ponto de partida e o destino; e orientação - como seguir de um ponto ao outro.

No decorrer da história foram criados diferentes instrumentos e formas de orientação, passando desde a orientação pelos astros até a utilização de instrumentos como a bússola, guiada pela rosa dos ventos e hoje temos diariamente o auxílio do popular *Global Positioning System* (GPS) guiado por satélites. Este último que nasceu como um instrumento separado e hoje é inserido por padrão em qualquer celular smartphone, o que torna seu uso ainda mais acessível. Vale enfatizar que todos os métodos e instrumentos de orientação citados ainda são utilizados hoje em dia ao redor do mundo, em diferentes contextos (PENA, 2018).

Quando se fala sobre localização e orientação, não se pode deixar de considerar a importância dos mapas. Estes se mostram úteis para mais do que indicar localidades, podendo ser considerados como ferramentas de comunicação devido sua capacidade de transmitir informações que ajudam a medir distâncias, fornecer conhecimento sobre um determinado espaço, e é claro seu impacto para localização e orientação.

Com as tecnologias atuais, a utilização de mapas pode fornecer uma gama ainda maior de informações. Os chamados totens interativos são utilizados em diversos estabelecimentos: museus, shoppings, hospitais e até universidades. A figura 1 abaixo ilustra um exemplo de totem interativo.

Figura 1 - Totem Interativo.

Fonte: Igor Pimentel, 2015.

Estas telas interativas permitem exibir diferentes tipos de conteúdo de imagem, sendo possível a exibição de mapas interativos. Graças a interatividade são criados mapas dos estabelecimentos que podem indicar rotas aos visitantes, como mostrado na Figura 2.

Figura 2 - Exibição de rota no mapa.

Fonte: Igor Pimentel, 2015.

É possível também consultar informações sobre a estrutura do local e o que ocorre nas localidades, como mostrado na Figura 3.

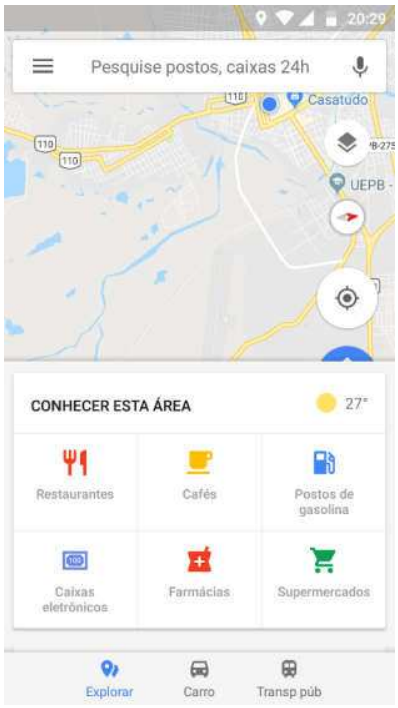
Figura 3 - Detalhes da localidade do estabelecimento.

Fonte: Igor Pimentel, Rio de Janeiro: Interface digital e design de totem interativo; 2015.

A praticidade dos smartphones continua a ser grande neste contexto. Com este dispositivo já mundialmente difundido que é vendido já com o GPS integrado, tornou-se possível a utilização de poderosas aplicações de localização e orientação como é o caso do aplicativo *Maps* da *Google*[1]. Essa *app* que além, de rastrear através do GPS do aparelho a posição onde o usuário está e traçar rotas para um destino para diferentes tipos de transporte (a pé, ônibus, automóvel, trem, etc.), apresenta informações sobre localidades e estabelecimentos, dando sugestões ao usuário de acordo com suas buscas. A figura 4 mostra a tela do aplicativo *Google Maps*.

[1] **Maps – Navegação e transporte público**. Google Play, 2018. Disponível em: <https://play.google.com/store/apps/details?id=com.google.android.apps.maps>. Acesso em: 23 mai. 2018.

Figura 4 - Tela do *Google Maps*.



Fonte: *Play Store, Google LLC*; 2018.

2.2 QR Code

O *QR*[2] *Code* ou código QR é uma forma de representar códigos de barras em formado bidimensional. Estes códigos são facilmente escaneados atualmente através de câmeras de *smartphones* e podem armazenar diversos tipos de informações (números, contatos telefônicos, links para sites, etc.) que ao serem lidos são transformados em texto. Este formato de código de barras foi lançado em 1994 pela empresa japonesa Denso Wave que embora possua os direitos de patente da tecnologia, escolheu mantê-la como uma licença livre para uso (PRASS, 2018).

Para Ferreira (2013) o QR é uma ferramenta poderosa para marketing de empresas devido sua simples usabilidade. Basta apenas apontar o leitor de QR para se obter a informação contida. Isso somado à capacidade de personalização permite que empresas utilizem esta tecnologia para chamar atenção de usuários e difundir com facilidade a sua marca.

A figura 5 mostra exemplos de QR Codes personalizados.

[2] *Quick Response* ou Resposta Rápida.

Figura 5 - QR Codes personalizados.

Fonte: Café com Galo; 2013.

Esta tecnologia recebe impulso também do mercado de marketing móvel, graças à em 2002, ter começado o surgimento de celulares com leitores de QR Code, o que auxiliou a expansão dos mesmos e aumentou também seu poder de comunicação. Com aparelhos com funcionalidades que permitem fornecer localização do usuário, as empresas podem utilizar os QR Codes para ter maior controle sobre como realizar seu marketing e comunicação na hora de atrair clientes (SOUSA, 2014).

2.3 Smartphones e tecnologia *Android*

De acordo com o IBGE (Instituto Brasileiro de Geografia e Estatística) o telefone celular é o meio mais utilizado pelos brasileiros para acessar a internet. Dados recolhidos em 2015 apontam que 92,1% dos domicílios brasileiros acessam a internet por meio do celular e 70,1% através de um microcomputador.

O Nordeste apresenta o terceiro maior percentual dentre as regiões do país à navegar mais pelo celular com 93,9%, perdendo apenas para as regiões Norte (96,7%) e Centro-Oeste (95,6%), e ficando à frente do Sudeste (91,5%) e do Sul (88,2%). Os resultados apontaram também, que pessoas que trabalham nas áreas de educação, saúde e serviços sociais foram as que mais utilizaram a internet: um total de 87,1% (IBGE, 2016).

Segundo a IDC (*International Data Corporation*) Brasil em 2017 as vendas de smartphones no Brasil atingiram um total de 47.7 milhões de unidades. Tal resultado indica um aumento de 9,7% em relação ao ano de 2016 que teve a marca de 43.48 milhões de aparelhos

vendidos. Foi apontado ainda pelos dados, que 95,1% dos aparelhos vendidos possuem o sistema operacional *Android* e apenas 4,9% possuem o *iOS*, algo que se repete desde 2014 (IDC, 2018).

O *Android OS* foi desenvolvido em 2003 por Andy Rubin, Rich Miner, Nick Sears e Chris White, empresários já iniciados no ramo da tecnologia, que fundaram a *Android Inc*. Dois anos depois a *Google* comprou o sistema e deu impulso ao que viria a se tornar anos mais tarde o sistema operacional mais utilizado no planeta, superando até mesmo o *Windows OS* da *Microsoft*. Nos dias atuais *Android* deixou de ser um sistema operacional exclusivo para smartphones e tornou-se uma tecnologia utilizada em diversos contextos: relógios (*Android Wear OS Smartwatches*), televisores, automóveis (*Android Auto*) e Internet das Coisas (MEYER, 2015).

2.4 Desenvolvimento móvel *Android*

O desenvolvimento móvel ou desenvolvimento *mobile*, como também é conhecido, é todo o processo que envolve desenvolver um sistema de software que será utilizado em uma plataforma móvel, tal como os smartphones. Este tipo de desenvolvimento geralmente se foca na criação de aplicativos para celulares. Aplicativos estes que necessitam de um sistema operacional para funcionarem, tal como um programa de computador (WIKIPÉDIA, 2017).

É importante enfatizar que existem duas abordagens para o desenvolvimento de um aplicativo móvel: a abordagem nativa e a abordagem híbrida.

Na abordagem nativa, a aplicação é desenvolvida para uma plataforma específica, à exemplo a plataforma *Android*. Aplicações nativas são mais rápidas em execução por utilizarem código em linguagens de programação específicas para a plataforma, além de facilitar a construção da aplicação e dialogar melhor com o hardware dos aparelhos. Já no caso do desenvolvimento híbrido, a aplicação é desenvolvida para várias plataformas, à exemplo *Android* e *iOS*. Este tipo de desenvolvimento é popular por ser bem mais rápido que o desenvolvimento nativo quando se fala em codificação e também porque existem diversos *frameworks* que permitem o uso de linguagens de programação e estilização vindas do desenvolvimento web como: HTML, CSS e JavaScript (SCUDERO, 2017).

Uma vez tomada a decisão de desenvolver nativamente, o passo seguinte é selecionar a versão da plataforma *Android*. Mesmo sendo uma tecnologia recente, ela é atualizada constantemente e já possui mais de oito versões diferentes em vigor ao redor do mundo.

Justamente por essa variedade de versões no mercado, é disponibilizada aos desenvolvedores uma tabela de distribuição que é constantemente atualizada. O Quadro 1 mostra a checagem de distribuição de versões do *Android* mais recente, realizada durante um intervalo de sete dias encerrado em oito de janeiro de 2018[3]. Versões do sistema com menos de 0,1% de distribuição não são exibidas.

Quadro 1 - Distribuição da plataforma *Android*.

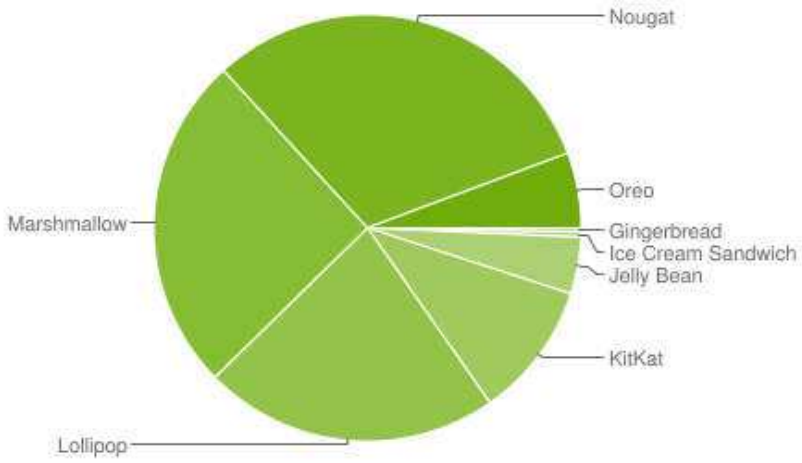
Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.5%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.3%
5.0	Lollipop	21	4.8%
5.1		22	17.6%
6.0	Marshmallow	23	25.5%
7.0	Nougat	24	22.9%
7.1		25	8.2%
8.0	Oreo	26	4.9%
8.1		27	0.8%

Fonte: *Android Developers*; 2018.

O gráfico 1 é uma representação auxiliar criada com as porcentagens do quadro.

[3]**Painéis**. Android Developers, 2018. Disponível em: <https://developer.android.com/about/dashboards/>. Acesso em: 18 mai. 2018.

Gráfico 1 - Distribuição da plataforma Android



Fonte: *Android Developers*; 2018.

A *Google* sugere o uso do *Android Studio* [4]que é o Ambiente de Desenvolvimento Integrado (IDE) próprio da plataforma como ferramenta para desenvolvimento. Algumas das características do software são: editor de código inteligente que permite escrever código nas linguagens Java, Kotlin e C/C++; emuladores e conexão com aparelhos físicos que possuam o sistema operacional para realizar testes; rápido editor de *layout* para construir as interfaces gráficas dos aplicativos, podendo-se fazer uso do sistema de arrastar e soltar para criar os elementos gráficos na tela ou podendo desenhá-los em linhas de código com a linguagem XML.

Por conter estas e várias outras funcionalidades integradas, o *Android Studio* é um software que requer um bom poder de hardware para funcionar, o que pode dificultar sua utilização durante a implementação da aplicação.

2.5 Tecnologias de Desenvolvimento

2.5.1 A linguagem Kotlin

Em 2017 a *Google* anunciou a que a linguagem de programação Kotlin, criada pela JetBrains seria a mais nova linguagem de programação oficial para desenvolvimento *Android*

[4]**Download do Android Studio e das ferramentas SDK**. Android Developers, 2018. Disponível em: <https://developer.android.com/studio/>. Acesso em: 18 mai. 2018.

nativo (antes grupo reservado para Java e C++). Dentre as várias razões para a adoção desta nova linguagem, a *Google* destaca o fato da linguagem ser mais concisa, expressiva e projetada para ser *type-safe* e *null-safe*. A linguagem é ainda multi-paradigma, agregando tanto o paradigma de orientação à objetos, quanto o paradigma funcional (AVRAM, 2017).

Kotlin tem a capacidade de se integrar totalmente ao código Java, sendo possível misturar ambas as linguagens em um mesmo código. Além disso Kotlin também roda na JVM.

A linguagem possui também *frameworks* para melhorar a experiência de desenvolver para *Android* tornando mais simples a criação de código que interage com *views* e *threads* (MITRUT, 2017).

É interessante notar como esta linguagem não deixou de utilizar a linguagem Java, o que permite que novas aplicações sejam facilmente traduzidas de uma linguagem para outra. Java foi a primeira escolha quando o *Android* tornou-se livre para implementação, e não se pode deixar de lembrar que é uma poderosa linguagem de programação que segue o paradigma de Orientação à Objetos, sendo utilizada em diversos contextos no mundo da programação, mesmo que tenha maior popularidade no desenvolvimento de *websites*, em específico no *back-end* (Oracle; 2018).

Kotlin tornou-se tão inclusa no desenvolvimento móvel nativo para *Android*, que está incluído como uma das peças chave no novo *framework* de componentes lançado para a plataforma, o Android Jet Pack[5].

2.5.2 XML

XML (*Extensible Markup Language*) é uma linguagem de marcação especial, diferente do famoso HTML que é utilizado para estruturar conteúdo de páginas web. O XML tem bastante uso na construção de documentos que organizam dados de forma estruturada. No desenvolvimento nativo *Android* ela é utilizada para criar e estruturar os elementos da interface gráfica como botões, textos, listas de itens, etc. (W3C, 2008).

2.6 Metodologias ágeis de desenvolvimento

Para Sommerville (2011) metodologias ágeis são métodos de desenvolvimento incremental em que os incrementos são pequenos e, normalmente, as novas versões do sistema

[5] Disponível em< https://developer.android.com/jetpack/>. Acesso em 10 de novembro de 2018.

são criadas e disponibilizadas aos clientes a cada duas ou três semanas. Estas abordagens buscam sempre envolver os clientes no processo de desenvolvimento para obter feedback rápido sobre a evolução dos requisitos. Desta forma, a documentação é minimizada, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

Como mencionado, o desenvolvimento ágil busca realizar uma entrega contínua de software. Ao invés de uma única entrega com todo o pacote de funcionalidades, os clientes recebem o produto em partes menores, chamadas de releases. Cada release abrange uma parte dos requisitos do sistema. Como o produto é entregue em partes, os *stakeholders* participam em reuniões com a equipe de desenvolvimento para estabelecer os requisitos prioritários do produto, estes por sua vez serão criados primeiro e os demais serão acrescentados através de um processo de entrega iterativo (SOMMERVILLE, 2011).

Ainda segundo Sommerville (2011), muito se tem em comum dentre as diversas metodologias ágeis de desenvolvimento, entretanto cada uma delas tem seus detalhes que as diferenciam das demais. Abaixo são descritas algumas destas metodologias, incluindo a metodologia que foi utilizada na criação da versão atual do aplicativo proposto neste trabalho, que também será utilizada em sua continuação.

2.6.1 EXtreme Programming (XP)

A metodologia XP, cujo nome foi dado por Beck (2000) é provavelmente a mais conhecida abordagem ágil. Ela recebeu este nome pois seu propósito era impulsionar práticas reconhecidamente boas, como o desenvolvimento iterativo, a níveis 'extremos'.

Por exemplo, segundo a metodologia XP, é possível que várias novas releases sejam desenvolvidas, implantadas e testadas em um único dia por programadores diferentes (SOMMERVILLE, 2001).

Segundo Sommerville (2012) este processo de trabalho 'extremo' se torna possível graças a dois pontos chave da XP: os requisitos do sistema são tratados como cenários de utilização (que dentro da metodologia são chamados de Estórias do Usuário ou *User Stories*) que na implementação são divididos em uma série de tarefas e é utilizada a programação em pares, onde dois desenvolvedores trabalham juntos alternando na codificação e desenvolvendo testes para cada tarefa das *User Stories* antes de escreverem o código em si. Quando o novo código é integrado ao sistema, todos os testes devem ser realizados com sucesso.

O Quadro 2 descreve as práticas da metodologia XP que refletem diretamente os princípios das abordagens ágeis.

Quadro 2 - Características da XP.

Princípio ou prática	**Descrição**
Planejamento incremental	Os requisitos são gravados em cartões de estória (ou *Story Cards*) e as estórias que serão incluídas em um release são determinadas pelo tempo disponível e sua relativa prioridade.
Pequenos releases	Em primeiro lugar, desenvolve-se um conjunto mínimo de funcionalidades úteis, que fornecem o valor do negócio. Releases do sistema são frequentes e gradualmente adicionam funcionalidades ao primeiro release.
Projeto simples	Cada projeto é realizado para atender às necessidades atuais, e nada mais.
Desenvolvimento *test-first*	Um framework de testes iniciais automatizados é usado para escrever os testes para uma nova funcionalidade antes que a funcionalidade em si seja implementada.
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente assim que encontrarem formas de melhorar o código. Isso mantém o código simples e manutenível.
Programa em pares	Os desenvolvedores trabalham em pares, verificando o trabalho dos outros e prestando apoio para um bom trabalho sempre.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo que não se desenvolvam ilhas de expertise. Todos os conhecimentos e todos os desenvolvedores assumem responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Assim que o trabalho em uma tarefa é concluído, ele é integrado ao sistema como um todo. Após essa integração, todos os testes de unidade do sistema devem passar.
Ritmo sustentável	Grandes quantidades de horas-extra não são consideradas aceitáveis, pois o resultado final, muitas vezes, é a redução da qualidade do código e da produtividade a médio prazo.
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível todo o tempo à equipe de XP. Em um processo

	de *Extreme Programming*, o cliente é um membro da equipe de desenvolvimento e é responsável por levar os requisitos de sistema para implementação.

Fonte: Sommerville, 2011.

2.6.2 Scrum

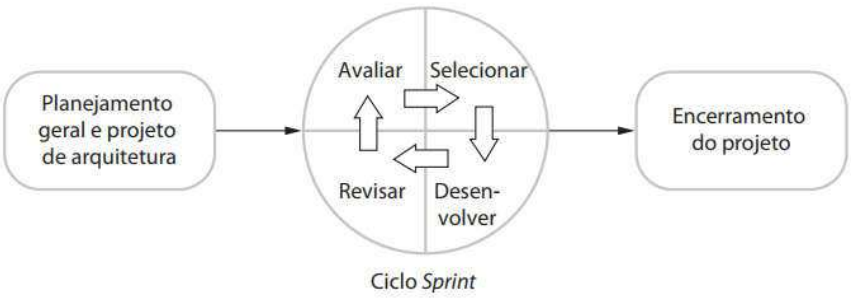
A abordagem Scrum (SCHWABER, 2004; SCHWABER e BEEDLE, 2001) é uma metodologia ágil geral, ou seja, ela pode ser aplicada em qualquer contexto pois Scrum é uma abordagem de gerenciamento de projetos ágeis com foco em gerenciar o desenvolvimento iterativo.

