CENTRO PAULA SOUZA

**FACULDADE DE TECNOLOGIA DE AMERICANA**

**Curso Análise e Desenvolvimento de Sistemas**

Leonardo de Oliveira Toledo

# Projeto de aplicativo e site para auxílio de consulta de linhas urbanas de transporte público em Sumaré-SP

**Americana, SP**

**2015**

CENTRO PAULA SOUZA

**FACULDADE DE TECNOLOGIA DE AMERICANA**

**Curso Análise e Desenvolvimento de Sistemas**

Leonardo de Oliveira Toledo

# Projeto de aplicativo e site para auxílio de consulta de linhas urbanas de transporte público em Sumaré-SP

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Analise e Desenvolvimento de Sistemas, sob a orientação do Prof. Anderson Luiz Barbosa

Área de concentração: Engenharia de Software.

**Americana, SP**

**2015**

Toledo, Leonardo de Oliveira
T583p Projeto de aplicativo e site para auxílio de consulta de linhas urbanas de transporte público em Sumaré - SP. / Leonardo de Oliveira Toledo. – Americana: 2015.
82f.

Monografia (Graduação em Tecnologia em Análise e Desenvolvimento de Sistemas). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Me. Anderson Luiz Barbosa

1. Dispositivos móveis – aplicativos 2. Transporte urbano I. Barbosa, Anderson Luís II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.519
656.5

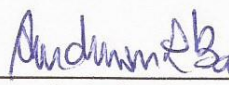
Leonardo de Oliveira Toledo

# Projeto de aplicativo e site para auxílio de consulta de linhas urbanas de transporte público em Sumaré - SP

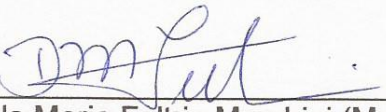
Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – Fatec/ Americana.
Área de concentração: Engenharia de Software

Americana, 10 de dezembro de 2015.

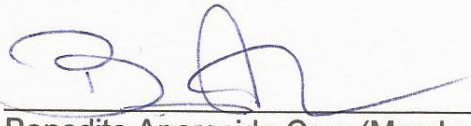
**Banca Examinadora:**



Anderson Luiz Barbosa (Presidente)
Mestre
Faculdade de Tecnologia de Americana - FATEC



Daniela Maria Feltrin Marchini (Membro)
Mestre
Faculdade de Tecnologia de Americana - FATEC



Benedito Aparecido Cruz (Membro)
Mestre
Faculdade de Tecnologia de Americana - FATEC

Aos meus pais, irmã, meus avós, minha namorada, a toda minha família e a todos os amigos que me apoiaram neste trabalho.

## AGRADECIMENTOS

A Faculdade de Tecnologia de Americana, pela oportunidade de fazer o curso.

Ao meu orientador, Anderson Luiz Barbosa, pela ajuda, correções e incentivos.

Aos meus pais, namorada e família, pelo incentivo e apoio.

Ao todos os meus amigos, em especial a Luram Archanjo, pela ajuda e troca de experiência durante todo o curso e elaboração deste trabalho.

A estes e a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

## RESUMO

Hoje em dia o ônibus é um dos principais meios de transporte, e através dele milhares de pessoas se locomovem todos os dias para o trabalho, escola, entre outros. Visto que na cidade de Sumaré – SP não existe meio eletrônico para consulta de horários das linhas urbanas de ônibus, e sabendo que os *smartphones* são cada vez mais usados pela população, o objetivo deste trabalho foi desenvolver um projeto de aplicativo e site para providenciar a consulta dos horários por meio eletrônico para a população, que ultrapassa os 240.000 habitantes. Para o desenvolvimento do projeto foram utilizados os diagramas da UML e DER e conceitos de engenharia de software. Os objetivos foram alcançados, todas as funções esperadas foram acrescentadas ao projeto, a documentação foi concluída o projeto pronto para a etapa de desenvolvimento.



**Palavras Chave:** Projeto; Diagramas; Ônibus; Aplicativo; Site.

***ABSTRACT***

*Today the bus is one of the main means of transport, and through it, thousands of people get to work, school, among others ways, every day. In the city of Sumaré - SP there isn't any electronic way to consult schedules of urban bus lines. Knowing that smartphones are increasingly used by the population, the aim of this study was to develop an application project and website to provide the consultation of urban bus lines schedules by electronic means for the population, which exceeds 240,000 inhabitants. To develop the project we used the UML diagrams, DER and concepts of software engineering. The objectives were achieved, all the expected functions have been added to the project, the documentation was completed and the project is ready for development stage.*

***Keywords:*** *Project; Diagrams; Bus; Application; Site.*

## LISTA DE FIGURAS

**SUMÁRIO**

## INTRODUÇÃO

A mobilidade urbana por ônibus é muito importante para as cidades do Brasil, pois é através desse meio de locomoção que grande parte da população vai para a escola, ao trabalho e compromissos em geral, sendo de uso diário e fundamental para muitas das pessoas. Entretanto, há muitos problemas com esse meio de transporte, como tempo de viagem, acessibilidade, lotação, entre outros, sendo um dos problemas que mais causa desconforto entre os usuários a pontualidade. Atrasos de em média cinco minutos não costumam causar muito incomodo, porém, atrasos maiores são motivo de muita insatisfação. Outro motivo relacionado à pontualidade que causa muito transtorno é que casualmente certas linhas de ônibus não passam nos pontos pois o veículo quebra ou o motorista decide descansar ao invés de fazer sua viagem.

Após realizar uma pesquisa na cidade de Sumaré-SP, foi notado que a empresa de viação que realiza o transporte na cidade não disponibiliza os horários das linhas urbanas em seu site, sendo através de telefone o único modo de consultar os horários, modo pouco funcional, pois é difícil encontrar o número de telefone correto e nem sempre há funcionário para atender a ligação, e ainda, muitas vezes é fornecido o horário errado. (OURO VERDE, 2015)

Levando em conta de que os computadores e *smartphones* fazem parte do dia a dia de grande parte da população, o objetivo desse trabalho é desenvolver um projeto de site e aplicativo para disponibilizar os horários, tornando a consulta fácil e prática.

A ideia principal é que futuramente a aplicação seja desenvolvida por uma equipe e comprada pela empresa, que administrará o sistema, como as entradas de dados sobre ônibus, funcionários, entre outros. Os horários de cada linha serão adicionados ao site pelo funcionário, e na mesma hora sincronizado ao aplicativo, sendo necessária conexão com a internet. Haverá disponível um modo para os usuários reportarem possíveis atrasos e ocorrências, tendo como informação o número da linha e horário.

Para a realização do projeto serão utilizados alguns dos principais diagramas da UML, tais como o diagrama de sequência, diagrama de classes e diagrama de casos de uso, e para a representação do banco de dados será utilizado o diagrama entidade relacionamento (DER). Também serão utilizados alguns conceitos de engenharia de software.

No projeto, o aplicativo Android será desenvolvido utilizando linguagem Java, por ser a linguagem mais utilizada para o desenvolvimento de aplicativos Android e pelo fácil acesso ao conhecimento, e o site utilizando PHP, Javascript e CSS, que são as linguagens mais utilizadas em aplicações web. O banco de dados será desenvolvido utilizando MySQL, por ser um dos mais utilizados e ser compatível com diversas plataformas, e para efetuar a comunicação entre o aplicativo e o banco de dados será utilizado Web Service.

O trabalho será desenvolvido em algumas etapas:

• Levantamento de requisitos;

• Definição das funções;

• Desenvolvimento dos diagramas (UML e DER);

• Desenvolvimento dos layouts;

• Possibilidade de melhorias.

Este trabalho será feito com o objetivo de solucionar o problema referente aos horários das linhas urbanas de ônibus em Sumaré-SP, problema que já ocorre há anos, e que está relacionado a um dos meios de transporte mais importantes no dia a dia dos moradores da cidade. A importância é grande, visto que é uma cidade que cobre 153,465km² e tem mais de 240.000 habitantes. (IBGE, 2015)

A pesquisa para o trabalho será feita utilizando livros e *sites* sobre programação e engenharia de *software*, conhecimento próprio adquirido durante o curso, e a ajuda de amigos, familiares e professores.

# 1 REVISÃO BIBLIOGRÁFICA

## 1.1 Sistemas *web*

Os sistemas *web* são aplicações que funcionam através da internet, e que podem ser acessadas através de um navegador por clientes, funcionários, entre outros, de acordo com a necessidade. Em várias ocasiões, os sistemas *web* se mostram mais vantajosos do que aplicações desktop.

> Com o passar dos anos, as aplicações *web* evoluíram rapidamente de simples *web sites* cujo propósito era apenas navegação sobre a informação para verdadeiros sistemas de informação altamente complexos, repletos de dados e transações, voltados para a implementação de processos de negócio intra- e interorganização. Diante deste quadro, a necessidade de um processo de software sistemático que ajude a gerenciar o ciclo de vida de tais aplicações surge naturalmente. (JACYNTHO, 2009)

A utilização de sistemas *desktop* está se tornando cara, não há muita praticidade para desenvolver interfaces amigáveis e são necessários computadores de alto desempenho para executar aplicações de grande porte.

Através da utilização de sistemas *web*, muitos desses problemas são solucionados. Algumas vantagens da utilização de sistemas *web* são:

• O acesso é flexível: é possível acessar o sistema em vários locais e a qualquer hora;

• Oferece melhor desempenho: a interface é muito amigável e não necessita de grande capacidade do computador, precisando apenas haver um navegador disponível;

• O custo é baixo, tanto para licenças quanto para equipamentos. Como dito no tópico acima, é preciso apenas que o computador execute um navegador. Quanto aos sistemas, é possível desenvolvê-los utilizando linguagens como PHP e Javascript, que não necessitam de licença.

Dois dos pontos que requerem atenção no desenvolvimento e uso de uma aplicação *web* são:

• Segurança: é preciso observar todos os detalhes, como senhas, criptografia, servidores bem configurados, controle contra vírus, firewall, brechas no desenvolvimento, etc.

• Comunicação: é necessário verificar se a conexão é adequada ao uso da aplicação, pois conexões de baixa qualidade/velocidade poderão causar a indisponibilidade do sistema.

### *1.1.1 Web services*

A finalidade dos *Web Services* é proporcionar a integração entre aplicações e sua comunicação através da internet, de maneira padronizada. Como as aplicações podem ter linguagens diferentes, existem formatos universais para possibilitar a comunicação, como Json, XML, etc. Esses arquivos são decodificados para interagir com as aplicações, tornando-as assim interoperáveis.

> [...] A Internet surgiu diante de empresas que já faziam uso de seus sistemas computacionais e que, geralmente, não foram desenvolvidos para serem interoperáveis, por exemplo, com os sistemas computacionais de seus clientes, fornecedores, etc. Diante da necessidade da interação entre as aplicações distribuídas de diferentes organizações, uma nova caracterização de sistemas distribuídos surgiu possibilitando assim a troca de informações e a integração com os sistemas legados existentes - os *Serviços Web* (MELLO, et al., 2006, p. 02).

## 1.2 Aplicativos *mobile*

Quando foram lançados os primeiros *smartphones*, ninguém sabia o que esperar deles. Hoje é possível verificar o tamanho do sucesso que eles alcançaram, e trouxeram junto com eles um novo segmento de desenvolvimento: o desenvolvimento de aplicativos *mobile*, que movimenta bilhões de dólares e gera muitos empregos todos os anos. Não há dúvidas de que os aplicativos *mobile* fazem parte da vida de grande parte das pessoas de todo o mundo, interconectando-as, facilitando atividades diárias, entre uma infinidade de outras funcionalidades e

facilitações. Em 2010 os aplicativos móveis se tornaram tão populares, que a palavra "*app*" foi assinalada com o a palavra do ano pela *American Dialect Society*.

> Ter acesso a um *smartphone*, definitivamente, mudou o jeito como as pessoas vivem as cidades. Consultar o mapa das ruas, os horários do ônibus e as críticas dos restaurantes deixam até a maior das metrópoles acessível, com tudo que ela tem a oferecer. [...] proporcionando ao indivíduo uma interação de forma conectada, descentralizada e simultânea, características que configuram uma metrópole polifônica (TAROUCO, 2013, p. 13).

Geralmente, os aplicativos são baixados através de uma loja online, como a *Google Play* ou *App Store*.

No desenvolvimento, a complexidade vai de acordo com a estrutura do aplicativo, do número de dispositivos em que o aplicativo irá operar, as plataformas em que será disponibilizado, entre outros. Tudo isso deve ser levado em conta antes da criação de um aplicativo. Duas das plataformas mais comuns atualmente são o Android e o iOS.

### *1.2.1 iOS*

O iOS (*iPhone Operating System*) é o sistema operacional para dispositivos móveis da Apple. Primeiramente foi desenvolvido exclusivamente para iPhone, mas também passou a ser utilizado no iPod, iPad, e recentemente no Apple TV.

> O mercado de aplicações para os dispositivos móveis está em altíssimo crescimento, sendo o mercado da Apple um dos principais mercados deste ramo. Diversas empresas querem estar disponíveis nestes dispositivos e para isso necessitam de desenvolvedores. [...] O mercado é muito grande e segue com uma taxa muito alta de crescimento. O que falta no mercado são profissionais qualificados [...] (MILANI, 2012)

Como citado acima, um dos principais mercados é o da Apple, que fica atrás apenas do sistema Android, sendo a quota de mercado aproximadamente 82,8% do Android e 13,9% do iOS. (IDC, 2015).

Existem alguns requisitos para desenvolver para dispositivos iOS. É preciso ter um computador MAC e ter cadastro como Desenvolvedor Apple. O cadastro como Desenvolvedor Apple garante o acesso à ferramenta de desenvolvimento, códigos, etc. O cadastro é feito pelo *iPhone Developer Program* gratuitamente,

porém, para ter direito a testar os aplicativos em um dispositivo real e publicá-lo na *App Store* é preciso pagar anualmente o valor de US$99,00. Após fazer o cadastro é permitido o download do iOS SDK. Fazem parte do iOS SDK:

-Xcode: Ambiente de desenvolvimento.

-*Instruments*: Ferramenta que analisa alguns aspectos do aplicativo, como desempenho, consumo de memória, etc.

-*iPhone Simulator*: Simula o dispositivo, utilizando os recursos do computador, para que seja possível testar o aplicativo.

-*Interface builder*: Para criação das interfaces.

### *1.2.2 Android*

Android é o sistema operacional para dispositivos móveis (principalmente *smartphones* e *tablets*), desenvolvido pela Google, baseado no kernel Linux. Também é utilizado em videogames, câmeras digitais, entre outros.

É o sistema operacional mais utilizado do mundo para dispositivos móveis. Em 2013, o Android já ultrapassava em 80% o número de vendas de *smartphones* em relação aos concorrentes (IDC apud tecnoblog) e já tinha acima de 1 milhão de aplicativos disponíveis para download na *Google Play*, somando mais de 50 bilhões de *downloads* (PhoneArena). Em 2014, foi anunciado na conferência anual Google I/O que existiam mais de 1 bilhão de usuários do sistema ativos.

O sistema Android foi projetado com o objetivo de possibilitar com que os desenvolvedores pudessem criar aplicações que aproveitassem do máximo que o dispositivo pode oferecer. Por ser de código aberto, é possível que um aplicativo se comunique com as funções do núcleo do dispositivo, como realizar chamadas, utilizar câmera, obter localização, entre outros. O fato de ser código aberto também ajuda na evolução do sistema, pois a comunidade de desenvolvedores sempre estará reproduzindo novas melhorias e soluções.

Mesmo com os aplicativos podendo ter acesso as funções do celular, o sistema Android é forte em questão de segurança: toda vez que um aplicativo é instalado, o Android cria um usuário Linux com diretórios em que só aquele aplicativo terá acesso.

> Cada processo da aplicação no Android é considerado uma *sandbox*. Só é possível acessar outras aplicações caso tenha as permissões explicitamente declaradas, para que elas possam ser conhecidas desde o momento da instalação, e nada irá fazer com que sejam alteradas após isto. (PEREIRA, 2009)

Sendo assim, um aplicativo não consegue ter acesso às funções de outro, pois ficam isolados em diretórios diferentes. A única maneira de um aplicativo acessar informações de outro aplicativo é através da permissão do usuário.

Os aplicativos para Android são desenvolvidos através do sistema de desenvolvimento SDK, que inclui bibliotecas, documentação, depurador, etc.

> O Android SDK é o *software* utilizado para desenvolver aplicações no Android, que tem um emulador para simular o celular, ferramentas utilitárias e uma API completa para a linguagem Java, com todas as classes necessárias para desenvolver as aplicações. (LECHETA, 2013)

Há também ferramentas para novos desenvolvedores, como *o Google App Inventor*.

O assunto sobre desenvolvimento para Android será tratado no capítulo 3 deste trabalho.

## 1.3 Engenharia de software

É uma área voltada ao desenvolvimento e manutenção de software, com práticas de gerenciamento de projetos, entre outras. Tem como objetivo alcançar melhor qualidade, organização e produtividade do software.

> Visando melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de Software. A Engenharia de Software trata de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de software. (FALBO, 2005)

### *1.3.1 Áreas de conhecimento*

Segundo o *Software Engineering Body of Knowledge* (SWEBOK, 2014, p. xxv), as áreas de conhecimento da engenharia de *software* são:

#### *1.3.1.1 Requisitos de software*

Requisitos de *software* são propriedades que um *software* deve ter para resolver um problema do mundo real. (Ex.: Automatizar determinada tarefa através do sistema). Geralmente, os requisitos são extraídos de várias pessoas que atuam no ambiente em que o *software* é/será utilizado.

> A área de conhecimento Requisitos de *Software* trata da elicitação, análise, especificação e validação dos requisitos de *software*. A má condução dessas atividades tornam projetos de engenharia de *software* criticamente vulneráveis. (SWEBOK, 2014)

A extração de requisitos está ligada ao resultado final do sistema, por isso é de extrema importância. Os requisitos podem ser funcionais ou não funcionais: os funcionais são funções em que o *software* deve executar, e os não funcionais são restrições (Ex.: Desempenho, confiabilidade, organização, entre outros).

#### *1.3.1.2 Design de software*

No processo de design de *software* define-se a arquitetura, componentes e a interface do *software*. Nesse processo os requisitos são usados para estipular como será a estrutura interna do sistema e a interface

> O Design de *Software* emprega uma importante regra no desenvolvimento de *software*: isso permite engenheiros de *software* produzirem vários modelos que formam um tipo de plano de solução para ser implementado. Nós podemos analisar e avaliar estes modelos para determinar se eles nos permitirão atender as diversas necessidades. (SWEBOK, 2014)

#### *1.3.1.3 Construção de software*

No processo de construção de *software*, o *software* é construído de maneira mais detalhada, abrangendo códigos, testes, estrutura, qualidade, etc. envolvendo todas as outras áreas de conhecimento.

> Os fundamentos da construção de *software* incluem:
>
> [...]A) Minimizar complexidade
>
> Um fator principal em como as pessoas transmitem suas intenções aos computadores, é a capacidade muito limitada das pessoas em memorizar complexas estruturas e informações [...]. Isto conduz a um dos mais fortes controles em construção de *software*: minimizar complexidade. A necessidade de reduzir a complexidade, se aplica essencialmente a todos os aspectos da construção de *software*, e é particularmente crítico ao processo de verificação e testes de construção de *software*. [...]. (SWEBOK, 2014)

Como dito acima, minimização da complexidade é muito importante e está presente em todas as faces da construção de *software*, pois exige menos capacidade de memorização das pessoas em informações complexas, facilitando o processo de construção, etc.

> B) Antecipação de mudança
>
> A maior parte dos *softwares* irá mudar com o tempo, e a antecipação das mudanças controla muitos aspectos da construção de *software*. A mutação de ambientes externos de *software* é parte inevitável, e as mudanças nesses ambientes externos afetam o *software* em diversas maneiras. [...]. (SWEBOK, 2014)

Com a constante mudança em *softwares*, entre outros, é preciso que se esteja atento ao que pode mudar no mercado. Com esse “movimento” antecipado é possível obter melhores resultados a curto e a longo prazo.

> C) Construção para verificação
>
> Construção para verificação significa construir o *software* de forma que as falhas podem ser prontamente encontradas pelo engenheiro de *software* [...]. (SWEBOK, 2014)

Construção para verificação é um ponto importante da construção de *software*, pois possibilita que o engenheiro de *software* encontre rapidamente os erros no *software*, evitando assim problemas futuros.

> D) Normas de construção
>
> Normas que afetam diretamente questões de construção incluem: Métodos de comunicação, linguagem de programação, plataforma, ferramenta; Uso de normas externas[...]. (SWEBOK, 2014)

Nas normas de construção são estipulados os padrões no qual o *software* será desenvolvido, como a linguagem, plataforma, ferramentas, métodos de desenvolvimento, entre outros.

Alguns modelos de construção são o modelo de ciclo de vida, cascata, prototipagem, SCRUM e XP.

#### *1.3.1.4 Teste de software*

No processo de teste, o sistema é testado para que seja avaliada a sua qualidade. Tem por objetivo encontrar problemas e defeitos e aprimorar o sistema. Durante o teste, é testado o comportamento do sistema diante de várias entradas (para tentar forçar ao erro), podendo existir vários critérios de teste.

> Um componente muito importante do teste bem sucedido é uma atitude colaborativa para o teste e as atividades de garantia da qualidade. Os gerentes têm um papel chave em promover uma recepção favorável para a descoberta da falha durante o desenvolvimento e a manutenção; por exemplo, impedindo uma atitude mental da posse de código entre programadores, de modo que não sintam responsáveis pelas falhas reveladas por seu código. (SWEBOK, 2014)

Como dito acima, nessa fase é importante que o gerente saiba administrar a descoberta de falhas durante o desenvolvimento, para que os programadores não se sintam culpados pelos erros em seu código, gerando conflitos.