Como Scrum não é necessariamente uma abordagem específica para desenvolvimento de software, ela não prescreve práticas de programação como a programação em pares ou o desenvolvimento *test-first* da XP (SOMMERVILLE, 2001).

Segundo Sommerville (2011), há três fases dentro do Scrum. A primeira é a fase de planejamento geral, em que se estabelecem objetivos gerais do projeto e da arquitetura do software. Em seguida, ocorre a fase central, chamada de ciclos de *sprint*, onde cada *sprint* é uma unidade de planejamento na qual o trabalho a ser feito é avaliado, os recursos para o desenvolvimento são selecionados e o software é implementado, produzindo assim um incremento do sistema. E por fim, a fase final do projeto encerra-o, completa a documentação exigida, como quadros de ajuda do sistema e manuais do usuário e avalia as lições que a equipe Scrum aprendeu durante a execução do projeto.

A figura 6 apresenta o processo das três fases da metodologia Scrum.

Figura 6 - Fases do Scrum.



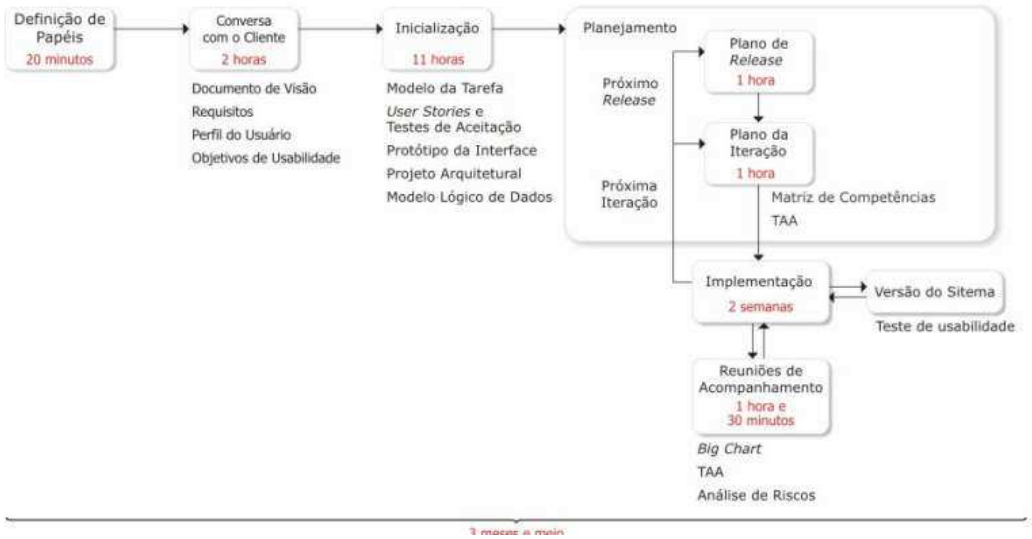
Fonte: Sommerville, 2011.

2.6.3 easYProcess (YP)

Criado em 2007 pelo Departamento de Sistemas de Computação da Universidade Federal de Campina Grande (UFCG) dentro do Programa de Educação Tutorial (PET), o *easYProcess* (YP) é um processo de desenvolvimento de software concebido para ser utilizado em meio acadêmico. Mais precisamente como metodologia utilizada nos projetos de disciplinas de Engenharia de Software (GARCIA et al, 2007).

Justamente por buscar atuar neste contexto, o YP busca ser um processo simples que sirva como uma porta de entrada para os estudantes no mundo da Engenharia de Software. Ele foi concebido utilizando as melhores práticas de metodologias ágeis que estavam bastante em vigor no mercado da época: *Extreme Programming* (XP), *Rational Unified Process* (RUP) e *Agile Modeling*. A figura 7 apresenta a síntese do fluxo do YP.

Figura 7 - Fluxo da YP.



Fonte: *easYProcess*, 2007.

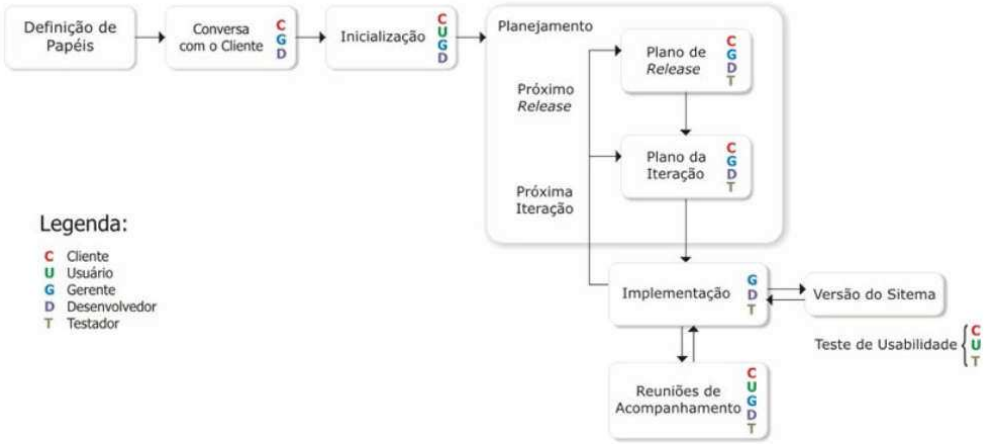
2.6.3.1 Definição de papéis

Segundo Garcia et al (2007), o primeiro passo para iniciar um projeto com YP é dividir a equipe do projeto dando a cada membro um ou mais papéis. Dentro do *easYProcess* há no total cinco papéis:

- Cliente: papel desempenhado por quem solicitou o desenvolvimento do software, o que não o condiciona ao uso do produto gerado, ou seja, nem sempre quem requisita o desenvolvimento do software vai ser usuário do mesmo.
- Usuário: é quem vai de fato utilizar o sistema que será produzido. A presença do usuário no processo permite que o sistema seja desenvolvido de acordo com seu perfil, suas habilidades e necessidades, já que ele é quem vai utilizar o sistema é necessário que o(s) desenvolvedor(es) entenda os gostos e preferências do usuário e respeite-os.
- Gerente: é o responsável por coordenar as atividades de todos os outros membros da equipe. O gerente deve ser capaz de gerenciar o desenvolvimento e tomar decisões referentes aos riscos e rumos do projeto.
- Desenvolvedor: o papel do desenvolvedor consiste em modelar os requisitos do sistema, e posteriormente produzir um código eficiente e correto que corresponda a tais requisitos. É seu dever verificar a existência de falhas em seu código e, se encontradas, corrigi-las.
- Testador: responsável por revisar o código gerado pelos desenvolvedores e por realizar testes de integração do sistema, além de implementar os testes de aceitação definidos pelo cliente.

A alocação de papéis deve ser feita de acordo com a necessidade do escopo do projeto, portanto deve-se levar em conta as habilidades e personalidades de cada membro da equipe na hora de dividir os papéis. É extremamente importante que todos os cinco papéis estejam presentes e muitas vezes pode haver acúmulo e rotação de papéis entre os membros da equipe. A figura 8 descreve a distribuição dos papéis do YP dentro do fluxo de processos.

Figura 8 - Distribuição de papéis do YP



Fonte: *easYProcess*, 2007.

2.6.3.2 Primeira conversa com o cliente

De acordo com o YP (2007) a primeira conversa com o cliente consiste em uma coleta inicial de informações sobre o sistema que será desenvolvido. A partir dessa conversa, o cliente e os desenvolvedores passam a ter um entendimento comum a respeito do sistema. O desenvolvedor deve tentar extrair do cliente a maior quantidade de informação possível sobre o sistema solicitado, ou seja, deve buscar por informações relacionadas às principais funcionalidades do sistema (requisitos funcionais), aos requisitos não-funcionais mais importantes, ao perfil do usuário, aos objetivos de usabilidade, aos testes de aceitação, entre outros.

Após a conversa com o cliente, o desenvolvedor deve montar o documento de visão, que é um documento onde as informações coletadas na conversa são transformadas em artefatos que irão guiar a equipe de desenvolvimento do projeto durante todo o processo de criação do produto de software. O documento de visão deve ser feito utilizando uma linguagem de fácil entendimento para o cliente, uma vez que deve ser usado como elo de comunicação entre este e o desenvolvedor. Uma vez produzido, o documento deve ser validado junto ao cliente no intuito de se descobrir se a visão de negócio do sistema, definida pela equipe de desenvolvimento, realmente está correta (*easYProcess*, 2007).

2.6.3.3 Inicialização

Nesta etapa a equipe de desenvolvimento deve conceber um modelo de tarefas para que sirva de base na construção do protótipo da interface que tem a função de receber o feedback do cliente nas fases iniciais de implementação e listar todas as *User Stories* do sistema (e seus respectivos Testes de Aceitação) a fim de desenvolver um projeto arquitetural e um modelo lógico de dados capazes de acomodar mudanças e que sejam estáveis o suficiente para que riscos sejam minimizados.

2.6.3.4 Planejamento

Na etapa de planejamento, a equipe deve estar ciente do tempo disponível para o desenvolvimento do projeto, e a partir de então, com o cliente, definir o número de *releases* e iterações necessárias para a conclusão do mesmo.

Como o *easYProcess* foi concebido para utilização em disciplinas de Engenharia de Software é recomendado que o projeto possua 3 releases por semestre letivo, cada uma contendo duas iterações de 2 semanas cada – três meses no total.

Como o YP propõe pequenas *releases* na tentativa de garantir uma maior interação entre a equipe de desenvolvimento e o cliente, as *User Stories* alocadas em cada release só podem ser consideradas como finalizadas após a realização de seus respectivos Testes de Aceitação. Caso alguma atividade não seja completada dentro do tempo que foi previsto para a iteração atual, deve-se alocar a(s) atividade(s) dentro da iteração seguinte.

Para a realização do projeto com sucesso o planejamento é dividido em duas partes:

- Plano de *Release*: consiste em definir o período das releases e a partir de então dividi-lo em iterações. Em seguida, as *User Stories* e os Testes de Aceitação associados às mesmas devem ser distribuídos nas iterações da release em questão. Cada *US* deve ter pelo menos um Teste de Aceitação.
- Plano de Iteração: O plano da iteração consiste em quebrar as *User Stories* em atividades menores (caso se note necessário) e em seguida alocar os membros da equipe de desenvolvimento como responsáveis pela realização de cada atividade.

2.6.3.5 Implementação

Consiste na realização das atividades estabelecidas no plano de iteração, onde será produzido o código do sistema, ou seja, onde as funcionalidades serão criadas e testadas, a fim de que seja liberada uma nova versão do sistema.

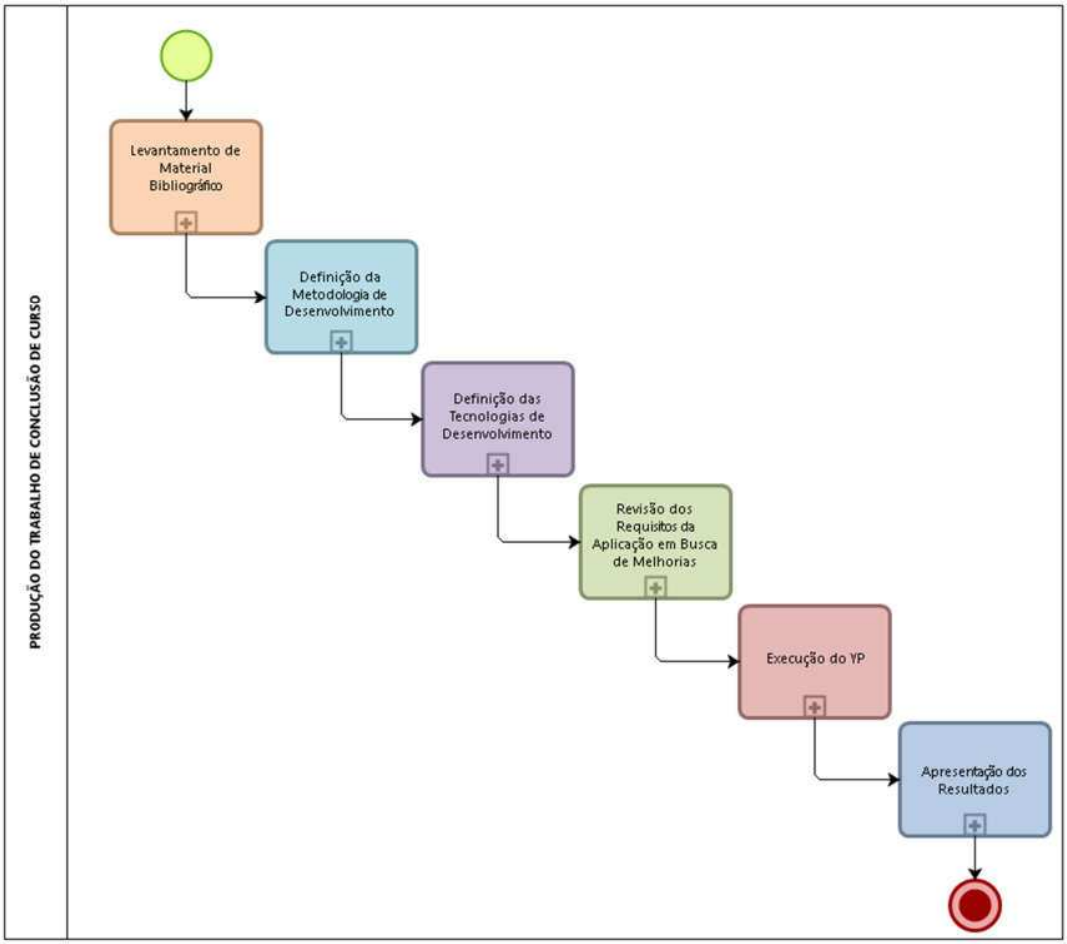
2.6.3.6 Reunião de acompanhamento

A reunião de acompanhamento ocorre semanalmente, de preferência ao final de cada semana de iteração, visando avaliar sistematicamente os resultados obtidos no projeto até o momento. A mesma deve ser coordenada pelo gerente de projeto, que tem fundamental importância na análise dos resultados obtidos. Todos os membros da equipe de desenvolvimento devem participar das reuniões de acompanhamento e nela são gerados artefatos que contêm métricas indicando o progresso do projeto.

3 METODOLOGIA

Nesta seção do documento é descrito todo o processo de execução para a contemplação dos objetivos deste trabalho. A metodologia YP foi selecionada como abordagem de projeto e desenvolvimento da aplicação descrita no presente trabalho. Suas etapas são contempladas levando em conta o que já foi produzido nas disciplinas de Laboratório de Engenharia de Software I e Paradigmas de Linguagens de Programação do Campus VII – Patos da UEPB. Na figura 9 é mostrado o fluxograma das etapas deste trabalho na forma de um macroprocesso.

Figura 9 - Fluxograma da Produção do Trabalho de Conclusão de Curso.

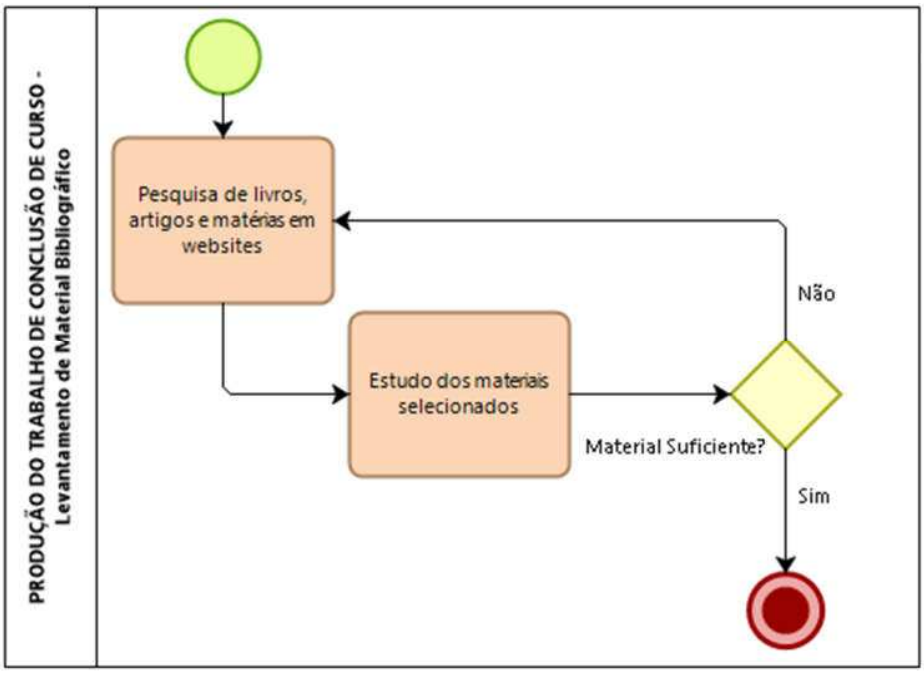


Fonte: própria, modelado com Bizagi Modeler, 2018.

3.1 Levantamento de material bibliográfico

Nesta etapa foram pesquisados e estudados artigos, livros e matérias em *websites* que abordassem os conteúdos descritos neste trabalho com o intuito de obter mais conhecimento sobre o que foi feito no mesmo. A figura 10 mostra o fluxograma desta etapa.

Figura 10 - Fluxograma do Levantamento de Material Bibliográfico.

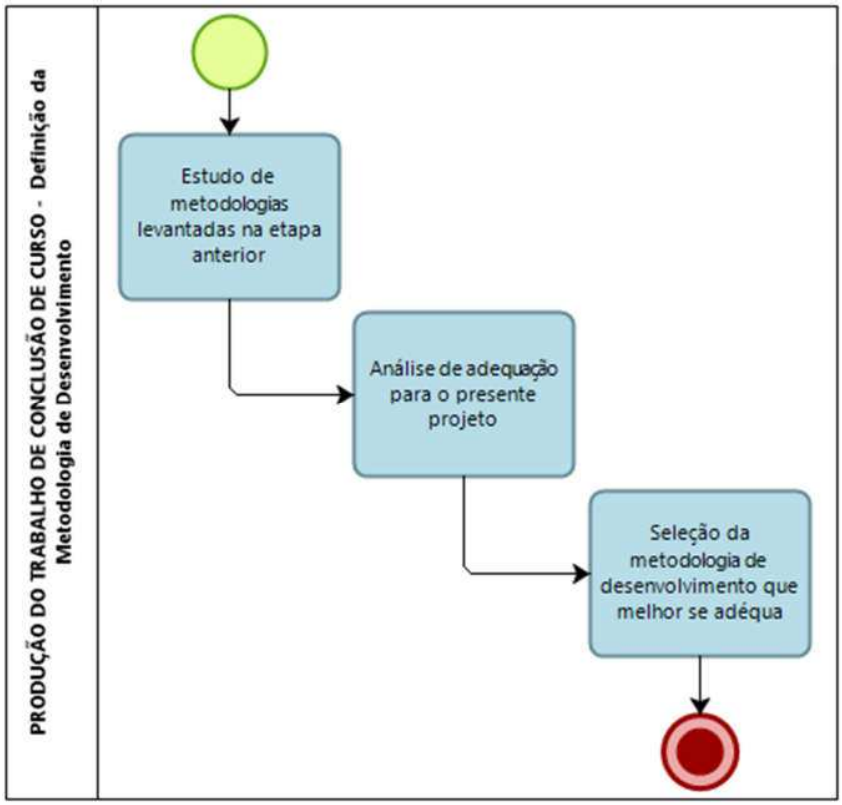


Fonte: própria, modelado com Bizagi Modeler, 2018.