#### *1.3.1.5 Manutenção de software*

Depois de implantado, o sistema apresenta mais erros e necessidade de ser mudado ou melhorado, por isso é necessária a manutenção.

Geralmente, é dado menos importância ao processo de manutenção de *software*, comparada ao processo de desenvolvimento.

> Manutenção de *Software* é definida [...] como a modificação de um produto de *software* após a entrega para corrigir falhas, para melhorar o desempenho ou outros atributos, ou adaptar o produto para um ambiente modificado. [...]. (SWEBOK. 2014)

#### *1.3.1.6 Gerência de configuração de software*

A configuração de *software* busca padronizar e controlar as características físicas e funcionais do projeto, a forma como são feitas alterações, manutenções,

etc. tendo como objetivo manter a integridade dos processos, identificando o que vai ser controlado, versões, e quais técnicas serão utilizadas no gerenciamento desses itens, de modo a facilitar o desenvolvimento.

> Gerência de configuração de *software* controla a evolução e integridade de um produto pela identificação de seus elementos, gerenciando e controlando mudanças, e verificando, registrando e apresentação informações sobre configuração. A partir da perspectiva do engenheiro de *software*, SCM facilita o desenvolvimento e a implementação de atividades de mudança. (SWEBOK, 2014)

#### *1.3.1.7 Gerência de engenharia de software*

Busca aplicar todas as atividades de gerenciamento, como o planejamento coordenação, etc. para que o desenvolvimento e manutenção sejam feitos de forma sistemática e padronizada.

> Relevante para esta KA (*knowledge area*) é a noção da gerência de projeto, como 'a construção de artefatos de *software* úteis' normalmente é gerida na forma de (talvez programas de) projetos individuais. Neste sentido, encontramos amplo apoio no Guia para o Conjunto de Conhecimentos da Gerência de Projetos (PMBOK) (SWEBOK, 2014)

É importante nessa área de conhecimento a noção em gestão de projetos, sendo o PMBOK um bom guia sobre o assunto.

#### *1.3.1.8 Processos de engenharia de software*

O processo de engenharia tem como objetivo encontrar a melhor maneira de definir os processos no projeto. É dividido em dois níveis. O primeiro contém as atividades técnicas e gerenciais durante o desenvolvimento, manutenção, etc. O segundo engloba a implementação, gerência, mudança, etc. do *software*.

Os processos podem ser procedimentos, normas, entre outros.

> [...] é importante notar que o contexto do projeto e da organização irão determinar a definição do tipo de processo que é mais útil. Variáveis importantes a considerar incluem a natureza do trabalho (por exemplo, a manutenção ou desenvolvimento), o domínio da aplicação, o modelo do ciclo de vida, e da maturidade da organização. (SWEBOK, 2014)

#### *1.3.1.9 Ferramentas e métodos de engenharia de software*

As ferramentas servem para ajudar em refazer e definir tarefas automatizadas, e, assim, permitir que a engenharia de *software* volte seu foco mais para outros aspectos.

Os métodos da engenharia de *software* são atividades e metas feitas de forma sistemática, para aumentar a chance de sucesso do projeto.

> Embora existam manuais detalhados, ferramentas específicas e numerosos trabalhos de investigações em relação a ferramentas inovadoras, técnicas genéricas, de escrita de ferramentas de software são relativamente escassas. Uma das dificuldades é a elevada taxa de mudança nas ferramentas de software em geral. Detalhes específicos são alterados regularmente. (SWEBOK, 2014)

#### *1.3.1.10 Qualidade de software*

"Qualidade de *software*" é definida de forma diferente por cada organização, mas, basicamente, é entregar o *software* com perfeição, alcançando todos os requisitos.

> A noção de "qualidade" não é tão simples quanto pode parecer. Para qualquer produto desenvolvido, existem várias qualidades desejadas relevantes para determinadas perspectivas do produto, que devem ser discutidas e determinadas durante a definição dos requisitos. Características de qualidade podem ser requeridas ou não, podem ser requeridas em maior ou menor grau, e ainda podem existir trade-offs entre elas. (SWEBOK, 2014)

Sendo assim, qualidade é algo bastante relativo, e de extrema importância para a engenharia de *software*, estando presente em todas as áreas de conhecimento.

### *1.3.2 Processo de software*

O processo de engenharia de *software* é uma sequência de práticas que visa o melhor desenvolvimento do sistema. Essas práticas envolvem atividades de especificação, *design*, implementação e testes. No processo de *software*, são utilizados modelos para uma representação das atividades envolvidas. Alguns modelos de processo de *software* são: Modelo do ciclo de vida, Desenvolvimento iterativo e incremental, V-model, Espiral, Sequencial (ou Cascata), Formal, Ágil,

Desenvolvimento rápido de aplicação (RAD), e entre eles o escolhido para o desenvolvimento desse trabalho: Scrum (tema que será abordado no próximo capítulo).

# 2 SCRUM

Considerando que o desenvolvimento das aplicações será feito por uma equipe, o modelo de processo de software escolhido foi Scrum.

Scrum é uma metodologia ágil para gerenciamento de projetos de *software*.

Os papéis principais no Scrum são:

• *Product Owner*: representa a voz do cliente e precisa assegurar que a equipe agregue valor ao negócio. É a pessoa que gerencia o *Product Backlog.*

• *Scrum Team*: é a equipe com aproximadamente 7 pessoas que analisa, desenvolve, testa, etc.

• *Scrum Master*: É ele quem gerencia todos os processos. É considerado um mentor tanto do *Product Owner* quanto do *Scrum Team*. Seu trabalho é não deixar que nada impeça a equipe e mantê-la focada em atingir seus objetivos em cada *Sprint*.

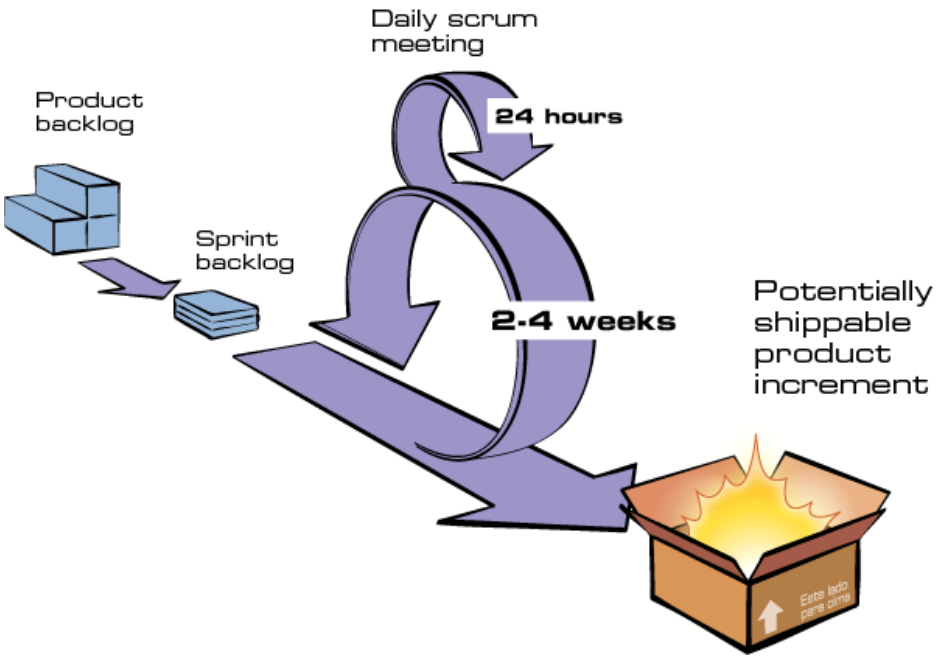
No Scrum os projetos são separados em ciclos, chamados de *Sprints*, que geralmente tem duração de quatro semanas. Os requisitos a serem implantados no projeto ficam em uma lista que é chamada de *Product Backlog*. No começo de cada *Sprint* é feita uma reunião para serem decididos quais requisitos farão parte do *Sprint* que se inicia. Essa reunião é chamada de *Spring Planning Meeting*. Nessa reunião, o *Product Owner* vê o *Product Backlog* e verifica quais requisitos tem maior prioridade e podem ser adicionados no novo *Sprint*. Esses requisitos vão para o *Sprint Backlog*. O *Scrum Team* prioriza esses requisitos e os põe em prática.

Enquanto uma *Sprint* é executada, em todos os dias é feita uma rápida reunião (geralmente com duração de 15 minutos e feita durante a manhã), chamada de *Daily Scrum*.

> [...] São feitas três perguntas-chave e respondidas por todos os membros da equipe [Noy02]:
>
> • O que você realizou desde a última reunião de equipe?
>
> • Quais obstáculos está encontrando?
>
> • O que planeja realizar até a próxima reunião da equipe? (PRESSMAN, 2011).

Nessa reunião é verificado o que cada pessoa fez no dia anterior, a fim de se saber o progresso da tarefa, as dificuldades que cada um encontrou e definir o que será feito no intervalo entre as reuniões. No final do *Sprint*, é feita uma reunião para que sejam apresentadas as novas funcionalidades implementadas, reunião chamada de *Sprint Review Meeting*. Depois disso, é feita a *Sprint Retrospective*, que serve para verificar o que pode ser melhorado nas próximas *Sprints* e como melhorar. Feito isso, inicia-se o ciclo novamente. A figura 1 mostra o ciclo scrum.

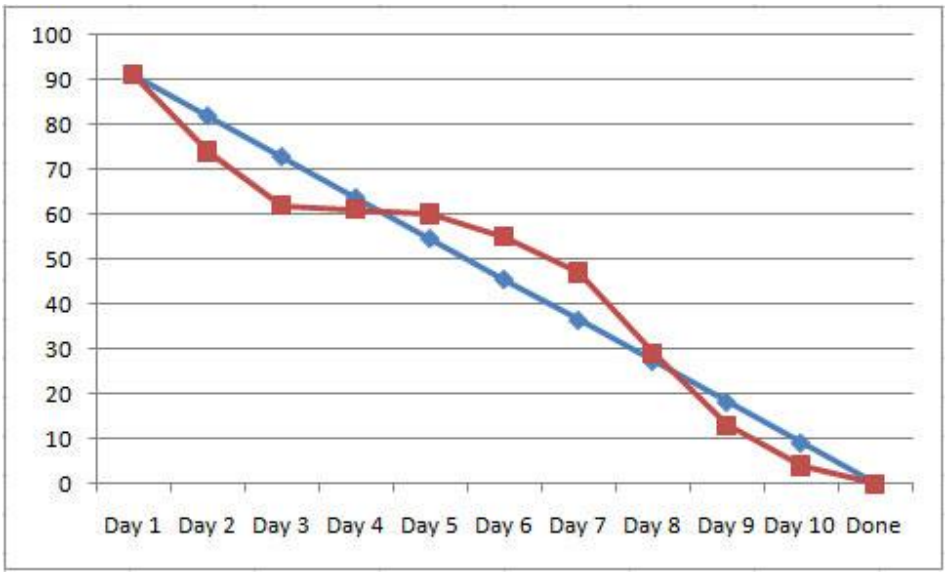
Figura 01 – Ciclo Scrum.




Fonte: www.desenvolvimentoagil.com.br (2014).

Uma das maneiras para medir o progresso da *Sprint* e saber como está o processo do trabalho em equipe é utilizando o *Burndown Chart.* O *Burndown Chart* mostra o progresso diário do desenvolvimento do projeto em relação ao final da *Sprint*, e é importante para que se gerencie tempo e quantidade de trabalho. A figura 2 mostra um exemplo de *Burndown Chart*.

Figura 02 – *Burndown Chart*.




Fonte: www.p3praxis.com (2015).

No *Burndown Chart* há um eito X e um eixo Y. No eixo X são exibidos os dias da *Sprint,* e no eixo Y são exibidos os pontos que foram estipulados para formar a *Sprint,* indo do máximo até zero.

Os motivos pelo qual o Scrum foi escolhido para este trabalho são:

• Garante bom uso do tempo;

• Fica mais fácil de gerenciar dividindo o projeto em *Sprints*;

• Bom para desenvolvimento em equipe;

• Funciona bem para projetos em rápido desenvolvimento.

# 3 TEORIA – DESENVOLVIMENTO PARA ANDROID

Nesse capítulo serão abordados o que é preciso para desenvolver aplicativos Android e algumas informações importantes, como ferramentas, *APIs*, Frameworks e outros detalhes voltados ao desenvolvimento.

Para desenvolver aplicativos para Android são precisos de um *SDK* Android e de uma *IDE*.

O teste dos aplicativos pode ser feito através de um emulador, que simula um dispositivo específico, ou então pode ser feito em um dispositivo Android real, colocando-o em modo desenvolvedor e transferindo o arquivo apk.

## 3.1 *SDK* Android

O *Software Development Kit* é um kit para desenvolvedores. O *SDK* oferece pacotes de ferramentas, código e documentação, que podem ser utilizados no desenvolvimento dos aplicativos.

> O Android SDK é o software utilizado para desenvolver aplicações no Android, que tem um emulador para simular o celular, ferramentas utilitárias e uma API completa para a linguagem Java, com todas as classes necessárias para desenvolver as aplicações. (LECHETA, 2013)

No *SDK* Android são inclusos um emulador, ferramentas, e uma *API* com todas as classes. Por ser em código aberto, a Google disponibiliza o Android *SDK* para download. É possível baixar o *SDK* Android no site oficial *Android Developers*.

## 3.2 IDE

O *Integrated Development Environment* é um programa que reúne ferramentas, entre outros, para facilitar o desenvolvimento de software, focando em produtividade.

São características do *IDE*, entre outras:

• Editor: Edita o código escrito na linguagem suportada pela IDE.

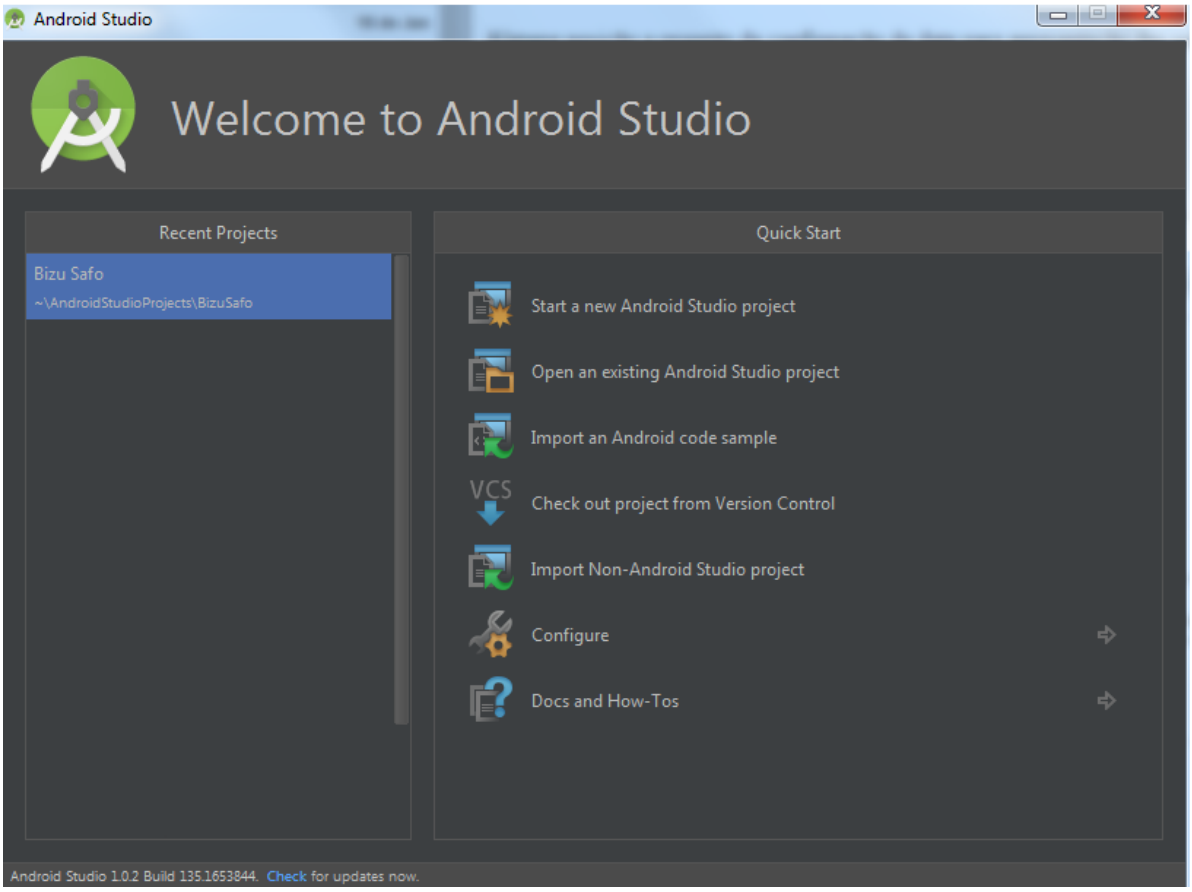
• Depurador: Usado para testar e eliminar os erros no código.

• Gerador de código: Os geradores de código presentes na maioria das *IDEs* são os geradores de código passivos. Os geradores de código passivos apenas geram o código incompleto e não final, precisando que o programador termine o código.

> [...] pode-se perceber o quanto um *IDE* bem estruturado e desenvolvido facilita o trabalho dos desenvolvedores de software. [...] oferece várias facilidades e suporte para vários tipos de projetos o que o torna mais livre para o desenvolvedor usar a criatividade de forma simplificada. No uso empresarial é de grande valia, pois oferece ferramentas que identificam erros com rapidez e eficiência, [...]. Tudo isso torna mais fácil programar no IDE do que em editores comuns, auxiliando na entrega rápida e confiável do produto ao cliente. (FARIA, 2010)

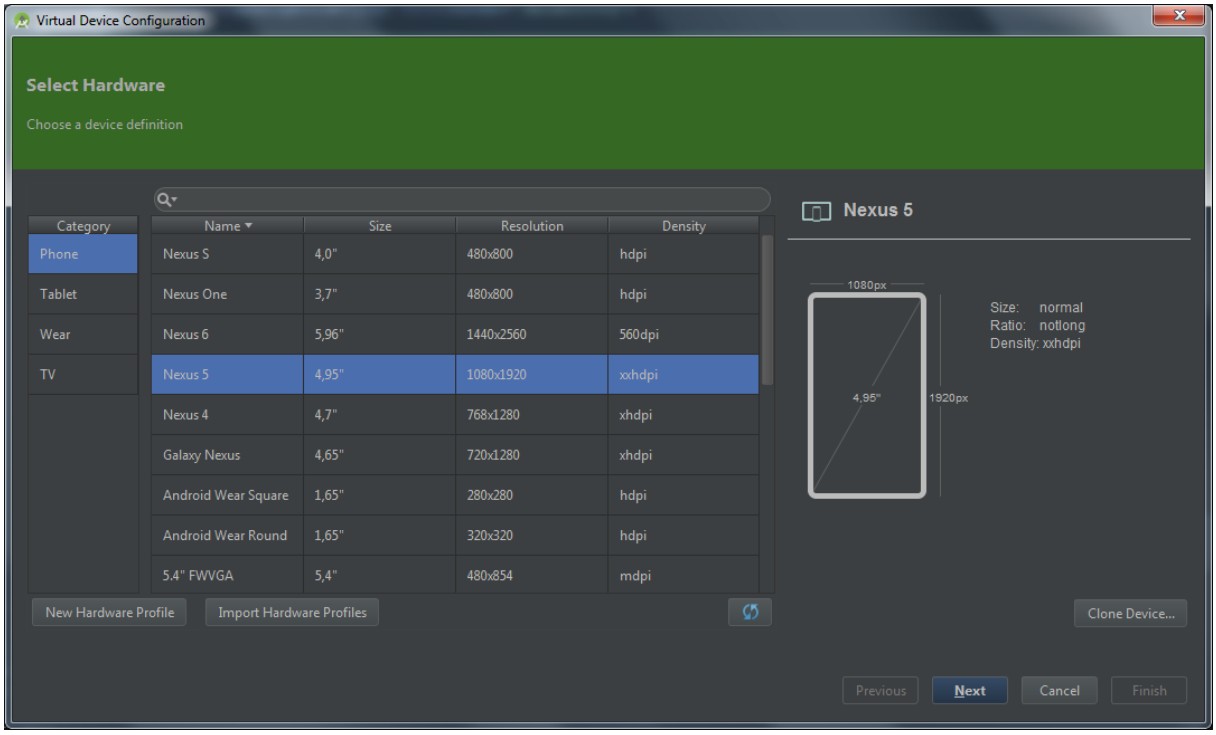
Existem várias *IDEs* para desenvolvimento Android, como por exemplo o Xamarin Studio, Eclipse, Visual Studio e, conforme exibida na figura 3 e 4, a *IDE* disponibilizada pela Google, Android Studio. Na figura 3 é exibida a tela de boas-vindas, e na figura 4 a seleção de hardware.

*Figura 03 – Tela inicial - IDE Android Studio.*




*Fonte: Ramon Rabello (2015).*

*Figura 04 – Seleção de Hardware - IDE Android Studio.*




*Fonte: Ramon Rabello (2015).*

## 3.3 *Framework*

Um *framework*, em desenvolvimento de aplicações, é uma união de códigos e classes utilizados em diversos softwares, resultando em uma função genérica. Essas classes devem ser flexíveis e possíveis de se estender, para que no desenvolvimento das aplicações seja necessário pouco esforço, sendo preciso apenas especificar os detalhes específicos de cada aplicação.