3.2 Definição da metodologia de desenvolvimento

Nesta etapa foram analisadas as metodologias de desenvolvimento ágil descritas na seção 2.7 deste trabalho e somada a experiência prévia obtida durante o desenvolvimento prévio do aplicativo, a escolha foi de manter a utilização da metodologia YP descrita na seção 2.6.3. A figura 11 mostra o fluxograma desta etapa.

Figura 11 - Fluxograma da Definição da Metodologia de Desenvolvimento.

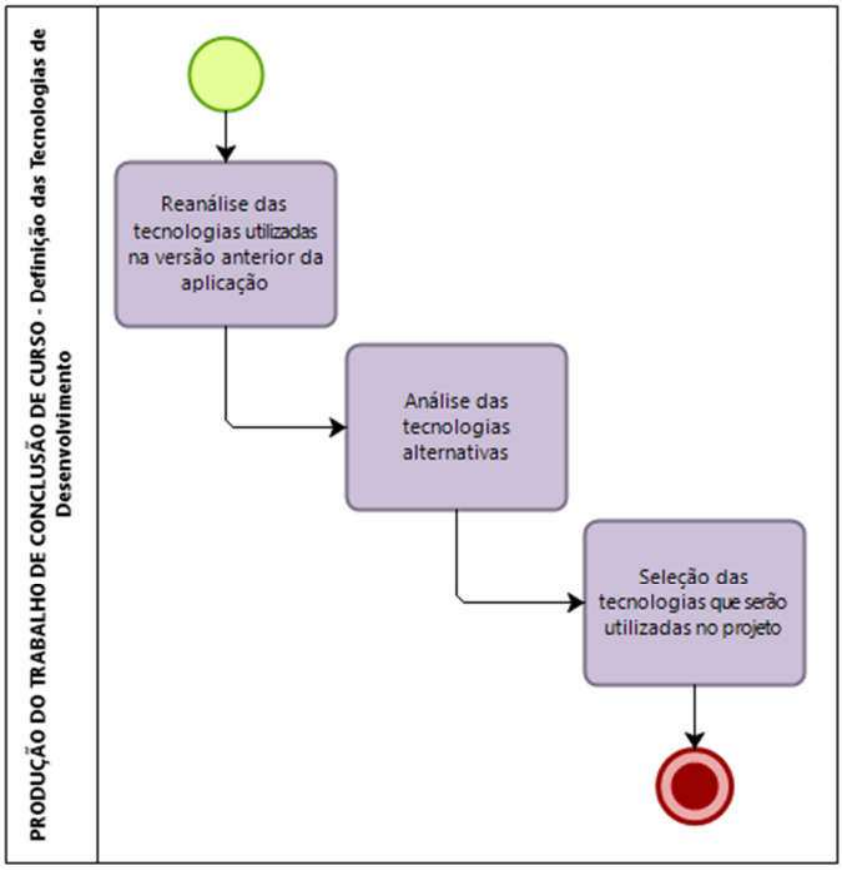


Fonte: própria, modelado com Bizagi Modeler, 2018.

3.3 Definição das tecnologias de desenvolvimento

Nesta etapa foram reanalisadas as tecnologias utilizadas na versão anterior do aplicativo, sendo elas Java, XML e SQLite, então foram analisadas as tecnologias alternativas como Kotlin, MongoDB e REST. Foram selecionadas as três últimas para o desenvolvimento da versão da aplicação produzida neste trabalho. A figura 12 mostra o fluxograma desta etapa.

Figura 12 - Definição das Tecnologias de Desenvolvimento.

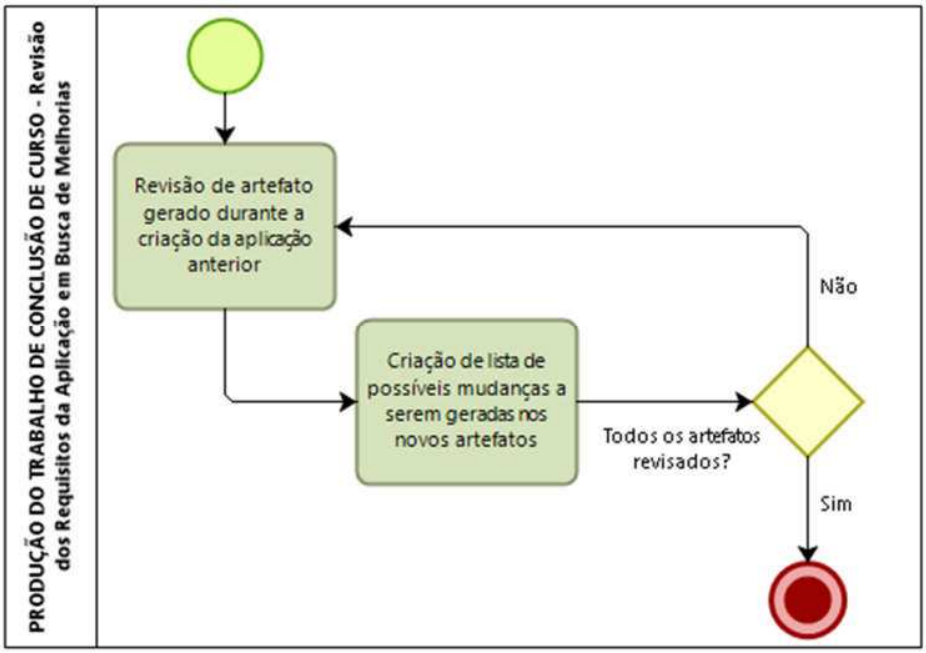


Fonte: própria, modelado com Bizagi Modeler, 2018.

3.4 Revisão dos requisitos da aplicação em busca de melhorias

Nesta etapa foram revisados os requisitos levantados para o desenvolvimento da versão anterior da aplicação, bem como os artefatos gerados durante o processo de projeto e desenvolvimento da versão. Em seguida foram listadas possíveis melhorias a serem feitas na criação da versão produzida neste trabalho. A figura 13 mostra o fluxograma desta etapa.

Figura 13 - Revisão dos Requisitos da Aplicação em Busca de Melhorias.



Fonte: própria, modelado com Bizagi Modeler, 2018.

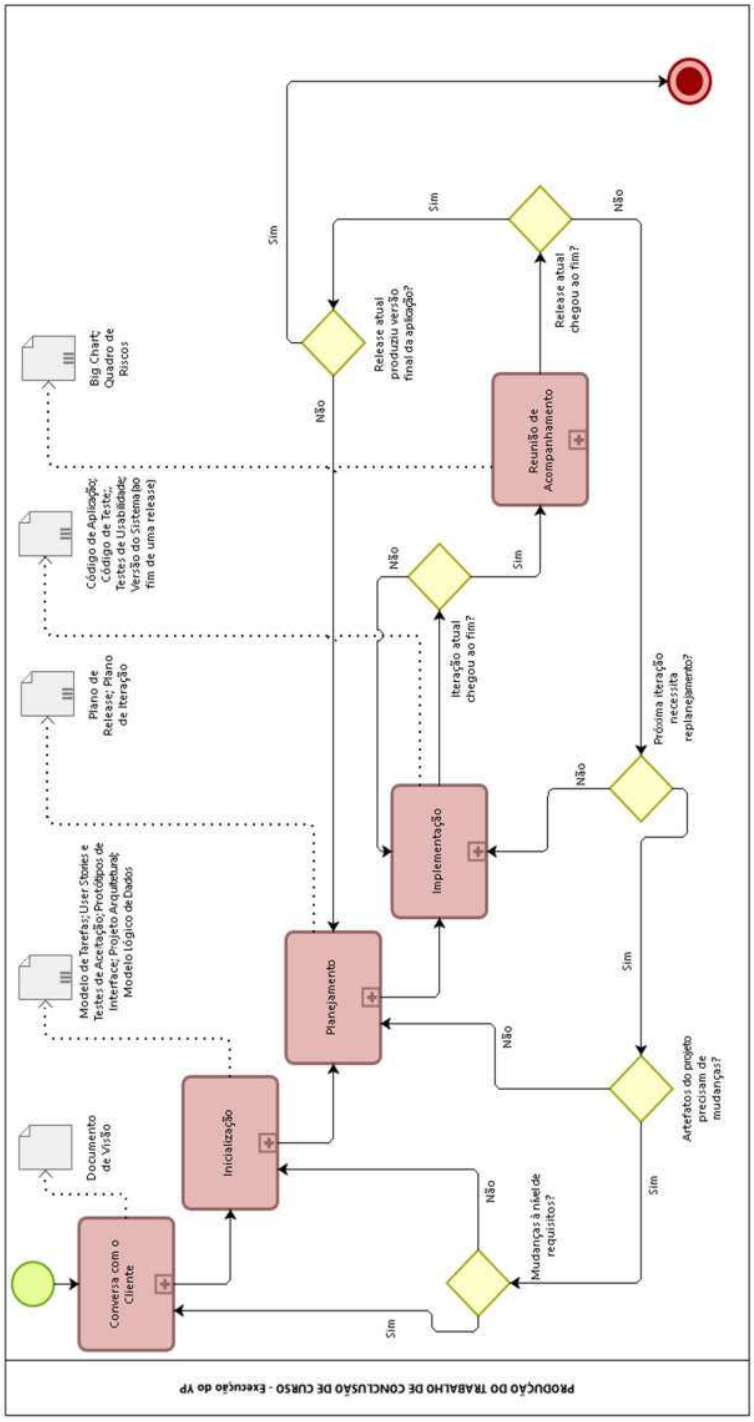
3.5 Execução da YP

Nesta etapa foram realizados os processos que compuseram ambos projeto e implementação do aplicativo utilizando como guia a metodologia YP. Durante este passo, cada uma das etapas que compõem a metodologia foram contempladas sendo também gerados os artefatos requeridos na YP para cada uma das mesmas. Estes artefatos são mostrados na seção 4.2.

Deixa-se claro aqui, que este projeto foi executado por apenas uma pessoa e portanto os cinco papéis presentes na metodologia YP, descritos na seção 2.6.3.1, foram todos assumidos por um mesmo indivíduo. Por conseguinte, a etapa de Definição de Papéis não foi contemplada pois seria redundante para este trabalho. Dito isto, do APÊNDICE A ao APÊNDICE E estão os fluxogramas que representam as etapas cumpridas, as últimas contêm raias mostrando os diferentes papéis. Esta medida foi adotada para que fique claro qual como os papéis foram alternados durante as atividades dentro de cada etapa, salvo as Reuniões de Acompanhamento, em que todos os membros da equipe participam e discutem. Por esta razão, esta etapa não apresenta a mesma divisão em raias das demais.

A figura 14 mostra o fluxograma desta etapa na forma de um macroprocesso.

Figura 14 - Fluxograma da Execução da YP.

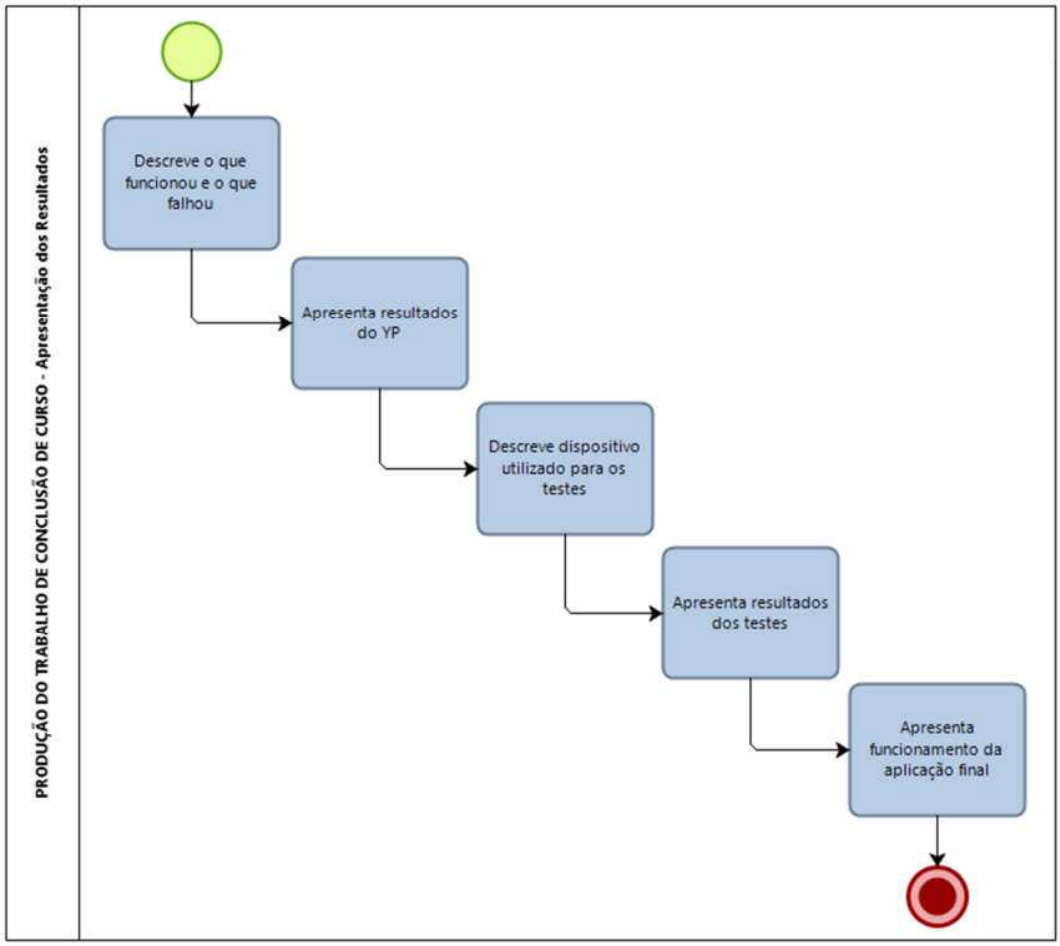


Fonte: própria, modelado com Bizagi Modeler, 2018.

3.6 Apresentação dos resultados

Nesta etapa foram documentos os resultados de todo o projeto. Todos os detalhes estão descritos na seção seguinte. A figura 15 mostra o fluxograma desta etapa.

Figura 15 - Fluxograma da Apresentação de Resultados.



Fonte: própria, modelado com Bizagi Modeler, 2018.

4 RESULTADOS DO TRABALHO

4.1 Quanto à tecnologia do QR Code

O tecnologia de QR Code mostrou-se simples e satisfatória. Embora o *Android framework* não possuísse bibliotecas de código para a implementação do scanner, isto não se mostrou um obstáculo visto que não houve problemas para utilizar bibliotecas externas ao *framework* que permitissem desenvolver esta funcionalidade.

Da mesma forma, a geração dos códigos para as salas do Campus foi simples, visto que o QR Code é de licença livre e bastante difundido, pode ser criado em sites da web.

Os testes realizados para comprovar o bom funcionamento do scanner mostraram ótimos resultados, mostrando rapidez na execução e na apresentação dos resultados.

4.2 Quanto aos resultados da YP

Relembra-se que seguindo a metodologia *easYProcess* para a concepção do aplicativo móvel cada uma das etapas descritas na seção 2.6.3 foi contemplada (excluindo a etapa de Definição de Papéis pois visto que este projeto foi executado por uma pessoa e não uma equipe a etapa seria redundante) e os devidos artefatos foram produzidos durante as mesmas. Abaixo segue a descrição do que foi criado em cada etapa.

4.2.1 Primeira conversa com o cliente

Nesta etapa foi reavaliado o Documento de Visão da versão anterior da aplicação. Tal artefato divide-se em cinco artefatos menores que juntos englobam os requisitos da aplicação.

O Documento de visão é dividido em cinco partes: descrição do sistema; requisitos funcionais; requisitos não-funcionais; perfil do usuário; objetivos de usabilidade.

4.2.1.1 Descrição do sistema

Para utilizar a aplicação, será necessário que os usuários realizem o Login. Para tanto, eles deverão informar sua Matrícula e a Senha do Controle Acadêmico. Caso não esteja

conseguindo acessar, o usuário poderá ir até o site de recuperação de senha do Controle Acadêmico através da mensagem "Não consegue acessar? Clique aqui." presente na tela de Login. Selecionar esta opção irá abrir a página do site no browser do dispositivo.

Após realizar o Login cada usuário do aplicativo encontrará nele seu RDM vindo direto do Controle Acadêmico da UEPB, horário completo de seu curso de acordo com as disciplinas ofertadas no Campus em cada semestre, ambos os itens estarão dispostos em forma de lista onde cada item será uma disciplina. A *app* contará também um scanner de QR Code para acessar as informações contidas nas placas presentes nos diferentes espaços do Campus VII.

Através do RDM e dos horários de curso, os usuários poderão selecionar nas disciplinas listadas a opção de "EXPANDIR" para que sejam mostrados mais detalhes sobre as mesmas. Com a diferença de que as disciplinas listadas no RDM de estudantes possuem detalhes apenas do turno no qual ele está matriculado. No caso dos professores, disciplinas que sejam lecionadas em ambos os turnos (diurno e noturno) irão mostrar ambos.

Haverá também nas disciplinas a opção "LOCALIZAR SALA" através da qual o aplicativo irá abrir o scanner de QR Code. Ao apontar o scanner para um QR Code em alguma placa da faculdade, irá aparecer uma caixa de diálogo na tela do usuário informado se aquela é a sala que o usuário está buscando. A opção de localização de sala irá aparecer no item da lista nas telas de RDM e Horários do Curso ao lado da opção "EXPANDIR" apenas se a disciplina em questão possuir apenas um turno associado. Caso a disciplina possua ambos os turnos, a opção "LOCALIZAR SALA" só será visível quando o usuário expandir a disciplina e visualizar os detalhes de cada turno.

Se for o caso de estar procurando uma sala livre para estudar, através da aba/tela do QR Scanner o usuário pode utilizar diretamente o scanner de QR Code em qualquer placa da universidade e então o aplicativo mostrará os horários em que o espaço referente àquela placa está ocupado.

O usuário poderá consultar a Ajuda do aplicativo através de um botão de ação na barra do aplicativo. Esse será mostrado na forma de um ícone de Ajuda (ou ícone de interrogação).

4.2.1.2 Requisitos funcionais

Os requisitos funcionais descrevem as funções que a aplicação deve conter. Aqui eles estão divididos na forma de componentes, para melhor separar os conjuntos de funções do aplicativo.

Componente 01 – Login do Usuário

- Realizar Login
- Acessar recuperação de senha no CA

Componente 02 – RDM e Horário do Curso

- Visualizar lista de disciplinas do RDM/Horário do Curso
- Expandir disciplinas para obter mais detalhes
- Localizar sala de aula através do scanner de QR Code

Componente 03 – Scanner de QR Code

- Consultar diferentes localidades através do scanner

Componente 04 – Ajuda

- Consultar a ajuda do aplicativo

4.2.1.3 Requisitos não-funcionais

Os requisitos não-funcionais representam caraterísticas da aplicação em termos de confiabilidade, desempenho, usabilidade, tecnologias, entre outros.

Quadro 3 - Requisitos não-funcionais.