Assim, o principal objetivo dos *frameworks* é aumentar a produtividade, diminuindo esforços que poupam muito tempo dos desenvolvedores, fornecendo ferramentas embutidas para ajudar os desenvolvedores a trabalhar na parte lenta e difícil do desenvolvimento.

### *3.3.1 Frameworks para desenvolvimento em Android*

Hoje em dia, quase toda linguagem de programação tem *frameworks* para ajudar os desenvolvedores. Abaixo estão alguns *frameworks* para auxiliar no desenvolvimento para Android:

• Corona SDK: Corona é um *framework* que provê um desenvolvimento muito rápido. Suas *APIs* transformam tarefas complexas em tarefas simples, e o desenvolvimento é feito em Lua, que é uma linguagem fácil de aprender. É um *framework* bastante utilizado por desenvolvedores de jogos.

• Phonegap: Phonegap é um *framework* que torna possível o desenvolvimento em HTML5, CSS e JavaScript.

• Xamarin: *Framework* que torna possível que o código seja todo feito em C#.

• Appcelerator: O Titanium SDK que faz parte do Appcelerator utiliza linguagem JavaScript para construir aplicativos com boas performances.

• JQuery *Mobile*: Ajuda os desenvolvedores a construir sites e aplicativos que são acessíveis em todos os *desktops*, *tablets* e *smartphones*.

• Theappbuilder: Esse *framework* construído em HTML5 possui uma interface de arrastar e soltar, sem códigos, que permite um desenvolvimento rápido. Inclui envio direto a Google *Play*.

• OrmLite: utilizado para livrar o desenvolvedor de programar em SQL na parte relacionada ao banco de dados.

## 3.4 *API*

O *Application Programming Interface* é um conjunto de padrões e instruções que fornecem uma biblioteca para ser usada por outros programas. Funciona como uma interface que roda por trás do programa, conectando a aplicação ou site a diversos outros aplicativos e sistemas, de modo que o usuário não percebe quando esse recurso está em ação.

As empresas disponibilizam suas *APIs* para que desenvolvedores de software possam utilizar o serviço em suas aplicações. Por exemplo, quando o usuário

compra ingressos para um jogo de futebol online, o site usa uma *API* pra enviar as informações do cartão de crédito para um aplicativo que faz a verificação de validação dos dados. Após o procedimento de validação e confirmação do pagamento, esse aplicativo envia a resposta ao site, que libera os ingressos.

Um exemplo de *API*, que é utilizada em vários aplicativos relacionados a transporte, e que seria viável a implantação nesse projeto, é a Google Maps API.

### *3.4.1 API Android*

As APIs Android disponibilizam diversas classes, pacotes e funções.

> Um ponto forte das *APIs* básicas do Android é a otimização que estas possuem, focando a utilidade dos pacotes, em conjunto com um bom aproveitamento, deixando de fora pacotes pesados e pouco evoluídos. (PEREIRA, 2009)

Algumas das principais APIs do Android são:

• ***android.telephony***: para funções relacionadas a telefonia;

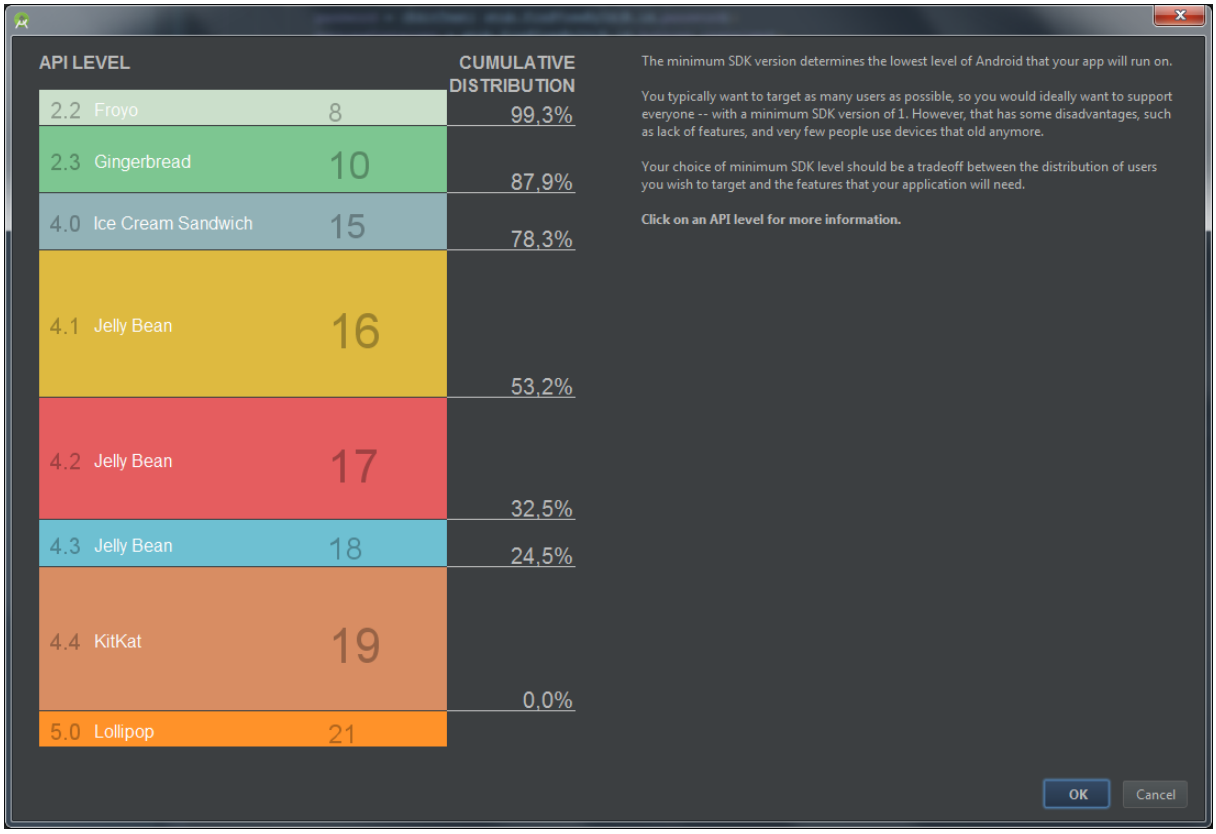
• ***android.maps***: para a utilização de mapas;

• ***android.os***: relacionado ao sistema operacional;

• ***android.database***: para a ligação ao bando de dados;

Como pode ser verificado na figura 5 a seguir, cada conjunto de níveis de API chega-se a uma versão diferente do Android, onde a cada nível são adicionados ou removidos alguns desses pacotes, classes, métodos e campos.

*Figura 05 – Seleção do nível de API – IDE Android Studio.*




*Fonte: Ramon Rabello (2015).*

## 3.5 Persistência de dados

Existem várias maneiras de salvar os dados dos aplicativos. A escolha de qual é o modo ideal depende das necessidades específicas, como se os dados devem ser privados ou devem ser acessíveis para outras aplicações, quanto espaço é necessário, etc.

Os modos de armazenamento dos dados são:

• *Shared Preferences*: permite salvar dados primitivos privados, como *int, float, string*, etc, em pares de chave-valor. Os dados permanecem na sessão do usuário.

• *Internal Storage*: Armazena os dados na memória do dispositivo.

• *External Storage*: Armazena os dados públicos no armazenamento externo compartilhado.

• Conexão de rede: Armazena os dados na *web*, no seu próprio servidor de rede.

• *SQLite*: Armazena dados de forma estruturada em um banco de dados privado.

### *3.5.1 SQLite*

O *SQLite* é uma boa opção para armazenamento dos dados, pois não exige instalação nem configuração, sendo prático e de bom desempenho.

É possível criar o banco no armazenamento externo para maior acessibilidade.

A manipulação do banco de dados e a execução de *queries* é feita a partir da classe *SQLiteDatabase*.

A biblioteca do Android junto com o SQLite torna o gerenciamento dos dados simples, sendo possível armazenar os dados no dispositivo com tranquilidade.

# 4 DESENVOLVIMENTO DO PROJETO

## 4.1 Requisitos do projeto

### *4.1.1 Funcionais*

#### *4.1.1.1 Requisitos gerais*

- Funcionalidades intuitivas;
- Botões fáceis de encontrar;
- Linguagem em Português do Brasil;

#### *4.1.1.2 Site*

Para o administrador:

- Controle de acessos;
- Controle de reports;
- Controle de funcionários e recuperação de senha;
- Controle/visualização das linhas e horários de ônibus;

Para o funcionário:

- Controle de horários de ônibus;
- Acesso à recuperação de senha;
- Visualização das linhas e horários de ônibus;

Para o usuário:

- Acesso à visualização das linhas e horários de ônibus.

*4.1.1.3 Aplicativo*

• Acesso aos horários e linhas de ônibus;

• Função de *report* de ônibus;

*4.1.2 Não funcionais*

Nessa área é muito importante incluir os atributos de qualidade de software, *assunto citado no capítulo sobre engenharia de software.*

*4.1.2.1 Funcionalidade*

Capacidade de providenciar funcionalidades que façam o usuário satisfeito em suas vontades explícitas e implícitas.

Características:

• Adequação: o quanto a função é adequada à exigência do cliente;

• Interoperabilidade: como o software interage com outros sistemas;

• Segurança: capacidade do sistema em proteger informações;

• Conformidade: projeto padronizado, dentro das normas;

• Acurácia: Representa a precisão do software em fornecer resultados.

*4.1.2.2 Confiabilidade*

Produto que se mantém o de desempenho em diversas situações.

Características:

• Maturidade: capacidade de evitar falhas provenientes do software;

• Recuperabilidade: capacidade do software de se recuperar após um erro;

• Tolerância a falhas: Capacidade de se manter em funcionamento, mesmo quando há falhas.

*4.1.2.3 Usabilidade*

A facilidade com que se consegue manusear o software e aprender suas funcionalidades.

Características:

• Inteligibilidade: facilidade em o usuário entender as funções;

• Operacionalidade: como o software facilita a operação do usuário, tolerando erros de operação;

• Atratividade: capacidade do software em um usuário ao sistema, podendo ser relacionado à sua interface gráfica.

• Apreensividade: Facilidade de aprendizado do software para seus usuários.

*4.1.2.4 Eficiência*

Capacidade do software em ter o desempenho ao nível dos recursos envolvidos.

Características:

• Utilização de recursos: mede os recursos consumidos em comparação com o quanto o software pode utilizar de recursos do sistema;

• Comportamento em relação ao tempo: Verifica se o tempo de resposta está de acordo com as especificações.

*4.1.2.5 Manutenibilidade*

Capacidade de o software ser modificado, como melhorias de funcionalidades, correções de erros, entre outros.

Características:

• Testabilidade: capacidade em testar as modificações no sistema;

• Estabilidade: capacidade em evitar problemas provenientes de novas atualizações;

• Analisabilidade: como o software verifica problemas e exibe as causas;

• Modificabilidade: Facilidade em ser modificado.

#### *4.1.2.6 Portabilidade*

Possibilidade de um sistema ser transferido de ambiente.

Características:

• Capacidade em ser instalado: Facilidade em modificar o ambiente do sistema;

• Capacidade para substituir outro sistema;

• Coexistência: Capacidade em funcionar com outros softwares instalados;

• Adaptabilidade: Capacidade em se adaptar a ambientes diversos sem configurações adicionais.

## 4.2 UML - Definição

A UML, ou Linguagem de Modelagem Unificada, é uma linguagem de modelagem que tem a função de representar um projeto de *software* de forma padronizada.

Os elementos da UML utilizados para a documentação desse projeto foram os Casos de Uso, Diagrama de Classes, Diagrama de Atividades e Diagrama de Sequência, que serão explanados a seguir.

### *4.2.1 Casos de Uso*

Um diagrama de Caso de Uso exibe as funcionalidades do sistema pelo ponto de vista do usuário.

O diagrama de Caso de Uso é representado por

• Atores: Usuários do sistema. Podem ser humanos ou outros dispositivos.

• Casos de uso: Funções grandes do sistema.

• Relacionamentos: Ajudam a descrever casos de uso. Existem alguns tipos de relações:

- Associação: faz a associação entre um ator e um caso de uso de modo a mostrar uma funcionalidade pelo ponto de vista do usuário.

- Generalização de atores: Um autor A recebe os casos de uso de um autor B. A continua tendo seus próprios casos de uso.

- Entre casos de uso: Include (B é essencial para comportamento de A), Extend (B pode ser acrescentado para descrever o comportamento de A) e Generalização (B é um caso de A).

### *4.2.2 Diagrama de Classes*

O Diagrama de Classes mostra o conjunto de objetos do projeto e seus relacionamentos.

Um Diagrama de Classes pode ser visto como:

• Conceitual: Exibe os conceitos básicos. Destinado ao cliente.

• Especificação: Focado nos principais métodos e interfaces, e não em como eles serão adicionados. Destinado a quem não precisa saber a parte de desenvolvimento muito detalhadamente.

• Implementação: Sendo a mais utilizada, exibe vários detalhes de implementação, como tipo dos atributos, métodos, entre outros. Voltado à equipe de desenvolvimento.

### *4.2.3 Diagrama de Atividade*

O Diagrama de Atividade exibe o fluxo de atividades em um processo, mostrando qual atividade depende da outra.

Um diagrama de atividade pode conter *swimlanes,* que são regiões que estão ligadas a um objeto. Por exemplo, no diagrama de um processo de retirar dinheiro de um caixa eletrônico, a parte de inserir o cartão estaria na região do cliente, enquanto a atividade de autorizar estaria na região do banco.

### *4.2.4 Diagrama de Sequência*

Um Diagrama de Sequência exibe como as mensagens são trocadas pelos objetos até que se termine uma operação. Por exemplo: Em uma operação de apagar funcionário, o Diagrama de Sequência exibe desde que a opção foi mostrada na interface e selecionada, até onde o usuário recebe o retorno de que o procedimento está completo.

Em um diagrama de sequência, há alguns elementos, como:

• Linhas verticais exibindo o tempo de vida do objeto;

• Essas linhas são preenchidas por barras verticais que indicam quando um objeto começou a existir. Quando o objeto deixa de existir, é marcado com um X na parte inferior;

• Linhas horizontais ou diagonais representando as mensagens trocadas entre os objetos.

• Condições são representadas por mensagens entre colchetes;

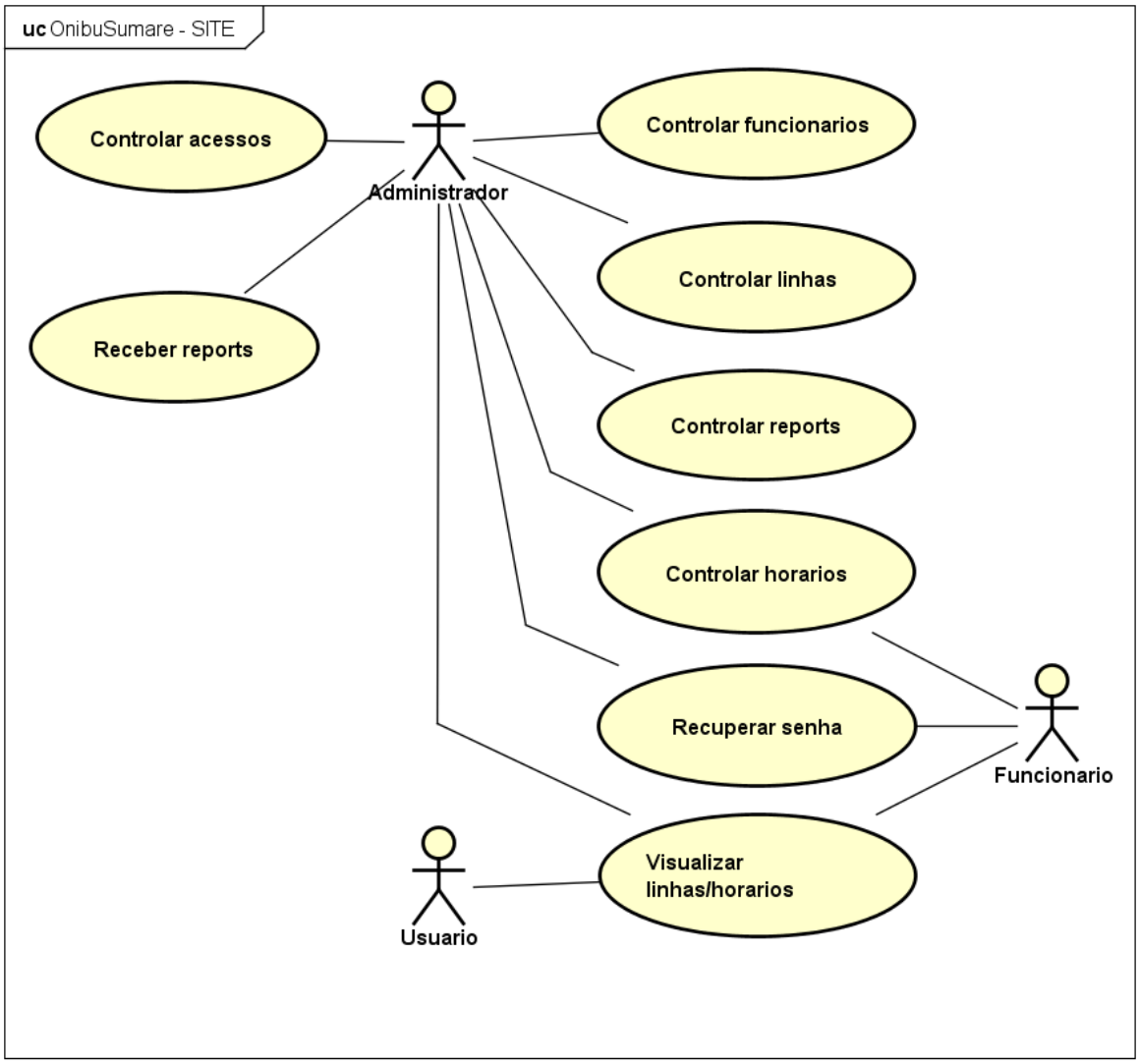
• Mensagens de retorno, que são representadas por linhas horizontais tracejadas. Para que o diagrama fique claro, esse tipo de mensagem só deve ser mostrada quando for fundamental.

## 4.3 Documentação

### *4.3.1 Site*

#### *4.3.1.1 Casos de uso*

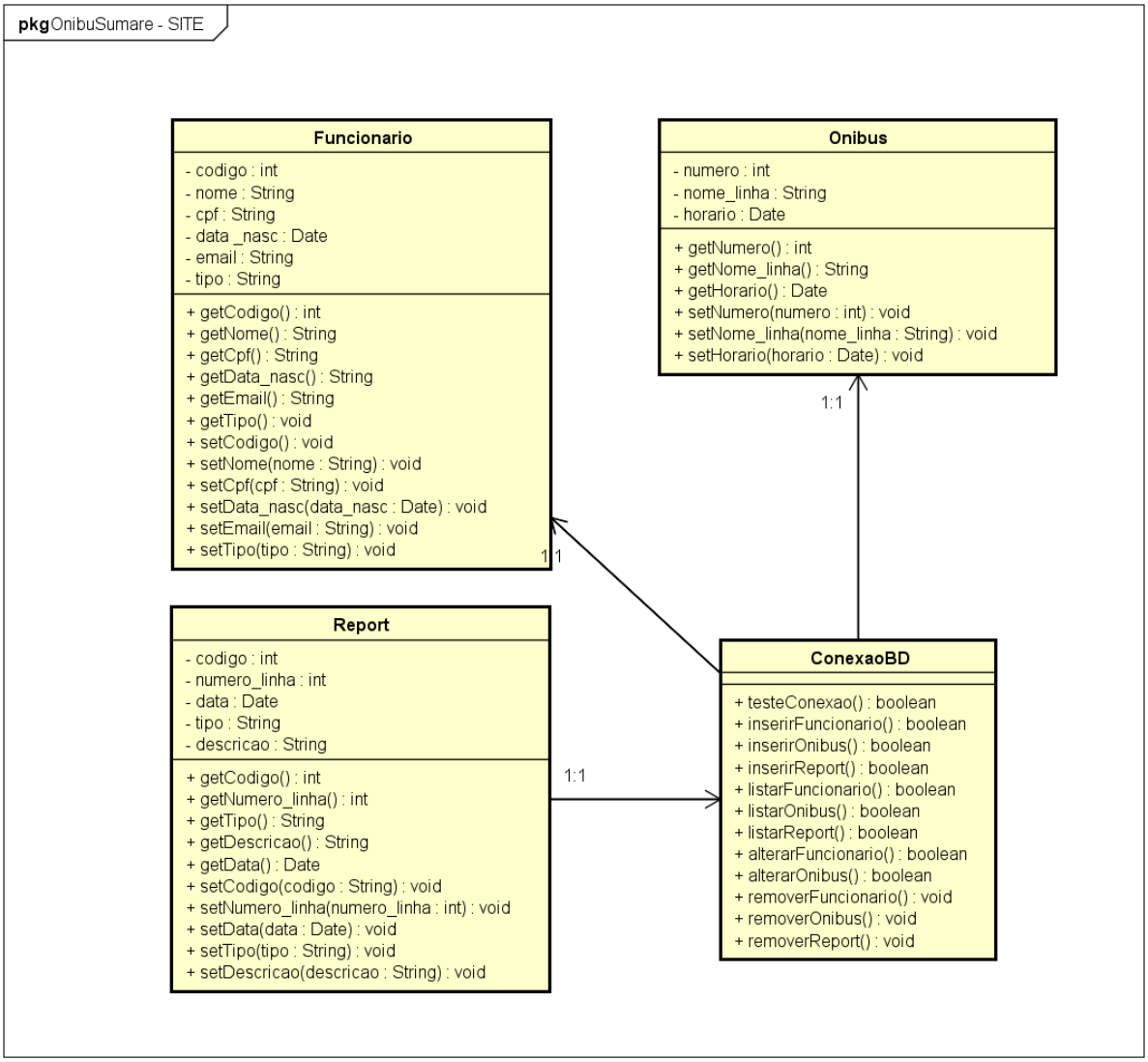
Figura 06 – Casos de Uso - SITE




Fonte: Elaborada pelo autor.