Requisitos	Mensuração/Descrição
Requisitos de facilidade de uso	Encontram-se descritos nos objetivos de usabilidade.
Requisitos de confiabilidade	Confiável
Requisitos de portabilidade	Android OS
Requisitos de interoperabilidade	Não se aplica ao software
Requisitos éticos	Não se aplica ao software
Requisitos de entrega	Entrega até 19/11/2018
Requisitos de implementação	Kotlin, Java

Requisitos de banco de dados	MongoDB
Requisitos de padrões	DAO, Adapter, Builder
Requisitos de desempenho	Resposta de até 4 seg. por comando que não necessite de internet e de 15 segundos para funções que necessitem de internet.
Requisitos de espaço	10 MB
Requisitos de privacidade	Público

Fonte: própria, 2018.

4.2.1.4 Perfil do Usuário

O perfil do usuário descreve diferentes características do usuário final do sistema. Na Parte I são apresentadas características gerais; na Parte II são descritos níveis de experiência que o usuário deve ter para utilizar o sistema e na Parte III é descrito o estilo cognitivo do usuário.

Quadro 4 - Perfil do Usuário.

PERFIL DO USUÁRIO					
Características do usuário, escolhidas pelo projetista de acordo com a relevância para o projeto. Levantamento baseado em:					
Fatos	X	**Opinião do Usuário**		**Dados medidos ou observados**	
Parte I - Características Gerais					
Faixa etária	A partir de 17 anos.		**Sexo**	Masculino e Feminino	
Habilidades necessárias para executar a tarefa	Ser estudante ou professor do Campus VII da UEPB; saber operar um smartphone				
	Níveis de percepção		Percepção visual ou tátil		
	Habilidades Motoras		Precisão, coordenação motora		

Grau de instrução	Médio
Função desempenhada na organização	Usuário
Tarefas realizadas na Função	Consulta de horário de curso, montagem de horário pessoal, visualização do mapa do Campus, consulta de localidade no mapa do Campus
Frequência de execução das Tarefas na Função	Semestral
Objetivos (o que pretende com o sistema)	Facilitar a localização de localidades do campus e a familiarização com o ambiente da faculdade
Motivações (por que usaria o sistema)	Saber encontrar alguma localidade específica no Campus VII
Preferências	Uso do toque na tela do smartphone

Parte II - Conhecimento Conceitual (necessário à execução de tarefas)	
Conhecimento Semântico	**Nível de Experiência**
Função	Médio
Método	Médio
Tarefa	Alto
Ferramenta de execução das tarefas	Alto
Conhecimento Sintático	**Nível de Experiência**
Uso da tela para toque	Alto
Uso de dispositivos especiais de interação	Baixo
Uso de terminologia específica	Alto

Parte III - Estilo Cognitivo	
Aprendizado	Com a prática

Capacidade de solucionar problemas		Sem ajuda	
Capacidade de reter o aprendizado		Alta	
Personalidade			
Nível de curiosidade		Médio a elevado	
Nível de persistência		Elevado	
Inovador		**Impulsivo**	
Conservador		**Reflexivo**	X

Fonte: própria, 2018.

4.2.1.5 Objetivos de Usabilidade

Os objetivos de usabilidade descrevem critérios que a aplicação deve cumprir para promover uma boa experiência ao usuário enquanto o mesmo utiliza o sistema.

Quadro 5 - Objetivos de Usabilidade.

Objetivos	Mensuração/Descrição
Facilitar no aprendizado	O usuário poderá aprender a manipular o aplicativo de forma rápida.
Ser atrativo ao usuário	O aplicativo será atrativo para que o usuário tenha satisfação ao utilizá-lo.
Possuir interface simples	Para que o usuário não tenha dificuldades no uso do aplicativo.
Resolução de erros	O usuário concluirá as tarefas realizadas sem falhas.
Disponibilizar boa documentação	Facilidade de encontrar as informações desejadas ao consultar a opção de ajuda.
Clareza nos conteúdos	O usuário terá capacidade de reter conhecimento mesmo sem uso frequente do aplicativo.

Fonte: própria, 2018.

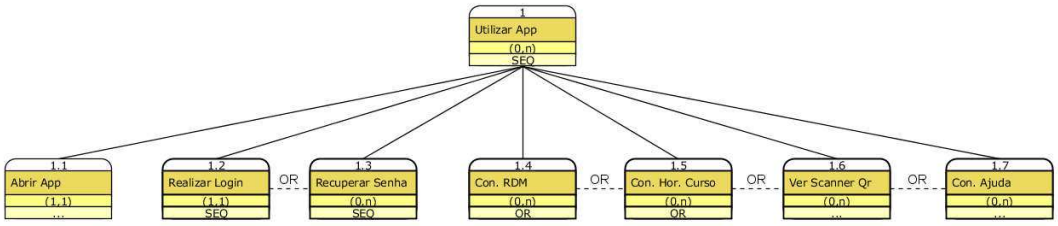
4.2.2 Inicialização

Na Inicialização os seguintes artefatos foram produzidos: Modelo de Tarefas, Protótipos de Interface, Modelo Arquitetural e Modelo Lógico de Dados. Estes itens foram alimentados pelos requisitos coletados através do Documento de Visão.

4.2.2.1 Modelos de Tarefa

Os modelos de tarefa apresentam diferentes fluxos de ações necessárias para executar as diferentes tarefas possíveis dentro da aplicação. A figura 16 mostra o modelo de tarefa da aplicação em seu nível mais alto. Os demais modelos de tarefa encontram-se no APÊNDICE F.

Figura 16 - Modelo de Tarefas da aplicação.



Fonte: própria, modelado com iTAOS, 2018.

4.2.2.2 *User Stories* e Testes de Aceitação

As *User Stories* são criadas tendo como base os requisitos funcionais levantados no Documento de Visão. São gerados também os Testes de Aceitação para cada uma das US, visto que estas, na YP, só são consideradas como completas ou terminadas, quando todos os TA são realizados e passam com resultado seus resultados esperados. O quadro 6 mostra as *User Stories* da aplicação e seus respectivos Testes de Aceitação.

Quadro 6 - *User Stories* e Testes de Aceitação.

US01	**Implementar a funcionalidade de Login**	**Estimativa: 16hrs**

TA01.1	Realizar Login informando matrícula e senha corretos (Login deve ser realizado com sucesso).	
TA01.2	Realizar Login informando matrícula ou senha incorretos (Login não deve ser realizado, uma mensagem de erro deve ser exibida para o usuário.	
TA01.3	Realizar Login deixando campo de matrícula ou de senha em branco (Login não deve ser realizado, uma mensagem de erro deve ser exibida para o usuário).	
US02	**Implementar a funcionalidade do RDM**	**Estimativa: 24hrs**
TA02.1	Expandir uma disciplina através da função "EXPANDIR" para visualizar todos os detalhes sobre a mesma (a disciplina se expande tomando toda a tela, mostrando os detalhes contidos no RDM).	
TA02.2	Localizar a sala de uma disciplina através da função "LOCALIZAR SALA". (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.	
TA02.3	Localizar a sala de uma disciplina através da função "LOCALIZAR SALA" dentro da telha de detalhes da disciplina. (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.	
US03	**Implementar a funcionalidade do Horários do Curso**	**Estimativa: 16hrs**
TA03.1	Expandir uma disciplina através da função "EXPANDIR" para visualizar todos os detalhes sobre a mesma (a disciplina se expande tomando toda a tela, mostrando os detalhes contidos no Horário).	
TA03.2	Localizar a sala de uma disciplina através da função "LOCALIZAR SALA". (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.	
TA03.3	Localizar a sala de uma disciplina através da função "LOCALIZAR SALA" dentro da telha de detalhes da disciplina. (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.	
US04	**Implementar as funcionalidade de Scanner de QR Code**	

		Estimativa: 16hrs
TA04.1	Localizar sala de aula **correta** através da função “LOCALIZAR SALA” das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está correta).	
TA04.2	Localizar sala de aula **incorreta** que fica à esquerda da correta através da função "LOCALIZAR SALA" das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também à quantas salas à esquerda fica a sala correta).	
TA04.3	Localizar sala de aula **incorreta** que fica à direita da correta através da função "LOCALIZAR SALA" das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também à quantas salas à direita fica a sala correta).	
TA04.4	Localizar sala de aula **incorreta** que fica no grupo de salas oposto à sala correta através da função "LOCALIZAR SALA" das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também que o usuário deve procurar nas salas opostas à esta).	
TA04.5	Localizar sala de aula **incorreta** que fica no grupo de salas de outro bloco, não pertencente à sala correta através da função "LOCALIZAR SALA" das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também que o usuário deve procurar em outro bloco de salas do Campus).	
TA04.6	Consultar uma sala de aula através de um QR Code para visualizar seus horários de uso (de acordo com o QR Code da sala, as informações e horários de funcionamento da mesma devem ser apresentadas ao usuário).	
US05	**Implementar a Ajuda do aplicativo**	**Estimativa: 4hrs**
TA05.1	Consultar tela de Ajuda (a tela de Ajuda deve ser aberta quando o usuário selecionar o botão de ajuda na barra do aplicativo).	

Fonte: própria, 2018.

4.2.2.3 Protótipos de Interface

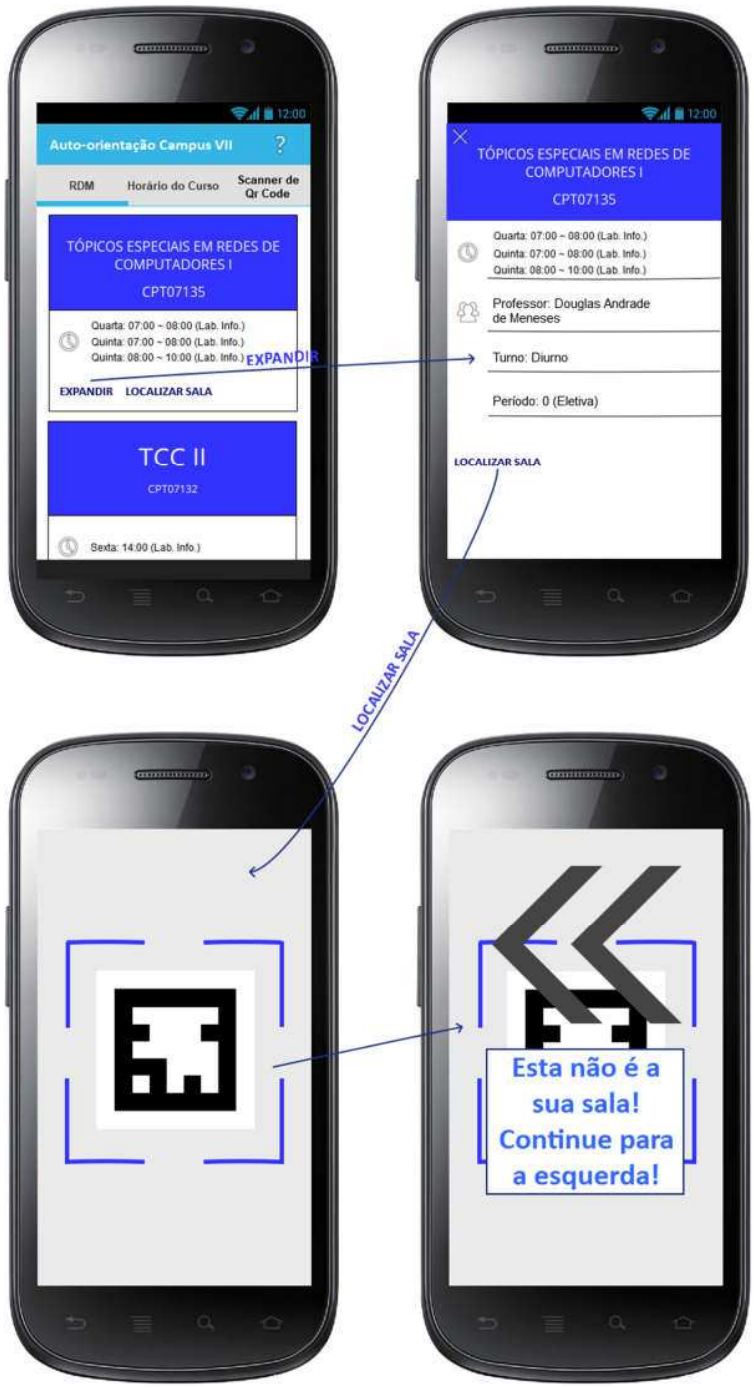
A YP não define um padrão a ser seguido para a criação dos protótipos de interface. Embora a metodologia sugira que os protótipos sejam criados com lápis e papel (de maneira a serem feitos rapidamente e na mesma rapidez, validados pelo usuário), não há uma regra que obrigue a criação ocorrer desta maneira. Por tanto, neste trabalho foi optado por utilizar um software para criação dos protótipos de interface da aplicação, visto que desta maneira as telas geradas seriam mais fáceis de se compreender e ficariam mais semelhantes às telas do produto final.

Figura 17 - Protótipo de Interface: Tela de Login.

Fonte: própria, criado com Pencil, 2018.

A figura 18 mostra o fluxo de telas partindo da tela de RDM, passando pela tela de detalhes de uma disciplina expandida e terminando na tela do scanner de QR code. O fluxo segue a direção das setas.

Figura 18 - Protótipos de Interface: da Tela de RDM a Scanner de QR Code.



Fonte: própria, criado com Pencil, 2018.

A figura 19 mostra o protótipo da tela de ajuda do aplicativo, bem como a forma de acessar a mesma através do botão de ajuda na barra do aplicativo.

Figura 19 - Protótipo de Interface: Tela de Ajuda.

Fonte: própria, criado com Pencil, 2018.

4.2.2.4 Projeto Arquitetural

O software que foi desenvolvido é um aplicativo para smartphones com o sistema *Android*. O aplicativo foi modularizado através de pacotes Java, que permite diminuir o acoplamento entre as partes que compõem o software.

A lógica de negócio está dividida entre componentes do sistema. Parte fica na camada de aplicação, onde a camada visual (XML) é responsável por realizar a interação entre o sistema e o usuário, apresentando as telas e as informações do aplicativo. A segunda parte da camada de aplicação, a camada lógica criada na linguagem Kotlin envia quando necessário, requisições através do protocolo HTTP para o Serviço Web e recebe deste último respostas com dados em

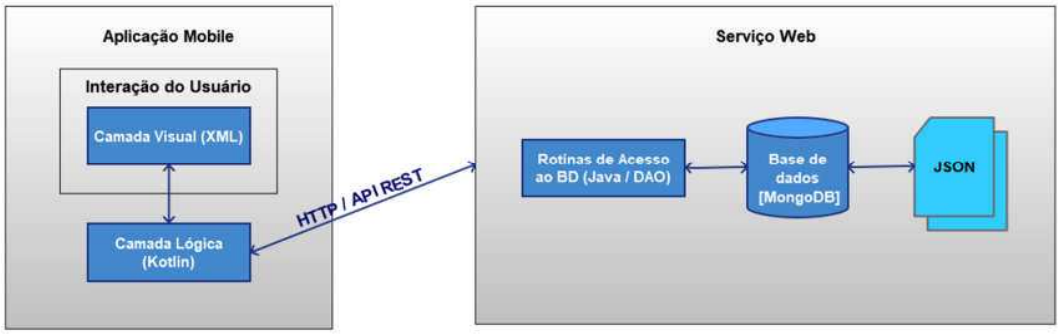
formato JSON. Os dados são tratados quando recebidos e apresentados pela camada de visual para o usuário. O aplicativo consume o Serviço Web através de uma API REST.

A programação foi feita na linguagem de programação Kotlin, que utiliza os Paradigmas de Orientação a Objetos (OO) e Funcional. As ferramentas para desenvolvimento foram o *Android Studio*, a IDE oficial para desenvolvimento *Android* e o Eclipse IDE, para desenvolver o *Web Service*. Este último desenvolvido na linguagem Java.

Na parte do armazenamento de dados, foi utilizado um banco de dados externo ao *Android*, chamado MongoDB. Este é um banco de dados NoSQL (*Not Only SQL*) orientado à documentos, sendo portando um BD não-relacional. Os dados persistidos no MongoDB alimentam o aplicativo através das respostas às requisições lançadas pela camada da aplicação. Os dados enviados são no formato JSON (*JavaScript Object Notation*).

A figura 20 ilustra a descrição da arquitetura da aplicação.

Figura 20 - Projeto Arquitetural.



Fonte: própria, criado com Pencil, 2018.

Ao final do desenvolvimento, para realizar a instalação do aplicativo nos smartphones, foi gerado um arquivo APK, padrão de executáveis para *Android*.

A qualidade do software foi mantida e assegurada através da realização de testes, assim como boas práticas como manter sempre o código limpo e fazer uso de padrões de projeto.

Por fim, foi realizada a entrega contínua do software até que esteja completamente finalizado, de maneira a prezar redução de erros e diminuição de estresse.

4.2.2.5 Modelo Lógico de Dados

Tendo em vista que o MongoDB é um banco orientado a documentos e também NoSQL, ele não segue o paradigma relacional. Desta forma, ao invés de um diagrama de Entidade-Relacionamento, que seria o comum para um modelo lógico em um banco relacional, abaixo seguem imagens que mostram os diferentes documentos mantidos no banco de dados acessado pela aplicação.

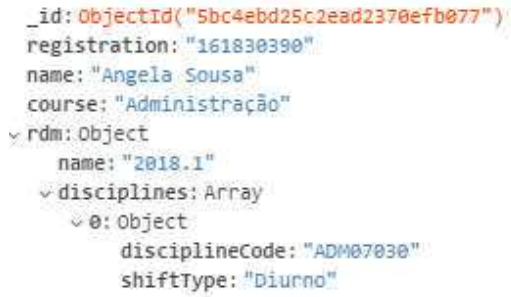
A figura 21 mostra dois exemplos de documentos de credenciais, que possuem campos de login e password de um usuário. Estes documentos são utilizados para validar um usuário no sistema em cada requisição recebida pelo serviço web.

Figura 21 - Documento de Credenciais.

Fonte: própria, 2018.

A figura 22 mostra um documento de usuário, que possue campos como nome, curso, além das informações do RDM.

Figura 22 - Documento de Usuário.



Fonte: própria, 2018.

A figura 23 mostra um documento de disciplina, que possue campos como nome, código, curso e turnos. Estes últimos contêm campos como tipo, professor, período e aulas do período em questão.

Figura 23 - Documento de Disciplina.

Fonte: própria, 2018.

4.2.3 Planejamento

Nesta etapa foram criados os planos de *release* e de iteração. A partir desde ponto, todas as etapas da YP foram revisitadas várias vezes, visto que esta é uma metodologia iterativa. A seguir são mostrados todos os planos produzidos durante todo o trabalho.

4.2.3.1 Planos de *Release*

Os planos de *release* mostram as iterações que serão realizadas para a próxima *release* do sistema, o tempo de duração de cada umas e quais *User Stories* as iterações planejam tratar. O quadro 7 mostra o plano de *release* 01. Os demais planos de *release* encontram-se no APÊNDICE G.

Quadro 7 - Plano da Release 01.

Release 01: 01/10 – 13/10		
Iteração	***User Story***	**Período**
Iteração 01	US01	01/10 – 06/10

Iteração 02	US02	08/10 – 13/10

Fonte: própria, 2018.

4.2.3.2 Planos de Iteração

Os planos de iteração mostram cada iteração separadamente, contendo a *User Story* à ele associada e também os Testes de Aceitação desta. Também são descritas todas as atividades criadas para atingir a desenvolvimento completo da US. Estas, são marcadas ao final da iteração com os *status* C (Completo) ou D (em Desenvolvimento). O quadro 8 mostra o plano da iteração 01. Os demais planos de iteração encontram-se no APÊNDICE H.