### *4.3.1.2 Diagrama de classes*

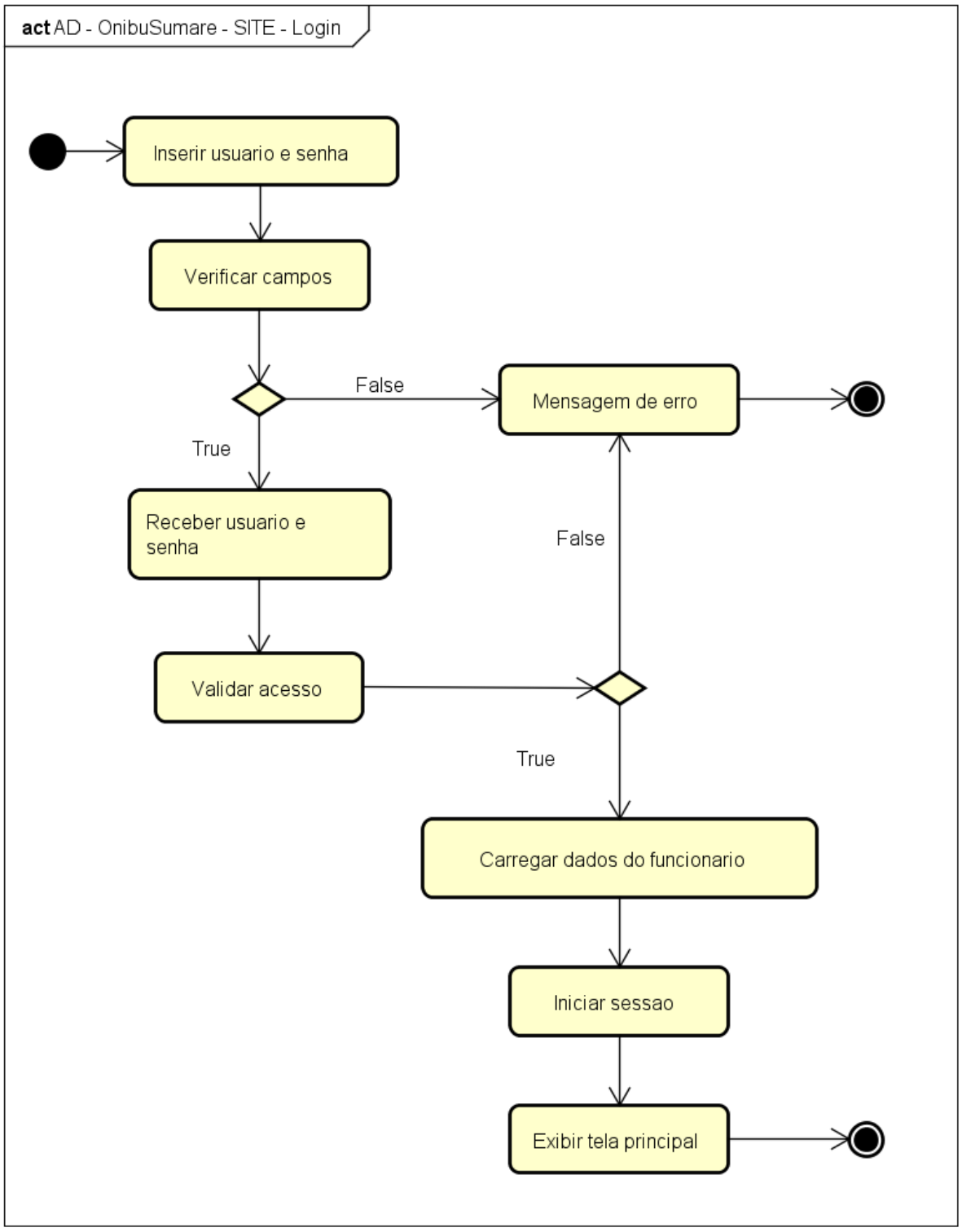
Figura 07 – Diagrama de Classes - SITE




Fonte: Elaborada pelo autor.

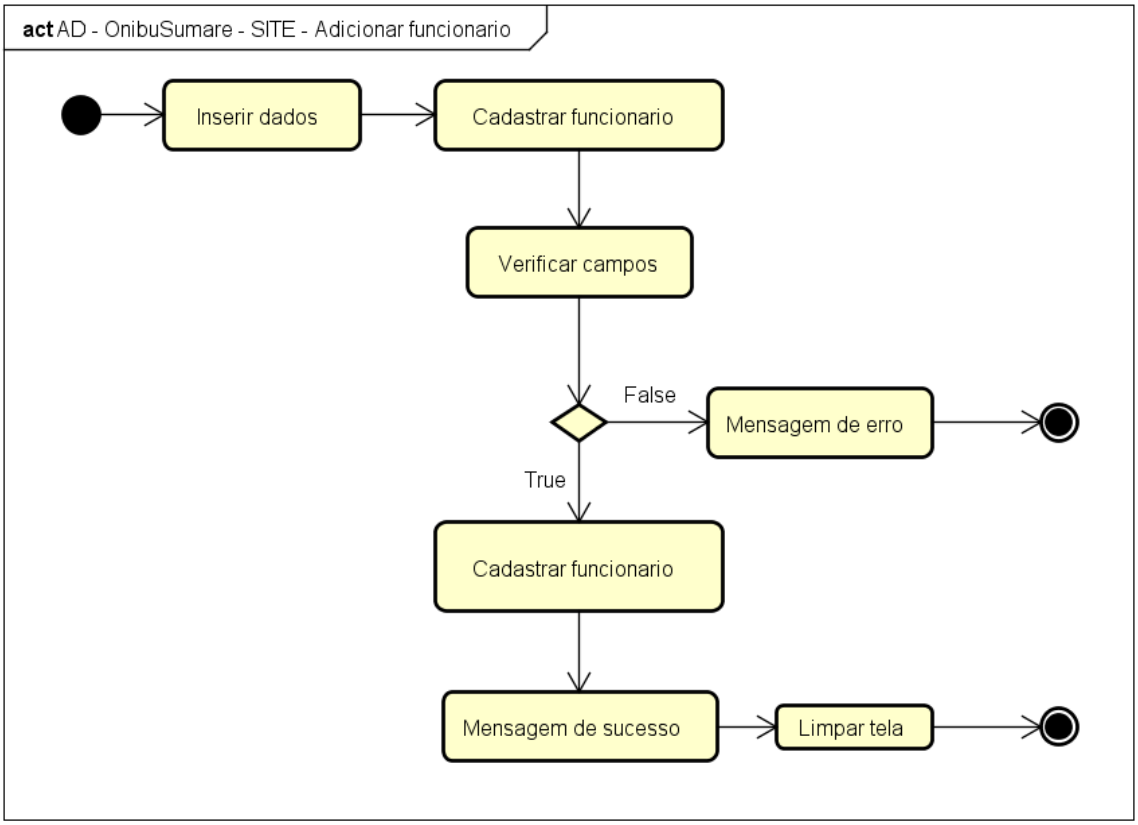
### *4.3.1.3 Diagramas de atividade*

Figura 8 – Diagrama de Atividade – SITE - Login



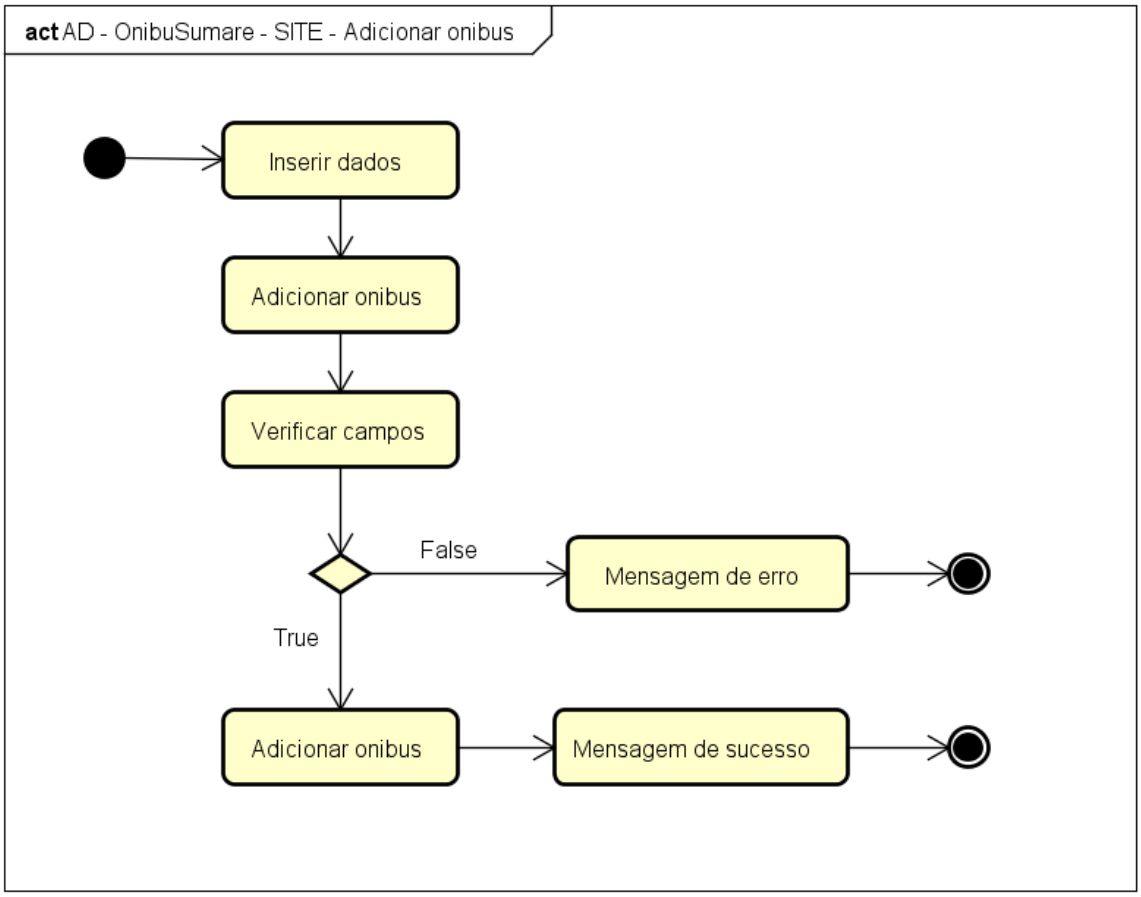

Fonte: Elaborada pelo autor.