Quadro 8 - Plano da Iteração 01 (Legenda: C: Completo, D: em Desenvolvimento).

Iteração 01 – (01/10/2018 – 06/10/2018)				
US01 - Implementar a funcionalidade de Login				
Testes de Aceitação				***Status***
TA01.1	Realizar Login informando matrícula e senha corretos (Login deve ser realizado com sucesso).			C
TA01.2	Realizar Login informando matrícula ou senha incorretos (Login não deve ser realizado, uma mensagem de erro deve ser exibida para o usuário.			C
TA01.3	Realizar Login deixando campo de matrícula ou de senha em branco (Login não deve ser realizado, uma mensagem de erro deve ser exibida para o usuário).			C
Atividade	**Descrição**	**Estimativa de Tempo**	**Tempo Real**	***Status***
A01.1	Implementar a tela de Login (XML)	2h30	2h	C
A01.2	Implementar a Activity do Login	3h	2h	C
A01.3	Implementar recurso do Web Service relacionado ao Login	3h	4h45	C
A01.4	Implementar testes para o recurso do Web Service	3h	5h	C
A01.5	Implementar código dos TA	4h30	8h	C

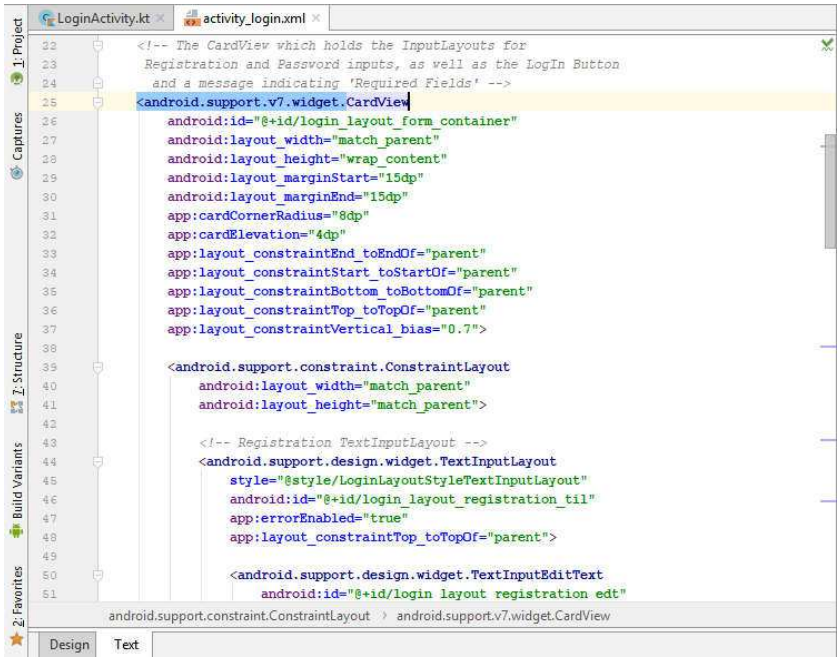
Fonte: própria, 2018.

4.2.4 Implementação

Nesta etapa foi desenvolvido todo o código-fonte da aplicação. Abaixo são mostrados alguns *prints* do desenvolvimento.

A figura 24 mostra um trecho do código na linguagem XML criado no desenvolvimento da tela de login do aplicativo na ferramenta *Android Studio*. Na linha 25, onde está posicionado o cursor, é a *tag* de abertura do componente *CardView*, que nesta tela age como um contêiner para os campos de matrícula e senha do usuário.

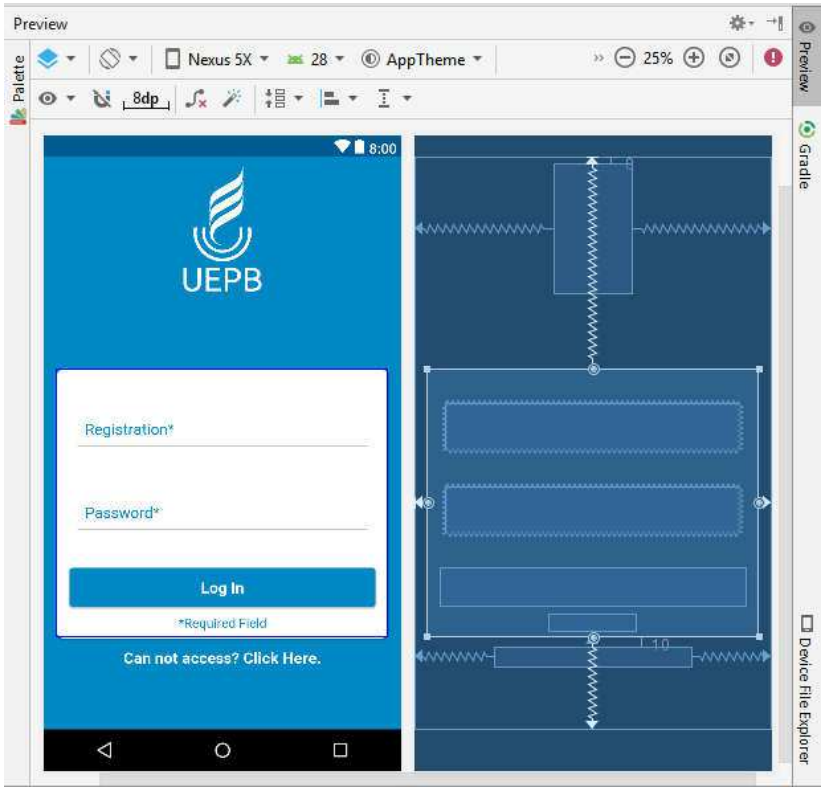
Figura 24 – Trecho do código XML da tela de Login no *Android Studio*.



Fonte: própria, 2018.

A figura 25 mostra a pré-visualização da interface gráfica que é gerada pelo código mostrado na figura anterior. O contêiner em cor branca, que na imagem aparece destacado com uma borda azul, refere-se ao *CardView* mostrado no trecho de código da figura anterior.

Figura 25 - Pré-visualização da tela de Login no *Android Studio*.



Fonte: própria, 2018.

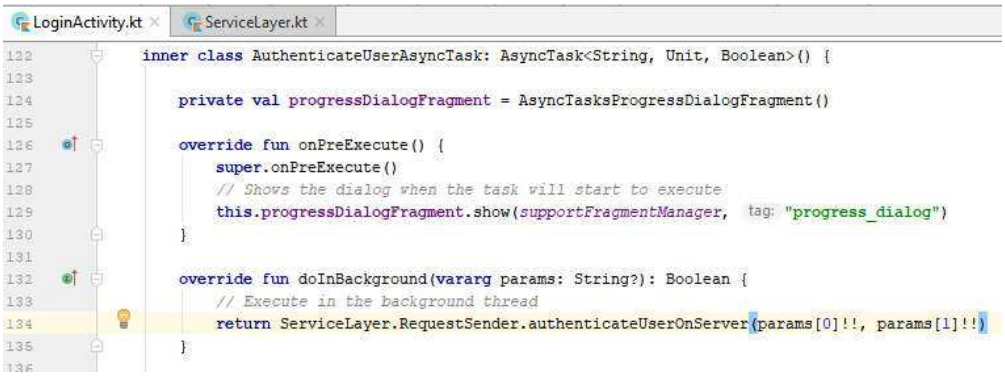
A figura 26 mostra um trecho do código na linguagem Kotlin referente à parte lógica da tela de login do aplicativo. Nas linhas 35 à 38 são criadas variáveis referentes aos componentes criados na tela. As linha 45 à 48 mostram estas variáveis sendo inicializadas recuperando a referência aos elementos em XML da camada visual.

Figura 26 - Trecho do código da Activity da tela de Login: recuperação dos elementos da camada visual.

Fonte: própria, 2018.

A figura 27 mostra mais um trecho do código na linguagem Kotlin referente à parte lógica da tela de login do aplicativo. A linha 134 mostra o ponto em que a *activity* de login envia uma requisição para o serviço web com o intuito de autenticar o usuário que está tentando realizar o login.

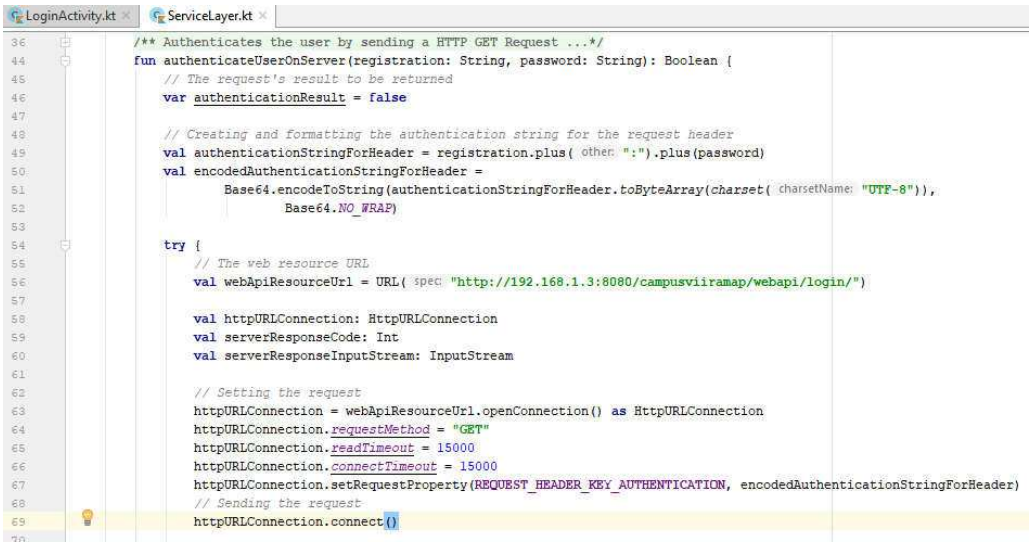
Figura 27 - Trecho do código da Activity da tela de Login: enviando requisição ao serviço web.



Fonte: própria, 2018.

A figura 28 mostra um trecho do código da classe responsável por montar as requisições que são enviadas para o serviço web. Nas linhas 49 à 52 é criado o cabeçalho de autenticação do usuário. Na linha 56 é definido o URL para o qual a requisição é enviada. Nas linhas 63 à 67 é montada a requisição. Na linha 69 a requisição é enviada.

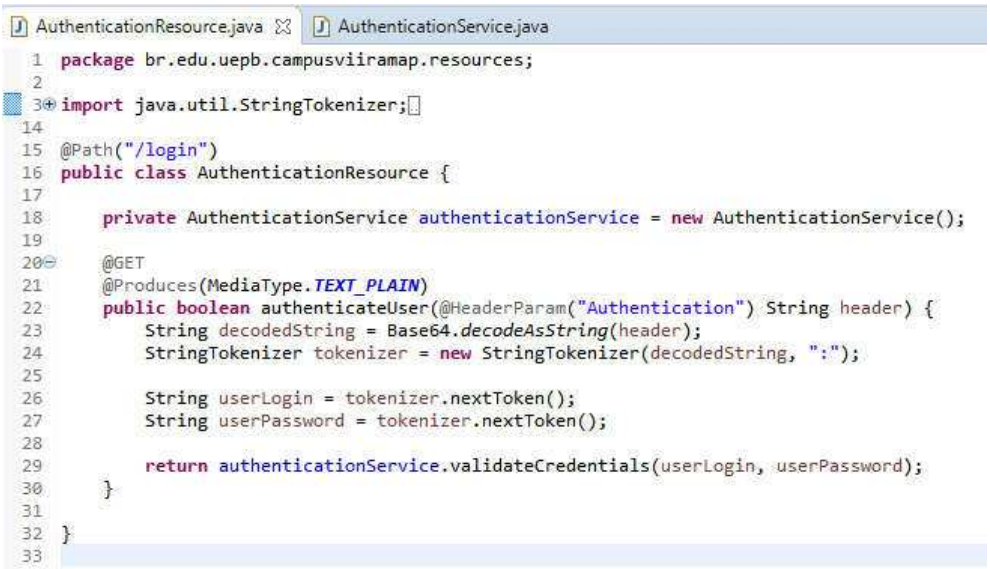
Figura 28 - Trecho do código da classe de envio de requisições para o serviço web.



Fonte: própria, 2018.

A figura 29 a classe em código Java do lado do serviço web responsável por receber a requisição de autenticação do usuário. No método mostrado, o cabeçalho da requisição é recuperado e dele são retirados a matrícula e senha do usuário, que em seguida são enviados para a classe que de fato faz a validação.

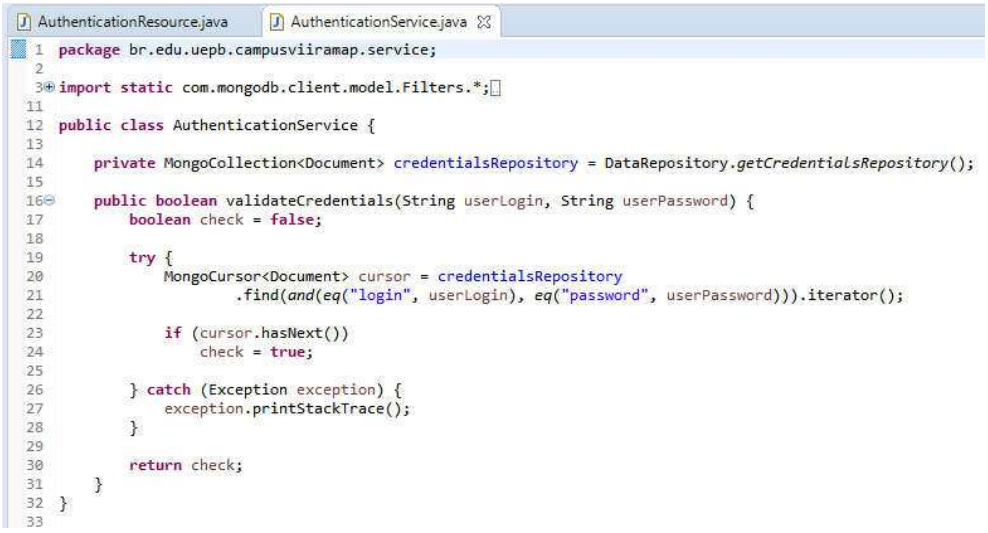
Figura 29 - Classe em Java AuthenticationResource presente no Web Service.



Fonte: própria, 2018.

A figura 30 a classe em código Java do lado do serviço web responsável por realizar a autenticação do usuário. Nas linhas 20 e 21, os dados de matrícula e senha são buscados no banco de dados. A linha 24 checa se o resultado da busca retornou um usuário.

Figura 30 - Classe em Java AuthenticationService presente no Web Service.



Fonte: própria, 2018.

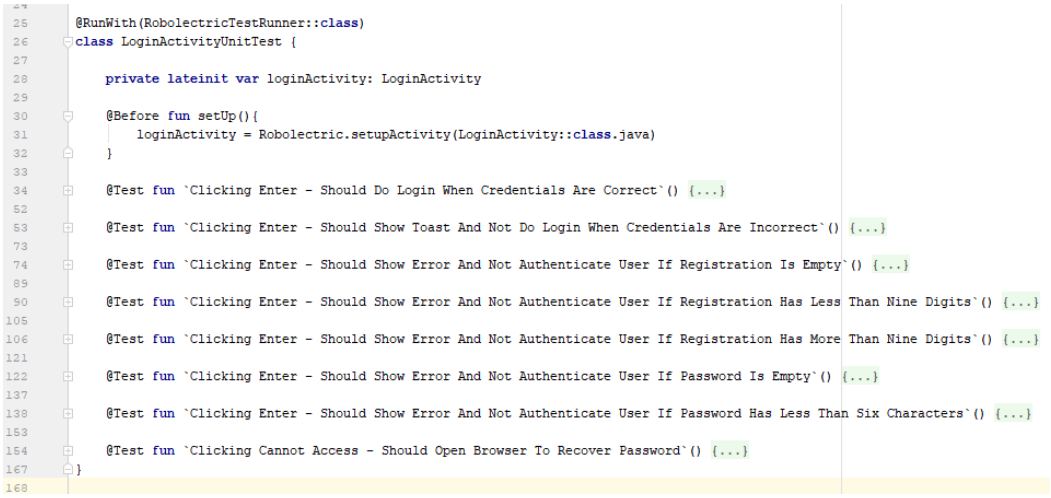
A figura 31 mostra um exemplo de teste instrumentado criado para testar o comportamento da tela de login. Nas linhas 78 e 79 são recuperados os campos de matrícula e senha, respectivamente, e neles são inseridos dados incorretos para realizar o teste. Na linha 81 é recuperado o botão de login e realizada a ação de clique sobre ele. As linhas 83 à 85 testam se a mensagem de erro no login foi exibida corretamente.

Figura 31 – Exemplo de teste da tela de login.

Fonte: própria, 2018.

A figura 32 mostra a nomenclatura dos testes de unidade para a tela de login, criados na linguagem Kotlin. Esta permitiu nomear os testes de unidade utilizando sentenças com espaços e traços entre as palavras, tornando os testes mais legíveis e fáceis de compreender.

Figura 32 - Nomenclatura de testes de unidade na linguagem Kotlin.

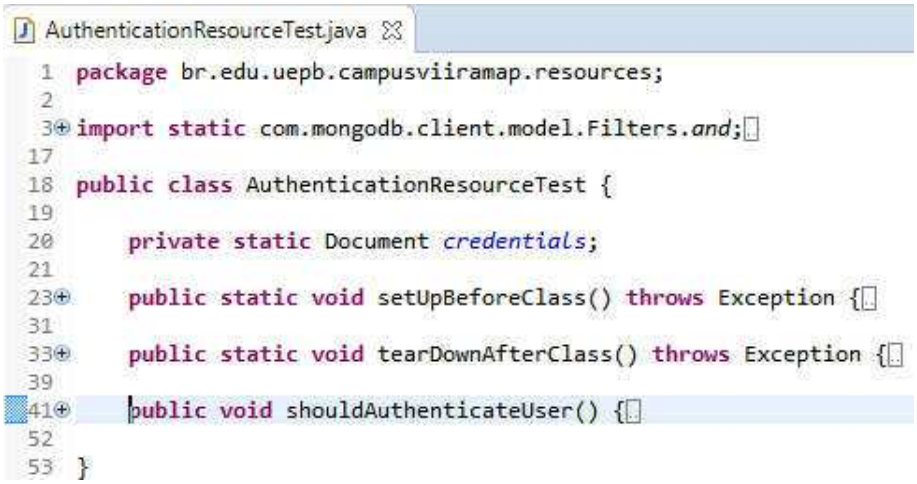


```
@RunWith(RobolectricTestRunner::class)
class LoginActivityUnitTest {

    private lateinit var loginActivity: LoginActivity

    @Before fun setUp(){
        loginActivity = Robolectric.setupActivity(LoginActivity::class.java)
    }

    @Test fun `Clicking Enter - Should Do Login When Credentials Are Correct`() {...}

    @Test fun `Clicking Enter - Should Show Toast And Not Do Login When Credentials Are Incorrect`() {...}

    @Test fun `Clicking Enter - Should Show Error And Not Authenticate User If Registration Is Empty`() {...}

    @Test fun `Clicking Enter - Should Show Error And Not Authenticate User If Registration Has Less Than Nine Digits`() {...}

    @Test fun `Clicking Enter - Should Show Error And Not Authenticate User If Registration Has More Than Nine Digits`() {...}

    @Test fun `Clicking Enter - Should Show Error And Not Authenticate User If Password Is Empty`() {...}

    @Test fun `Clicking Enter - Should Show Error And Not Authenticate User If Password Has Less Than Six Characters`() {...}

    @Test fun `Clicking Cannot Access - Should Open Browser To Recover Password`() {...}
}
```

Fonte: própria, 2018.

A figura 33 mostra a nomenclatura de testes do serviço web, estes feitos na linguagem Java. Esta que não permite o mesmo tipo de liberdade na nomenclatura dos testes da linguagem Kotlin.

Figura 33 - Nomenclatura de testes na linguagem Java.



```
AuthenticationResourceTest.java
 1  package br.edu.uepb.campusviiramap.resources;
 2
 3⊕ import static com.mongodb.client.model.Filters.and;
17
18  public class AuthenticationResourceTest {
19
20      private static Document credentials;
21
23⊕     public static void setUpBeforeClass() throws Exception {
31
33⊕     public static void tearDownAfterClass() throws Exception {
39
41⊕     public void shouldAuthenticateUser() {
52
53  }
```

Fonte: própria, 2018.

4.2.4.1 Testes de Usabilidade

Durante a etapa de implementação da YP foram criados testes de usabilidade para testar manualmente a funcionalidade do Scanner de QR Code. Isto foi realizado devido à utilização de bibliotecas de código externo ao *Android* para a implementação do scanner, que justamente por não serem parte do *framework Android*, não podiam ser testadas via código de testes.

Nestes testes de aceitação assume-se o papel de usuário do sistema para executar as diferentes tarefas da aplicação, buscando que a funcionalidade do Scanner de QR Code funciona como o esperado. O quadro 9 apresenta o teste de usabilidade referente ao primeiro teste de aceitação da *User Story* 04. Os demais testes de aceitação criados encontra-se no APÊNDICE I. Os *prints* das telas com os resultados destes testes são mostrados na sessão 4.3.6.

Quadro 9 - Teste de Usabilidade: Atividade 01.

Atividade 01	Localizar sala de aula **correta** através da função "LOCALIZAR SALA" das disciplinas
Roteiro de atividades:	1. Abrir o aplicativo; 2. Informar número de matrícula e senha na tela de login; 3. Clicar no botão "Entrar" para acessar o aplicativo; 4. No aplicativo, selecionar a aba com o nome RDM; 5. Selecionar a opção "Localizar Sala" em uma disciplina mostrada na tela do RDM; 6. Posicionar o Scanner de maneira que o QR Code fique enquadrado no espaço de leitura; 7. Constar que a caixa de diálogo na tela mostra a mensagem de que a sala está correta.
Resultado	Esperado: uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está correta.

Fonte: própria, 2018.

4.2.5 Reunião de Acompanhamento

As reuniões de acompanhamento foram realizadas ao término de cada iteração. Durante as mesmas foram coletadas métricas que indicaram o progresso do projeto, bem como observações sobre o que aconteceu em cada iteração. Estas informações foram montadas no *Big Chart* e do mesmo foi gerado um gráfico *burnup*. O quadro 10 mostra o big chart com a coleta das métricas.

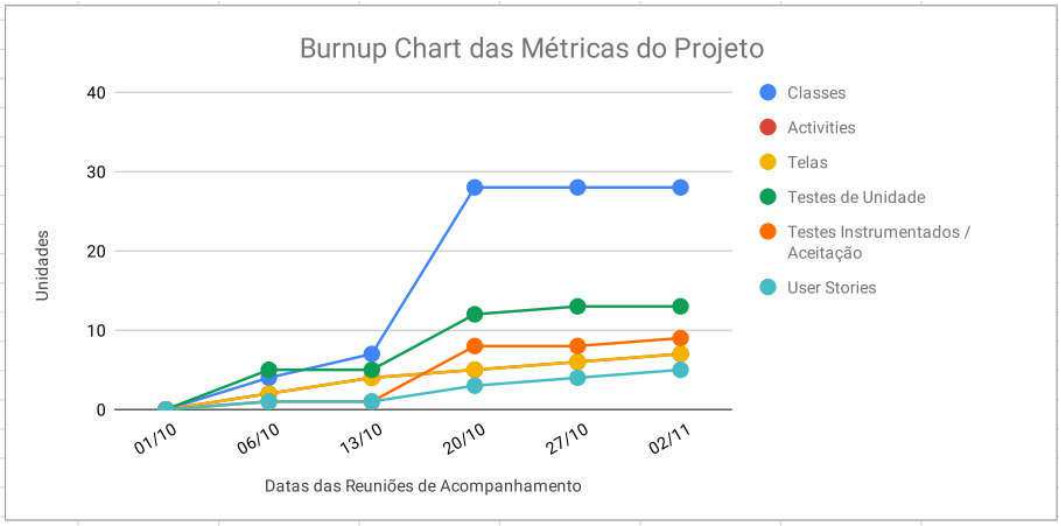
Quadro 10 - Big Chart.

Data	Classes	Activities	Telas	Testes de Unidade	Testes Instrumentados / Aceitação	*User Stories*	Observações
01/10	0	0	0	0	0	0	Nada a declarar
06/10	4	2	2	5	1	1	Nada a declarar
13/10	7	4	4	5	1	1	Houveram atividades que não foram concluídas durante esta iteração. Portanto a User Story relacionada a ela não foi consierada concluída. As atividades em questão foram movidas para a iteração seguinte.
20/10	28	5	5	12	8	3	Não foram implementados testes para tratar dos Testes de Aceitação da User Story 03 (Horário de Curso) vistos que as os testes realizados para a User Story 02 (RDM) já cobriam as funcionalidades.
27/10	28	6	6	13	8	4	Todas as funções restantes relacionadas ao Scanner de QR Code foram finalizadas nesta iteração. Entretanto, os Testes de Aceitação destas funcionalidades não puderam ser implementados pois se utilizou código de bibliotecas externas ao Android, logo não se podia testá-las. Para não deixar estas funcionalidades sem teste adequado, elas foram testadas via Testes de Usabilidade
02/11	28	7	7	13	9	5	Versão final produzida

Fonte: própria, 2018.

O gráfico 2 mostra o gráfico burnup gerado a partir dos dados do big chart.

Gráfico 2 - Burnup Chart.



Fonte: própria, 2018.

Foi criado também um quadro de análise de riscos. Os riscos foram coletados durante o desenvolvimento e postos no quadro tão logo foram notados. O quadro 11 mostra a análise de riscos.

Quadro 11 - Análise de Riscos.

Data de surgimento do risco	**Risco**	**Prioridade**	**Status**	**Providência / Solução**
06/10	Demora maior do que prevista implementação dos diferentes testes requeridos em uma aplicação Android	Alta	Superado	Estudar os diferentes tipos de testes de entender melhor suas características afim de realizar sua implementação mais rápido.
20/10	Para criar as funcionalidades do Scanner de QR Code serão utilizadas bibliotecas de código externo ao Android. Logo não haverá como realizar testes sobre este código.	Alta	Superado	Testar as funcionalidades do Scanner de QR Code através dos Testes de Usabilidade

Fonte: própria, 2018.

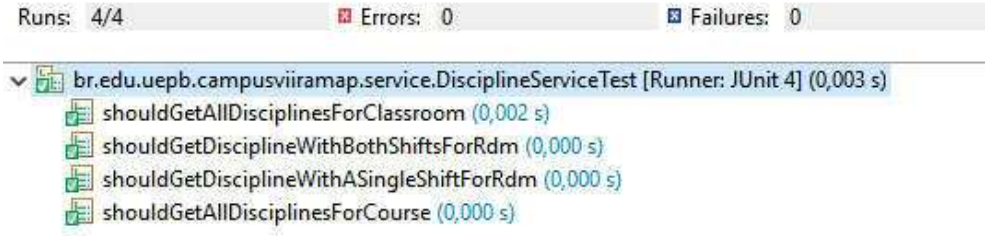
4.3 Quanto aos resultados dos testes

4.3.1 Testes do Web Service

Abaixo seguem os resultados de alguns testes do serviço web construído para alimentar a aplicação.

A figura 34 mostra os resultados dos testes da classe DisciplineService, responsável por recuperar os dados das disciplinas do banco de dados.

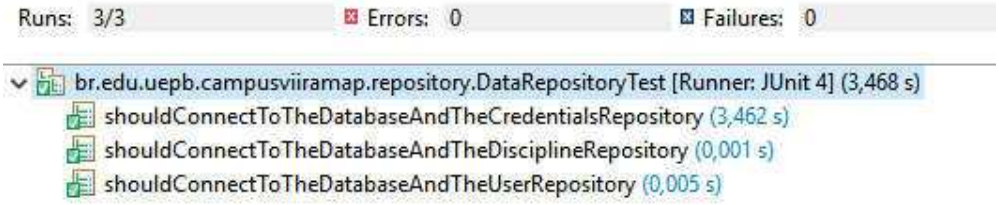
Figura 34 - Resultados de DisciplineServiceTest.



Fonte: própria, 2018.

A figura 35 mostra os resultados dos testes da classe DataRepository, responsável por conseguir o acesso ao repositório de banco de dados.

Figura 35 - Resultados de DataRepositoryTest.



Fonte: própria, 2018.

A figura 36 mostra os resultados dos testes da classe DisciplineResource, responsável por receber as requisições vindas do aplicativo e requisitar às classes de serviço que recuperem os dados necessários no banco de dados.

Figura 36 - Resultados de DisciplineResourceTest.

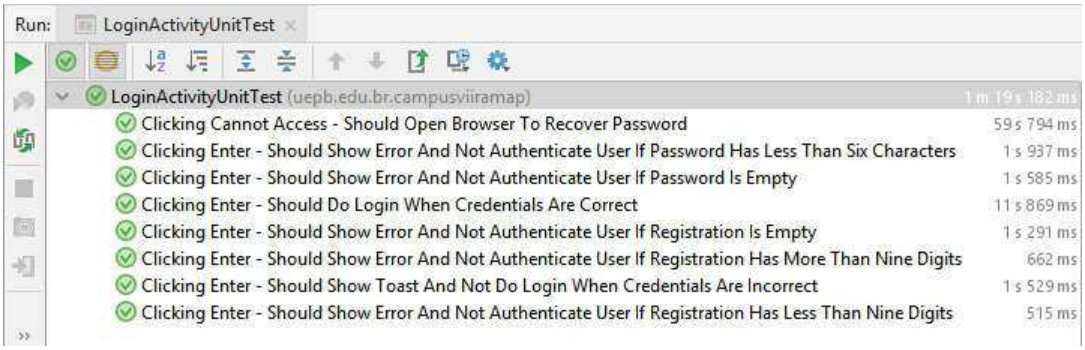
Fonte: própria, 2018.

4.3.2 Testes de Unidade do Android

Os testes seguintes referem-se ao aplicativo. Entretanto, estes testes de unidade não são rodados em um dispositivo *Android*. Eles são instanciados na JVM (*Java Virtual Machine*) da máquina na qual são rodados, assim como os testes do *Web Service* acima. Devido tais testes serem gerados em um ambiente de desenvolvimento *Android* (que geralmente não se comunica com a JVM) eles levam bem mais tempo do que outros testes para serem instanciados. Em cada um dos resultados pode-se notar na primeira linha que há grande diferença entre o primeiro resultado e os demais, justamente por causa da necessidade do *Android Studio IDE* fazer essa comunicação com a JVM.

A figura 37 mostra os resultados dos testes de unidade da classe LoginActivity, responsável pela parte lógica da tela de login do aplicativo.

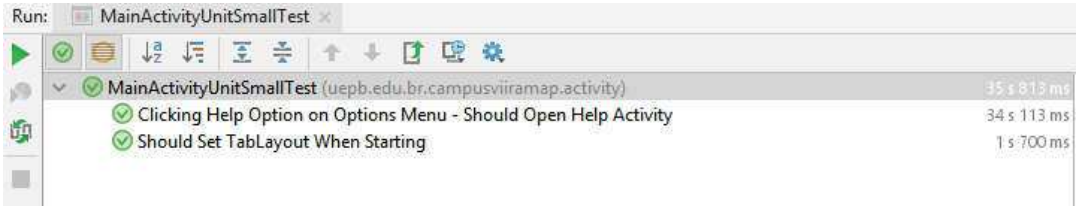
Figura 37 - Resultados de LoginActivityUnitTest.



Fonte: própria, 2018.

A figura 38 mostra os resultados dos testes de unidade da classe MainActivity, responsável pela parte lógica da tela principal do aplicativo.

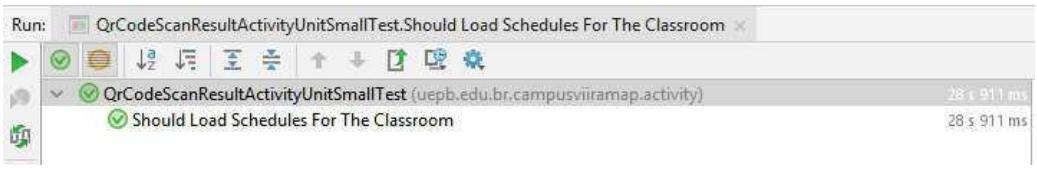
Figura 38 - Resultados de MainActivityUnitSmallTest.



Fonte: própria, 2018.

A figura 39 mostra os resultados dos testes de unidade da classe QrCodeScanResultActivity, responsável pela parte lógica da tela que mostra o resultado dos horários de uma sala após o QR Code da mesma ter sido escaneado.

Figura 39 - Resultados de QrCodeScanResultActivityUnitSmallTest.



Fonte: própria, 2018.

4.3.3 Dispositivo utilizado nos testes instrumentados

Todos os testes realizados deste ponto em diante são executados em um dispositivo *Android.* Portanto, aqui se informa as configurações de hardware do aparelho usado para os testes:

- Processador: Snapdradon 430 1.4GHz Octacore;
- Memória ROM: 32 GB;
- Memória RAM: 2 GM;
- Tela: 5,2”;
- Resolução: Full HD – 1920x1080.

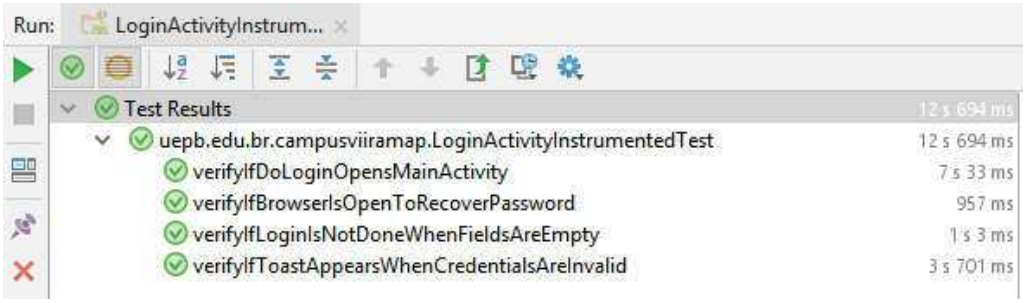
4.3.4 Testes instrumentados de nível médio

Os testes de nível médio são semelhantes aos testes de unidade, uma vez que testam componentes separados da aplicação. Entretanto, se nos testes de unidade testa-se se métodos estão se comportando da maneira esperada, nos testes instrumentados testa-se se os elementos

da interface se comportam da maneira esperada. Desta forma, os testes instrumentados testam o comportamento dos componentes do *Android* nas telas do aplicativo.

A figura 40 mostra os resultados dos testes instrumentados de nível médio da classe LoginActivity, neste caso testanto o comportamento dos elementos desta tela.

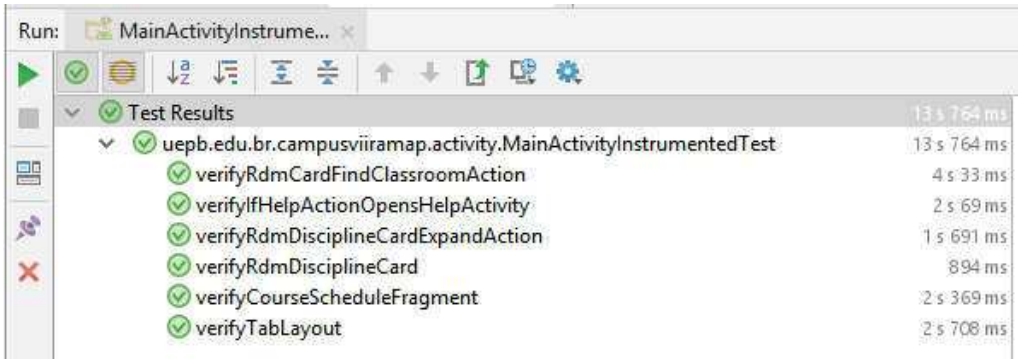
Figura 40 - Resultados de LoginActivityInstrumentedTest.



Fonte: própria, 2018.

A figura 41 mostra os resultados dos testes instrumentados de nível médio da classe MainActivity, neste caso testanto o comportamento dos elementos desta tela.

Figura 41 - Resultados de MainActivityInstrumentedTest.



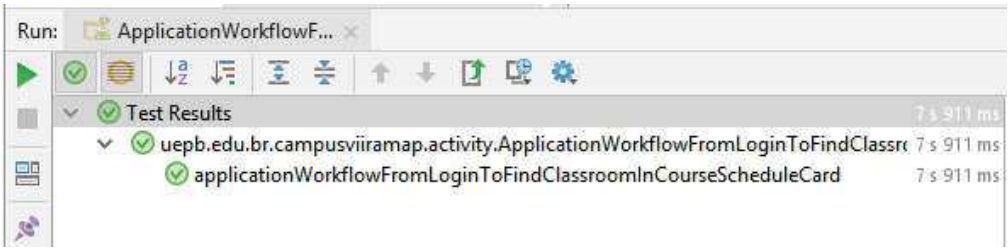
Fonte: própria, 2018.

4.3.5 Testes instrumentados de nível alto

Os testes instrumentados de nível alto (também chamados de testes de fluxo de trabalho) testam diferentes fluxos de execução do aplicativo. Ao contrário dos testes de nível médio onde componentes são testados em partes, aqui eles são testados interagindo uns com os outros, de maneira que os fluxos de execução de tarefas cruzam diferentes telas do aplicativo.

A figura 42 mostra os resultados do teste instrumentado de nível alto para o fluxo: da tela de *Login* até a tela do *Scanner de QR Code* acessado por um *card* de disciplina na tela de Horários do Curso.

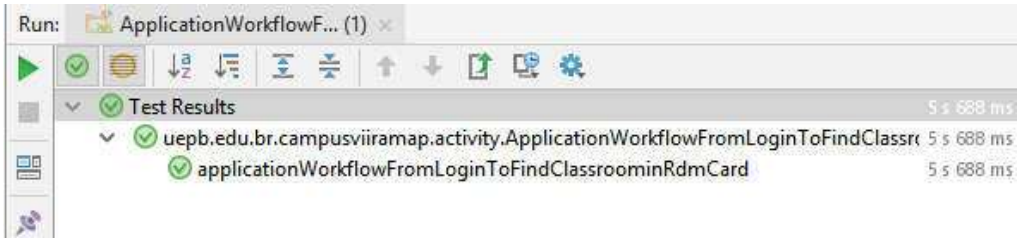
Figura 42 - Resultado de ApplicationWorkflowFromLoginToFindClassroomInCoourseScheduleCard.



Fonte: própria, 2018.

A figura 43 mostra o resultado do teste instrumentado de nível alto para o fluxo: da tela de *Login* até a tela do *Scanner de QR Code* acessado por um *card* de disciplina na tela de RDM.

Figura 43 - Resultado de ApplcationWorkflowFromLoginToFindClassroomInRdmCard.



Fonte: própria, 2018.

4.3.6 Testes de usabilidade

As telas à seguir são os resultados dos Testes de Usabilidade da seção 4.3.4.1 e do APÊNDICE I.

Deixa-se claro aqui, que todos os testes seguintes, com exceção do teste da atividade 06 foram realizados tendo em mente localizar a sala 105.

A figura 44 mostra o resultado do teste de usabilidade da atividade 01, no qual foi escaneado o QR Code da sala 105.

Figura 44 - Resultado da Atividade 01 dos Testes de Usabilidade.

Fonte: própria, 2018.

A figura 45 mostra o resultado do teste de usabilidade da atividade 02, no qual foi escaneado o QR Code da sala 104.

Figura 45 - Resultado da Atividade 02 dos Testes de Usabilidade.

Fonte: própria, 2018.

A figura 46 mostra o resultado do teste de usabilidade da atividade 03, no qual foi escaneado o QR Code da sala 106.

Figura 46 - Resultado da Atividade 03 dos Testes de Usabilidade.

Fonte: própria, 2018.

A figura 47 mostra o resultado do teste de usabilidade da atividade 04, no qual foi escaneado o QR Code da sala 205.

Figura 47 - Resultado da Atividade 04 dos Testes de Usabilidade.

Fonte: própria, 2018.

A figura 48 mostra o resultado do teste de usabilidade da atividade 05, no qual foi escaneado o QR Code da sala 408 – Laboratório de Informática.

Figura 48 - Resultado da Atividade 05 dos Testes de Usabilidade.

Fonte: própria, 2018.

A figura 49 mostra o resultado do teste de usabilidade da atividade 06, no qual foi escaneado o QR Code da sala 408 – Laboratório de Informática. Diferente das outras atividades, neste tentou-se não localizar a sala 408, mas sim obter a lista de horários em que esta sala é utilizada.

Figura 49 - Resultado da Atividade 06 dos Testes de Usabilidade.

Fonte: própria, 2018.

5 CONSIDERAÇÕES TEMPORÁRIAS E SUGESTÕES PARA TRABALHOS FUTUROS

O presente Capítulo apresenta as considerações temporárias deste trabalho, seguido das contribuições e limitações desta pesquisa, bem como sugestões de trabalhos futuros.

5.1 Considerações temporárias

Os resultados do processo de desenvolvimento possibilitaram ver através dos resultados dos testes, que o aplicativo criado mostrou-se eficiente. Não somente os testes dos componentes do *Android* mas também a integração com a tecnologia de QR Code, que permitiram cumprir os requisitos levantados para atacar a problemática.

A metodologia de desenvolvimento ágil YP foi extremamente útil por seu detalhamento, preocupado com o melhor desenvolvimento ágil. A divisão em etapas claras, somados ao controle de documentão proporcionam controle sobre diferentes visões do desenvolvimento do aplicativo.

5.2 Contribuições do trabalho

Diante do que foi apresentado, acredita-se que a principal contribuição deste trabalho é o aplicativo produzido, visto que o mesmo está completamente funcional, apresenta bom desempenho e usabilidade, e cumpre com os requisitos levantados para servir de apoio à problemática levantada.

5.3 Limitações do desenvolvimento

Devido a UEPB não possuir um serviço web que possibilitasse recuperar os dados oficiais dos estudantes e das disciplinas dos cursos do Campus VII, fez-se necessária a criação de um serviço web que espelhasse os dados contidos naquele sistema. Tal abordagem acrescentou um trabalho que seria desnecessário caso a universidade permitisse o acesso aos dados de alguma forma, o que contribuiria também para uma maior confiabilidade na aplicação.

5.4 Trabalhos futuros

Entre as possibilidades, destacam-se:

- Realizar uma refatoração completa do código da aplicação, produzindo documentação e em seguida tornando o código aberto no *GitHub*;
- Realizar estudo sobre tecnologias tais como o GPS Indoor que possam ser integradas para melhorar a aplicação.

6 REFERÊNCIAS

AVRAM, Abel. **Kotlin é agora uma linguagem oficial no Android**. Infoq, 2017. Disponível em: < https://www.infoq.com/br/news/2017/06/android-kotlin>. Acesso em: 19 mai. 2018.

DESENVOLVIMENTO de software móvel. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2017. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Desenvolvimento_de_software_m%C3%B3vel&oldid=49623957>. Acesso em: 18 mai. 2018.

FERREIRA, Rodrigo. **QR Code: Possibilidades de aplicação para a sua empresa**. Café com Galo, 2013. Disponível em: < http://www.cafecomgalo.com.br/qr-code-possibilidades-de-aplicacao-para-a-sua-empresa/>. Acesso em: 20 mai. 2018.

GARCIA, Francilene Procópio, et al. **easYProcess: um processo de desenvolvimento de software para uso no ambiente acadêmico**. Campina Grande, 2007. 8 p. Disponível em: < http://www.dsc.ufcg.edu.br/~pet/Artigos/ARTIGO_YP.pdf>. Acesso em: 20 mai. 2018.

IBGE. **PNAD 2015: 19,7% dos domicílios com TV necessitam adequação para receber sinal digital, em 2013 eram 28,5%**. Agência de notícias | IBGE – Instituto Brasileiro de Geografia e Estatística, 2016. Disponível em: < https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/9454-pnad-2015-19-7-dos-domicilios-com-tv-necessitam-adequacao-para-receber-sinal-digital-em-2013-eram-28-5>. Acesso em: 20 mai. 2018.

IDC. **IDC Releases**: Após dois anos, mercado de smartphones cresce em 2017 e atinge o segundo melhor desempenho de vendas. IDC – International Data Corporation, 2018. Disponível em: <http://br.idclatin.com/releases/news.aspx?id=2312>. Acesso em: 20 mai. 2018.

MEYER, Maximiliano. **A história do Android**. Oficina da Net, 2015-2018. Disponível em: <https://www.oficinadanet.com.br/post/13939-a-historia-do-android>. Acesso em: 20 mai. 2018.

MITRUT, Wellington. **Como Kotlin se tornou a nossa linguagem principal para Android**. Medium, 2017. Disponível em: <https://medium.com/blog-do-mitrut/como-kotlin-se-tornou-a-nossa-linguagem-principal-para-android-24c9492fa273>. Acesso em: 20 mai. 2018.

ORACLE. **Java SE, Oracle Technology** Network. ORACLE, 2018. Disponível em: <http://www.oracle.com/technetwork/java/javase/overview/index.html>. Acesso em: 22 mai. 2018.

PENA, Rodolfo Alves. **Instrumentos de Localização**. Escola Kids. Disponível em <https://escolakids.uol.com.br/instrumentos-de-localizacao.htm>. Acesso em 20 mai. 2018.

PIMENTEL, Igor. **Interface Digital e Design de Totem Interativo**. Rio de Janeiro, 2006-2018. Disponível em: <https://www.behance.net/gallery/31312939/Interface-digital-e-design-de-totem-interativo>. Acesso em: 23 mai. 2018.

PRASS, Ronaldo. **Entenda o que são os 'QR Codes', códigos lidos pelos celulares**. G1, 2011. Disponível em: < http://g1.globo.com/tecnologia/noticia/2011/05/entenda-o-que-sao-os-qr-codes-codigos-lidos-pelos-celulares.html>. Acesso em: 19 mai. 2018.

PROGRAMA DE EDUCAÇÃO TUTORIAL (PET) **– EasyProcess – Um Processo de Desenvolvimento de Software –** Campina Grande: 2007. 86 p.

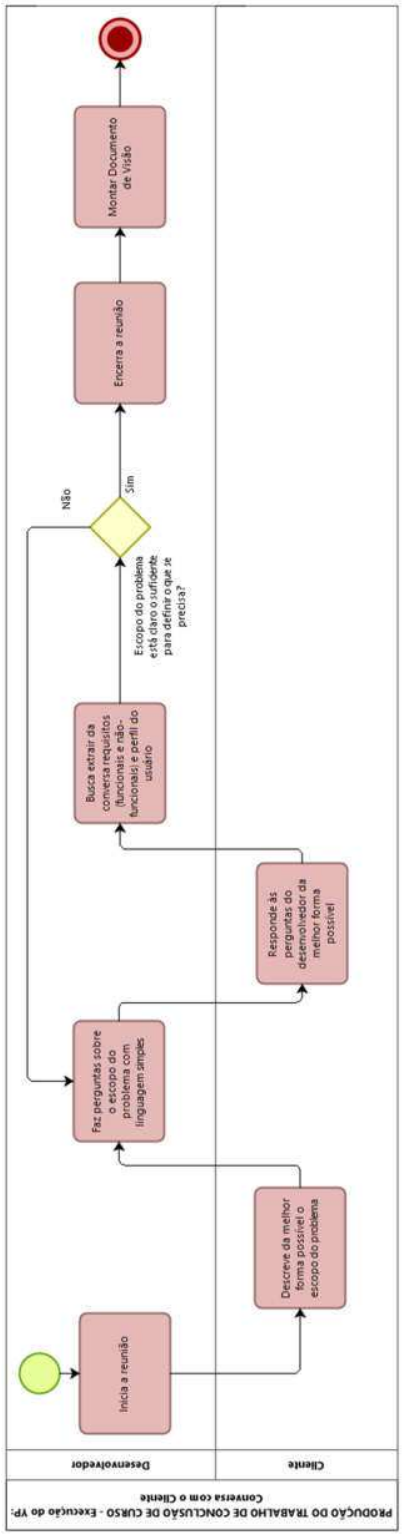
SCUDERO, Erick. "Desenvolvimento mobile multiplataforma ou nativo, qual é o melhor?; Becode. Disponível em< https://becode.com.br/desenvolvimento-mobile-multiplataforma-ou-nativo/>. Acesso em 19 de maio de 2018.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª. ed. São Paulo: Pearson Education, 2011. 544 p.

SOUSA, Ana Sofia de. **Uso do QR Code no Marketing Digital:** a perspectiva do utilizador português. Dissertação de Mestrado (Marketing Digital) - Instituto Superior de Contabilidade e Administração do Porto, Instituto Politécnico do Porto. Porto. 2014. Disponível em: <http://recipp.ipp.pt/bitstream/10400.22/5637/1/DMAnaSousa_2014.pdf>. Acesso em 22 mai. 2018.

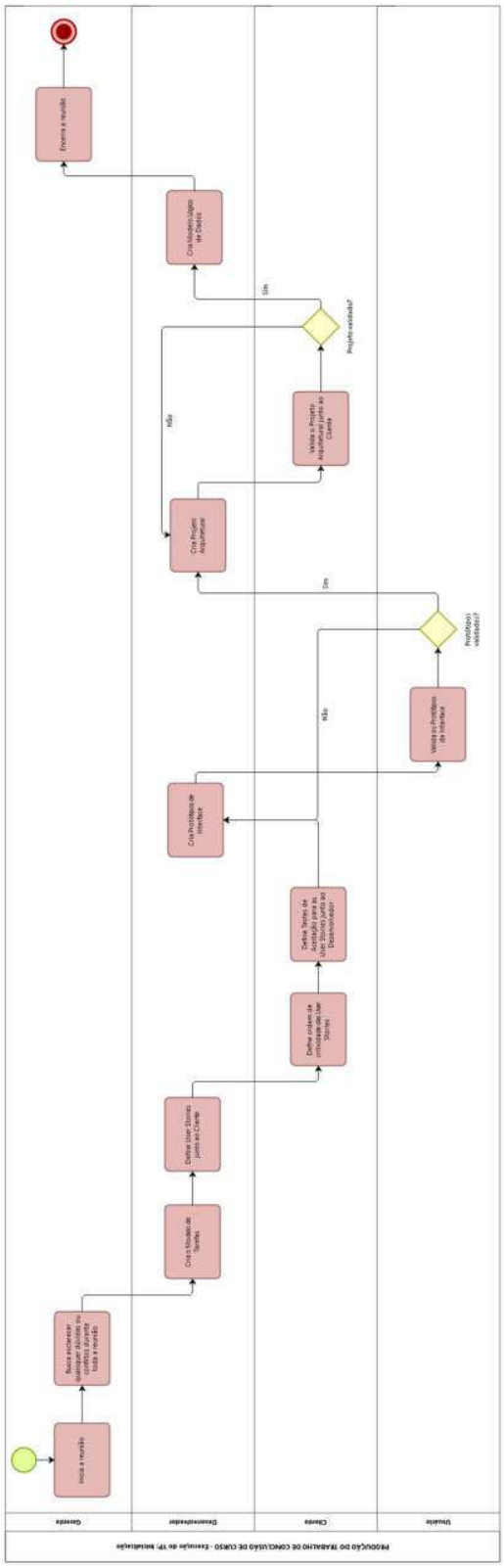
W3C. **Extensible Markup Language (XML) 1.0 (Figth Edition)**. W3C, 2008. Disponível em: <https://www.w3.org/TR/REC-xml/>. Acesso em: 22 mai. 2018.

APÊNDICE A – Fluxograma da Execução da YP: Conversa com o Cliente



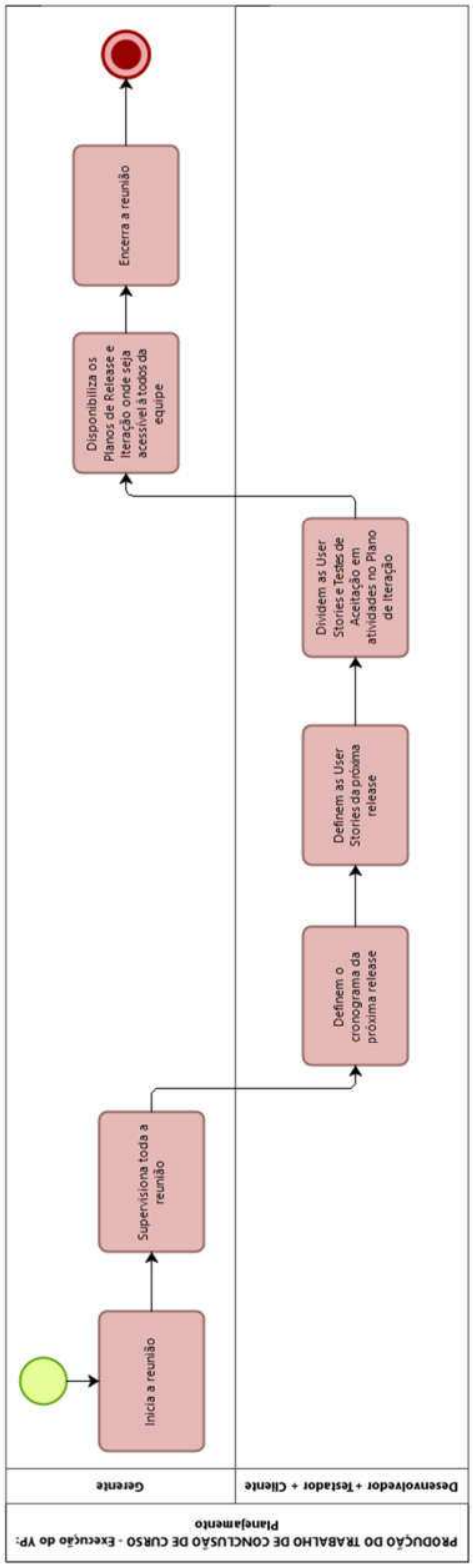
Fonte: própria, modelado com Bizagi Modeler, 2018.

APÊNDICE B – Fluxograma da Execução da YP: Inicialização



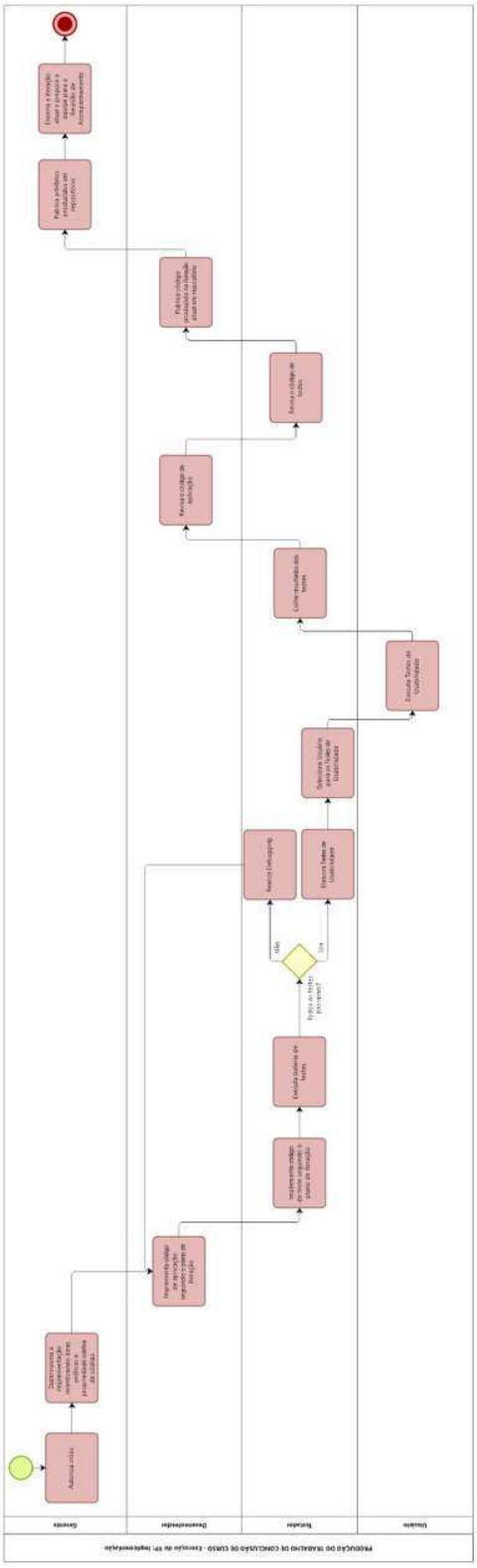
Fonte: própria, modelado com Bizagi Modeler, 2018.

APÊNDICE C – Fluxograma da Execução da YP: Planejamento



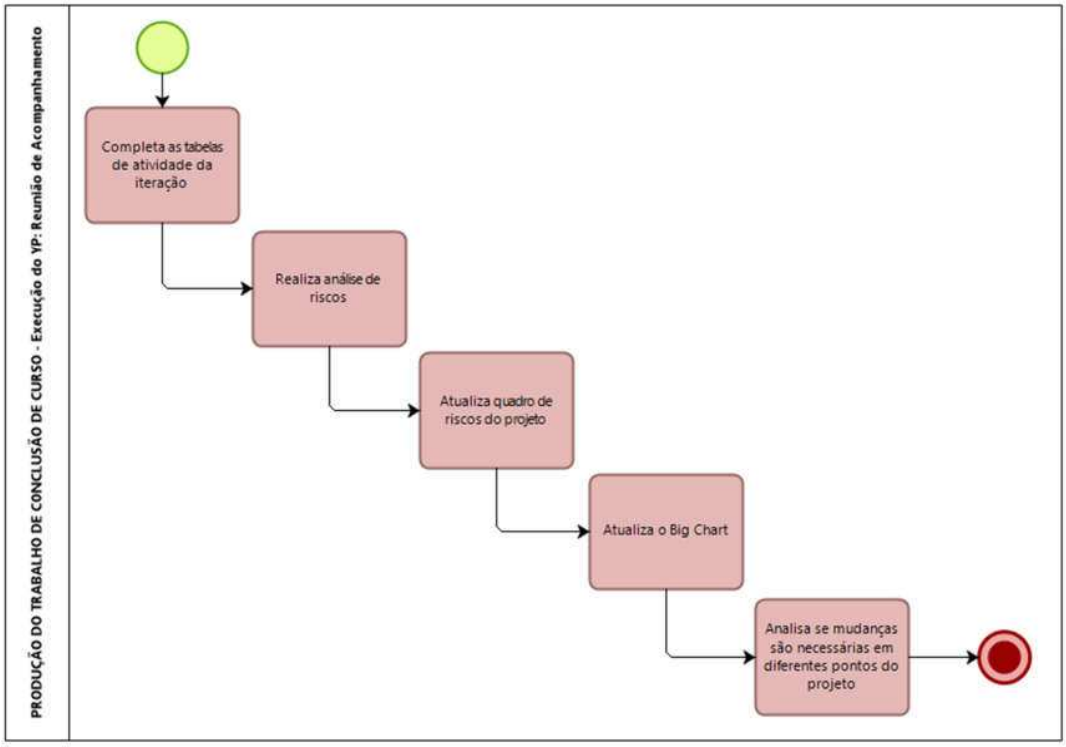
Fonte: própria, modelado com Bizagi Modeler, 2018.

APÊNDICE D – Fluxograma da Execução da YP: Implementação



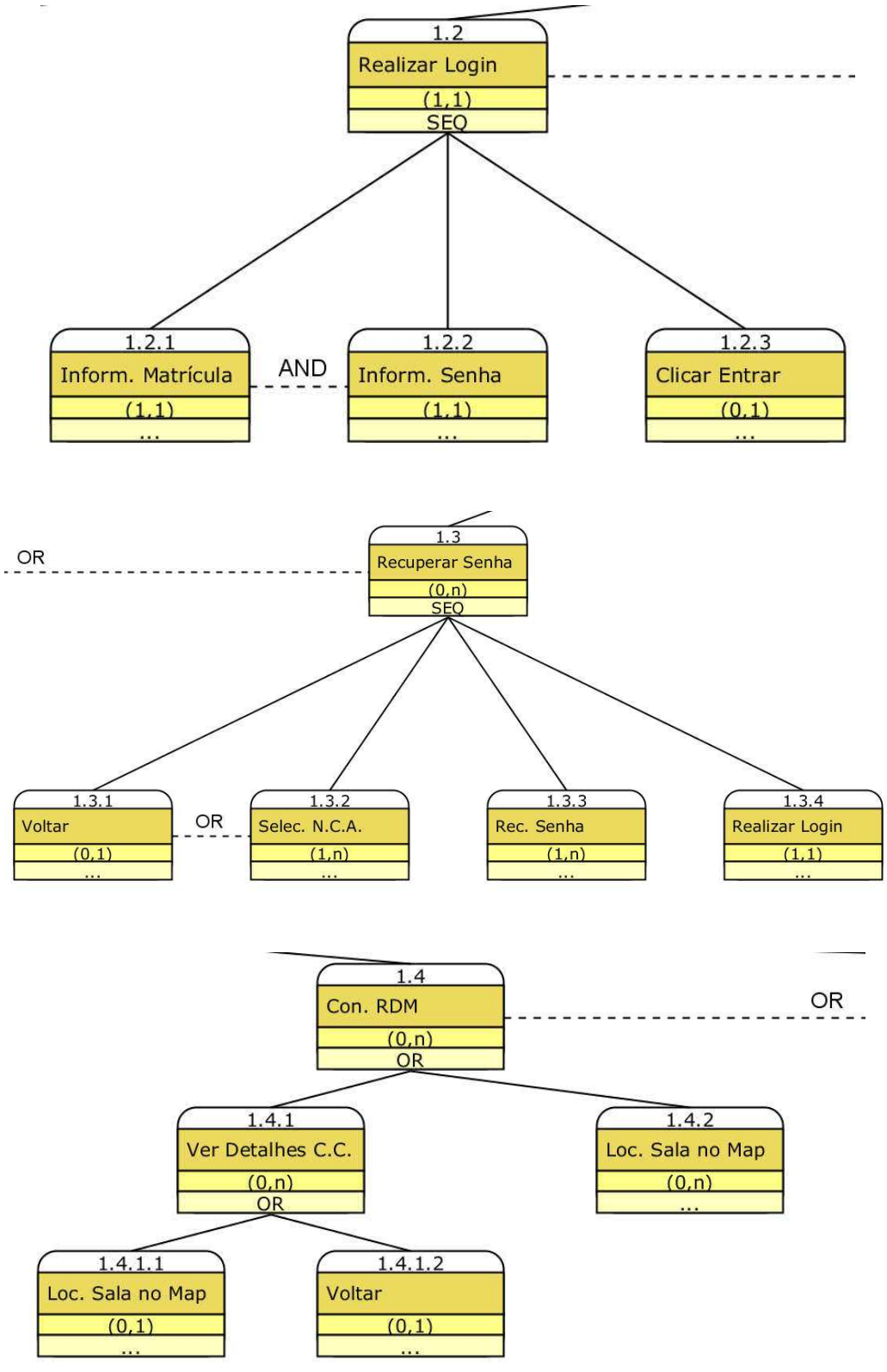
Fonte: própria, modelado com Bizagi Modeler, 2018.

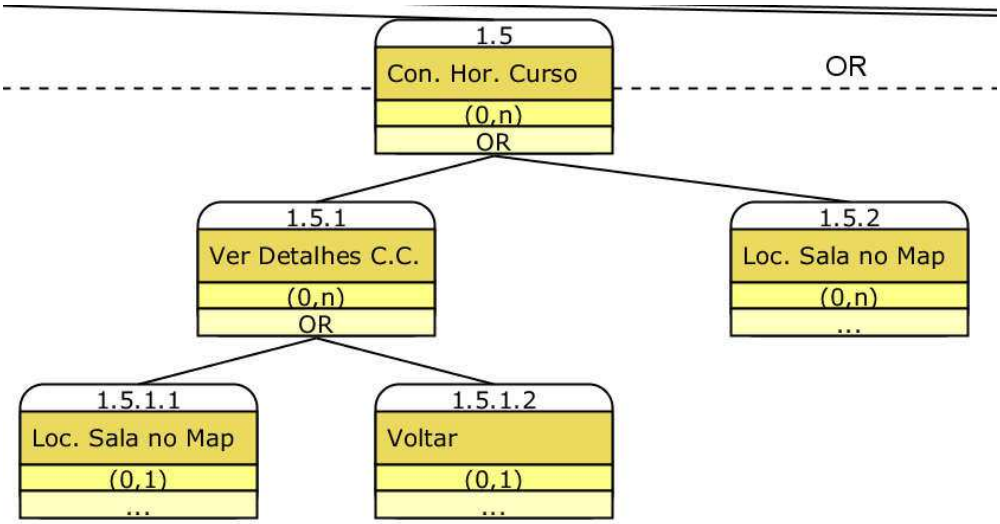
APÊNDICE E – Fluxograma da Execução da YP: Reunião de Acompanhamento

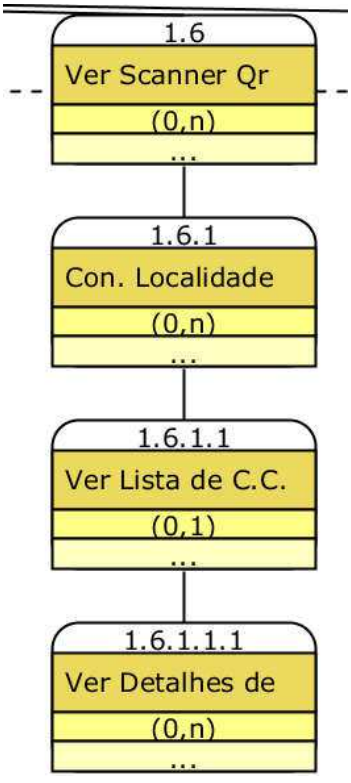


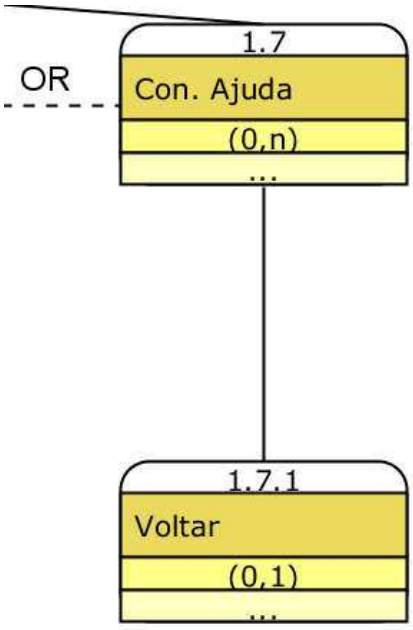
Fonte: própria, modelado com Bizagi Modeler, 2018.

APÊNDICE F – Modelos de Tarefas da aplicação




1.5
Con. Hor. Curso
(0,n)
OR
OR
1.5.1
Ver Detalhes C.C.
(0,n)
OR
1.5.2
Loc. Sala no Map
(0,n)
...
1.5.1.1
Loc. Sala no Map
(0,1)
...
1.5.1.2
Voltar
(0,1)
...


1.6
Ver Scanner Qr
(0,n)
...
1.6.1
Con. Localidade
(0,n)
...
1.6.1.1
Ver Lista de C.C.
(0,1)
...
1.6.1.1.1
Ver Detalhes de
(0,n)
...



Fonte: própria, modelado com iTAOS, 2018.

APÊNDICE G – Planos de *Release*

Release 02: 15/10 – 27/10		
Iteração	***User Story***	**Período**
Iteração 03	US02, US03	15/10 – 20/10
Iteração 04	US04	22/10 – 27/10

Release 03: 29/10 – 02/11		
Iteração	***User Story***	**Período**
Iteração 05	US05	29/10 – 02/11

Fonte: própria, 2018.

APÊNDICE H – Planos de Iteração

Iteração 02 – (08/10/2018 – 13/10/2018)				
US02 - Implementar a funcionalidade do RDM				
Testes de Aceitação				***Status***
TA02.1	Expandir uma disciplina através da função "EXPANDIR" para visualizar todos os detalhes sobre a mesma (a disciplina se expande tomando toda a tela, mostrando os detalhes contidos no RDM).			D
TA02.2	Localizar a sala de uma disciplina através da função "LOCALIZAR SALA". (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.			D
TA02.3	Localizar a sala de uma disciplina através da função "LOCALIZAR SALA" dentro da telha de detalhes da disciplina. (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.			D
Atividade	**Descrição**	**Estimativa de Tempo**	**Tempo Real**	***Status***
A02.1	Implementar a GUI da aba do RDM	1h30	1h	C
A02.2	Implementar a aba do RDM na Activity principal	1h	1h40	C
A02.3	Implementar a GUI das disciplinas no RDM	2h15	3h25	C
A02.4	Implementar Cards de disciplinas no RDM	2h15	5h	C
A02.5	Implementar Activity de detalhes de disciplina	2h30	2h30	C
A02.6	Implementar Activity de Scanner de QR Code	3h	8h50	C
A02.7	Implementar recursos no Web Service relacionados ao usuário e às disciplinas.	3h	-	D
A02.8	Implementar testes para os recursos do Web Service	4h	-	D
A02.9	Implementar códigos dos TA	5h30	-	D

Iteração 03 – (15/10/2018 – 20/10/2018)
US02 - Implementar a funcionalidade do RDM; **US03 - Implementar a funcionalidade do Horários do Curso**

Testes de Aceitação		*Status*
TA02.1	Expandir uma disciplina através da função “EXPANDIR” para visualizar todos os detalhes sobre a mesma (a disciplina se expande tomando toda a tela, mostrando os detalhes contidos no RDM).	C
TA02.2	Localizar a sala de uma disciplina através da função “LOCALIZAR SALA”. (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.	C
TA02.3	Localizar a sala de uma disciplina através da função “LOCALIZAR SALA” dentro da telha de detalhes da disciplina. (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.	C
TA03.1	Expandir uma disciplina através da função “EXPANDIR” para visualizar todos os detalhes sobre a mesma (a disciplina se expande tomando toda a tela, mostrando os detalhes contidos no Horário).	C
TA03.2	Localizar a sala de uma disciplina através da função “LOCALIZAR SALA”. (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.	C
TA03.3	Localizar a sala de uma disciplina através da função “LOCALIZAR SALA” dentro da telha de detalhes da disciplina. (Deve abrir o scanner de QR Code para poder orientar o usuário a encontrar a sala.	C

Atividade	**Descrição**	**Estimativa de Tempo**	**Tempo Real**	***Status***
A02.7	Implementar recursos no Web Service relacionados ao usuário e às disciplinas.	3h	5h	C
A02.8	Implementar testes para os recursos do Web Service	4h	3h	C
A02.9	Implementar códigos dos TA	5h30	7h	C
A03.1	Implementar a GUI da aba do Horário do Curso	1h30	45m	C
A03.2	Implementar a aba do Horário do Curso na Activity principal	1h	30m	C
A03.3	Implementar códigos dos TA	-	-	C

Iteração 04 – (22/10/2018 – 27/10/2018)	
US04 - Implementar as funcionalidade de Scanner de QR Code	
Testes de Aceitação	***Status***

TA04.1	Localizar sala de aula **correta** através da função “LOCALIZAR SALA” das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está correta).	C
TA04.2	Localizar sala de aula **incorreta** que fica à esquerda da correta através da função "LOCALIZAR SALA" das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também à quantas salas à esquerda fica a sala correta).	C
TA04.3	Localizar sala de aula **incorreta** que fica à direita da correta através da função "LOCALIZAR SALA" das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também à quantas salas à direita fica a sala correta).	C
TA04.4	Localizar sala de aula **incorreta** que fica no grupo de salas oposto à sala correta através da função "LOCALIZAR SALA" das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também que o usuário deve procurar nas salas opostas à esta).	C
TA04.5	Localizar sala de aula **incorreta** que fica no grupo de salas de outro bloco, não pertencente à sala correta através da função "LOCALIZAR SALA" das disciplinas (uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também que o usuário deve procurar em outro bloco de salas do Campus).	C
TA04.6	Consultar uma sala de aula através de um QR Code para visualizar seus horários de uso (de acordo com o QR Code da sala, as informações e horários de funcionamento da mesma devem ser apresentadas ao usuário).	C

Atividade	**Descrição**	**Estimativa de Tempo**	**Tempo Real**	***Status***
A04.1	Implementar a GUI da aba do Scanner de QR Code	3h	40m	C
A04.2	Implementar a aba do Scanner de QR Code na Activity principal	3h	4h50	C
A04.3	Implementar códigos dos TA	-	-	C

Iteração 05 – (29/10/2018 – 02/11/2018)				
US05 – Implementar a Ajuda do aplicativo				
Testes de Aceitação				***Status***
TA05.1	Consultar tela de Ajuda (a tela de Ajuda deve ser aberta quando o usuário selecionar o botão de ajuda na barra do aplicativo).			C
Atividade	**Descrição**	**Estimativa de Tempo**	**Tempo Real**	***Status***

A04.1	Implementar a GUI da tela de Ajuda	3h20	3h	C
A04.2	Implementar a ação de Ajuda acessada através da barra do aplicativo na Activity principal	10m	10m	C
A04.3	Implementar códigos dos TA	30m	15m	C

Fonte: própria, 2018.

APÊNDICE I – Testes de Usabilidade

Atividade 02	Localizar sala de aula **incorreta** que fica à direita da correta através da função "LOCALIZAR SALA" das disciplinas
Roteiro de atividades:	1. Abrir o aplicativo; 2. Informar número de matrícula e senha na tela de login; 3. Clicar no botão "Entrar" para acessar o aplicativo; 4. No aplicativo, selecionar a aba com o nome RDM; 5. Selecionar a opção "Localizar Sala" em uma disciplina mostrada na tela do RDM; 6. Posicionar o Scanner de maneira que o QR Code fique enquadrado no espaço de leitura; 7. Constar que a caixa de diálogo na tela mostra a mensagem de que a sala está incorreta e à quantas portas para a direita fica a sala correta.
Resultado	Esperado: uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também à quantas salas à direita fica a sala correta.

Atividade 03	Localizar sala de aula **incorreta** que fica à esquerda da correta através da função "LOCALIZAR SALA" das disciplinas
Roteiro de atividades:	1. Abrir o aplicativo; 2. Informar número de matrícula e senha na tela de login; 3. Clicar no botão "Entrar" para acessar o aplicativo; 4. No aplicativo, selecionar a aba com o nome RDM; 5. Selecionar a opção "Localizar Sala" em uma disciplina mostrada na tela do RDM; 6. Posicionar o Scanner de maneira que o QR Code fique enquadrado no espaço de leitura; 7. Constar que a caixa de diálogo na tela mostra a mensagem de que a sala está incorreta e à quantas portas para a esquerda fica a sala correta.
Resultado	Esperado: uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também à quantas salas à esquerda fica a sala correta.

Atividade 04	Localizar sala de aula **incorreta** que fica no grupo de salas oposto à sala correta através da função "LOCALIZAR SALA" das disciplinas
Roteiro de atividades:	1. Abrir o aplicativo; 2. Informar número de matrícula e senha na tela de login; 3. Clicar no botão "Entrar" para acessar o aplicativo; 4. No aplicativo, selecionar a aba com o nome RDM; 5. Selecionar a opção "Localizar Sala" em uma disciplina mostrada na tela do RDM; 6. Posicionar o Scanner de maneira que o QR Code fique enquadrado no espaço de leitura; 7. Constar que a caixa de diálogo na tela mostra a mensagem de que a sala não foi encontrada e deve-se procurar nas salas opostas.

Resultado	Esperado: uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também que o usuário deve procurar nas salas opostas à esta.

Atividade 05	Localizar sala de aula **incorreta** que fica à esquerda da correta através da função "LOCALIZAR SALA" das disciplinas
Roteiro de atividades:	1. Abrir o aplicativo; 2. Informar número de matrícula e senha na tela de login; 3. Clicar no botão “Entrar” para acessar o aplicativo; 4. No aplicativo, selecionar a aba com o nome RDM; 5. Selecionar a opção “Localizar Sala” em uma disciplina mostrada na tela do RDM; 6. Posicionar o Scanner de maneira que o QR Code fique enquadrado no espaço de leitura; 7. Constar que a caixa de diálogo na tela mostra a mensagem de que a sala não foi encontrada e que deve-se procurar em outro bloco de salas.
Resultado	Esperado: uma caixa de diálogo deve aparecer na tela indicando que a sala buscada está incorreta e também que o usuário deve procurar em outro bloco de salas do Campus.

Atividade 06	Consultar uma sala de aula através de um QR Code para visualizar seus horários de uso (de acordo com o QR Code da sala, as informações e horários de funcionamento da mesma devem ser apresentadas ao usuário).
Roteiro de atividades:	1. Abrir o aplicativo; 2. Informar número de matrícula e senha na tela de login; 3. Clicar no botão “Entrar” para acessar o aplicativo; 4. No aplicativo, selecionar a aba com o nome Scanner de QR Code; 5. Posicionar o Scanner de maneira que o QR Code fique enquadrado no espaço de leitura; 6. Constar que a tela que aparece mostra uma mensagem dizendo qual sala foi busca e mostra uma lista dos horários desta sala.
Resultado	Esperado: de acordo com o marcador de cada local, as informações e horários de funcionamento daquela localidade devem ser apresentadas ao usuário.

Fonte: própria, 2018.