Figura 9 – Diagrama de Atividade – SITE – Adicionar funcionário



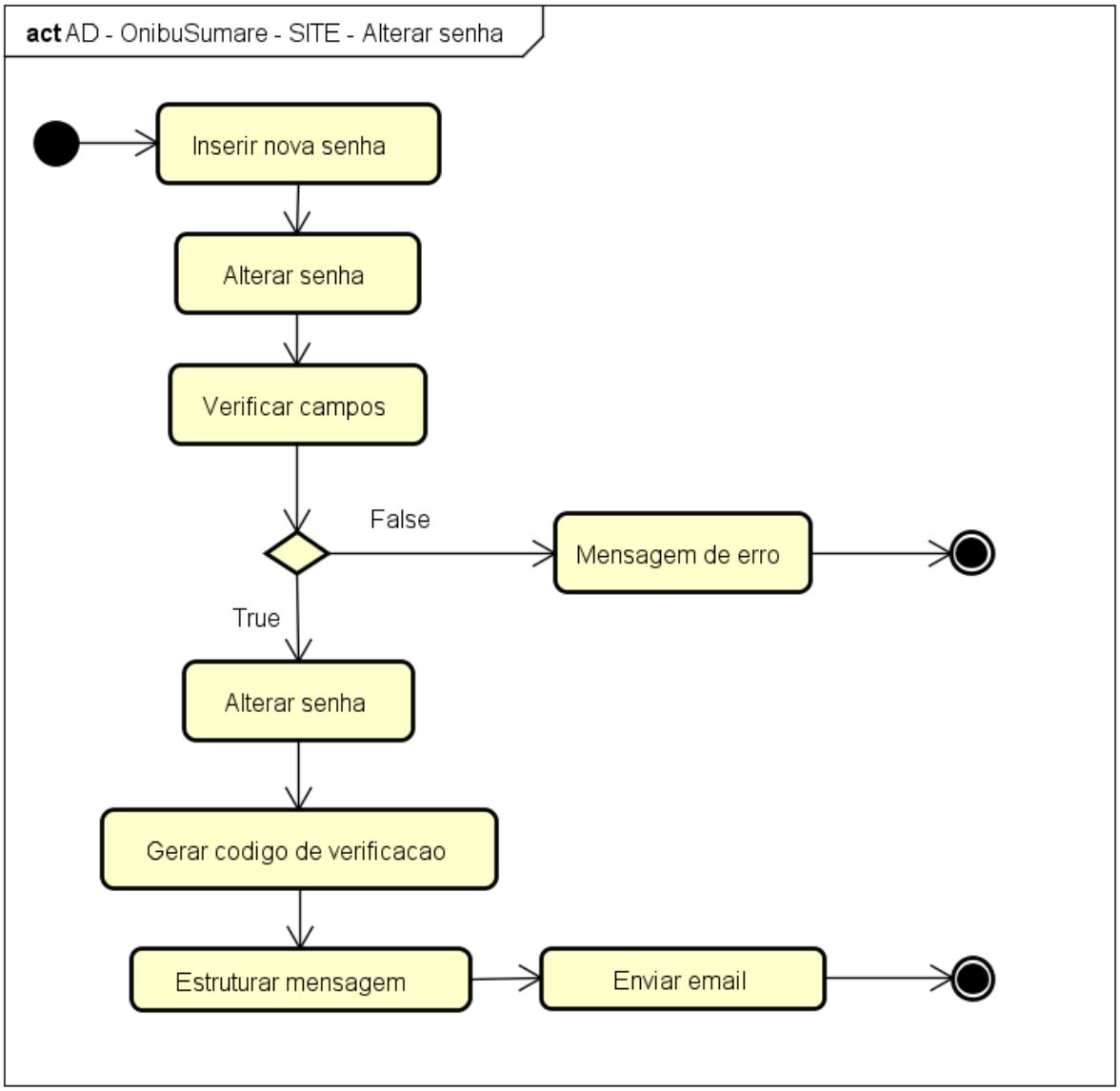

Fonte: Elaborada pelo autor.

Figura 10 – Diagrama de Atividade – SITE – Adicionar ônibus



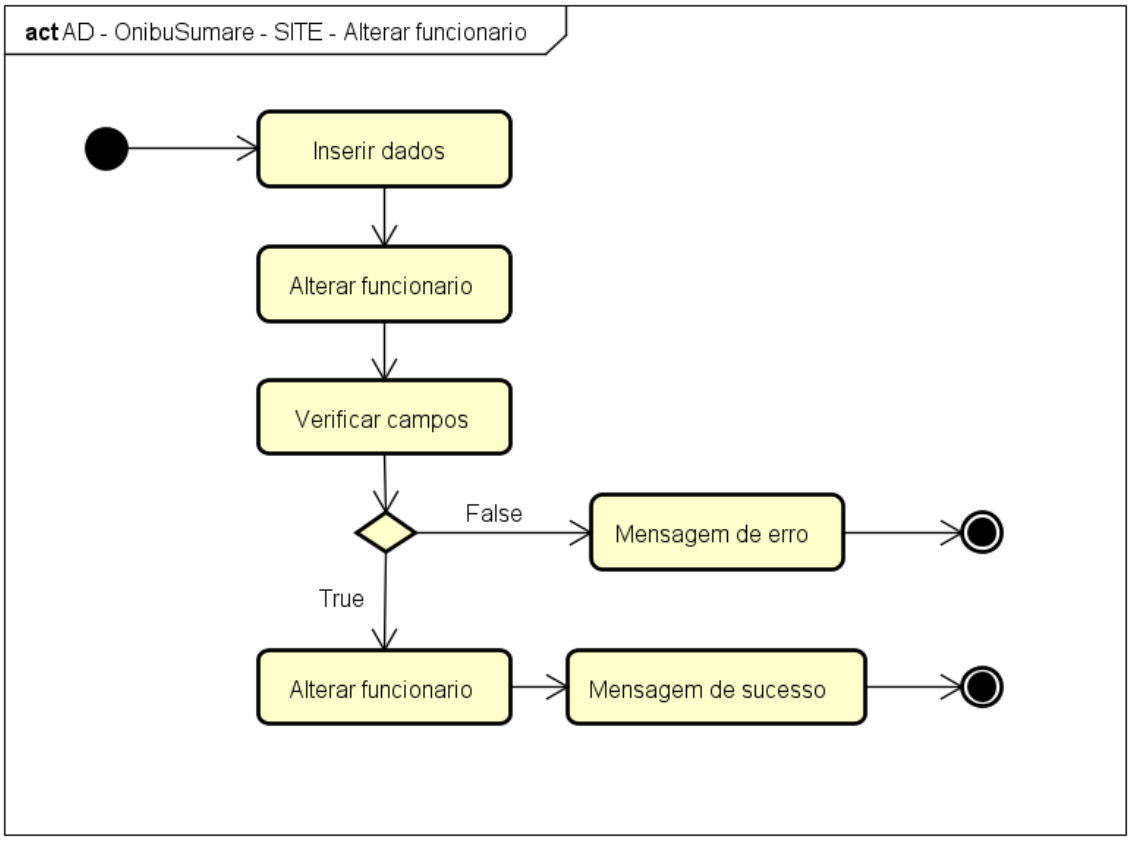

Fonte: Elaborada pelo autor.

Figura 11 – Diagrama de Atividade – SITE – Alterar senha



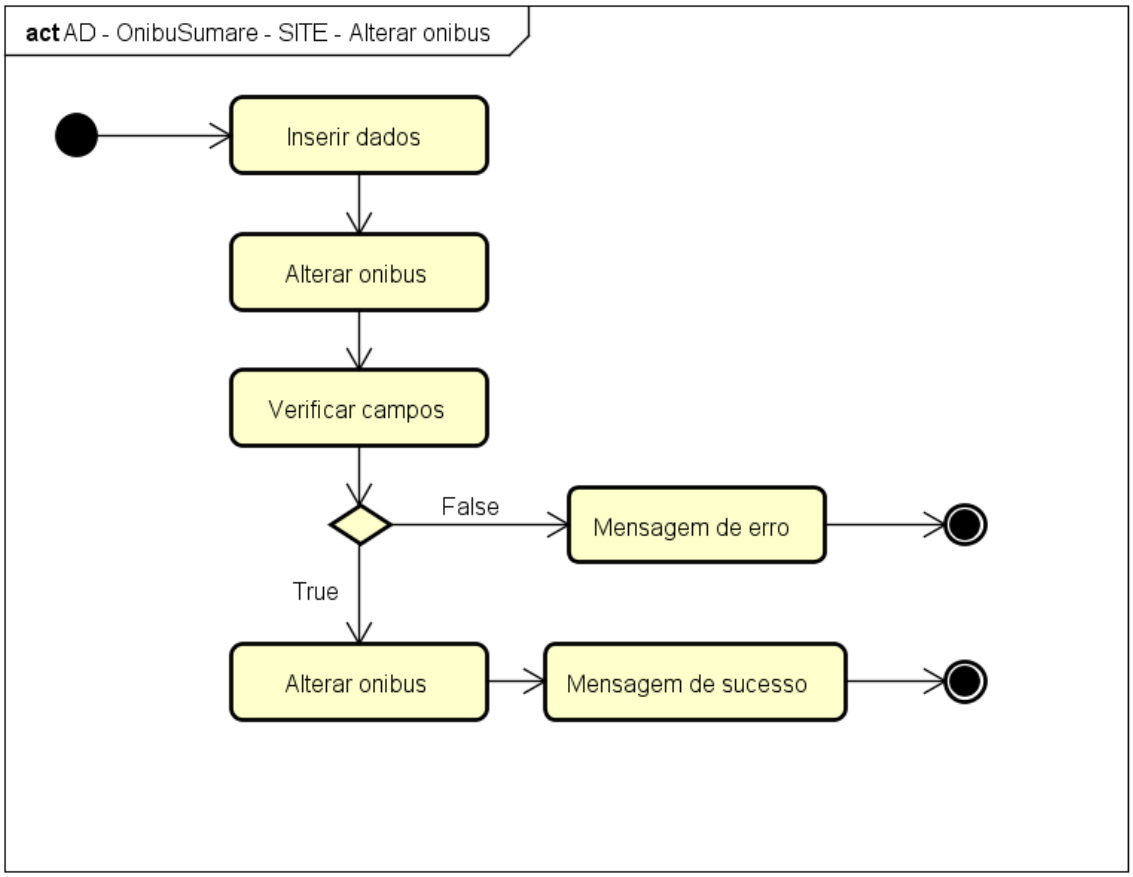

Fonte: Elaborada pelo autor.

Figura 12 – Diagrama de Atividade – SITE – Alterar funcionário



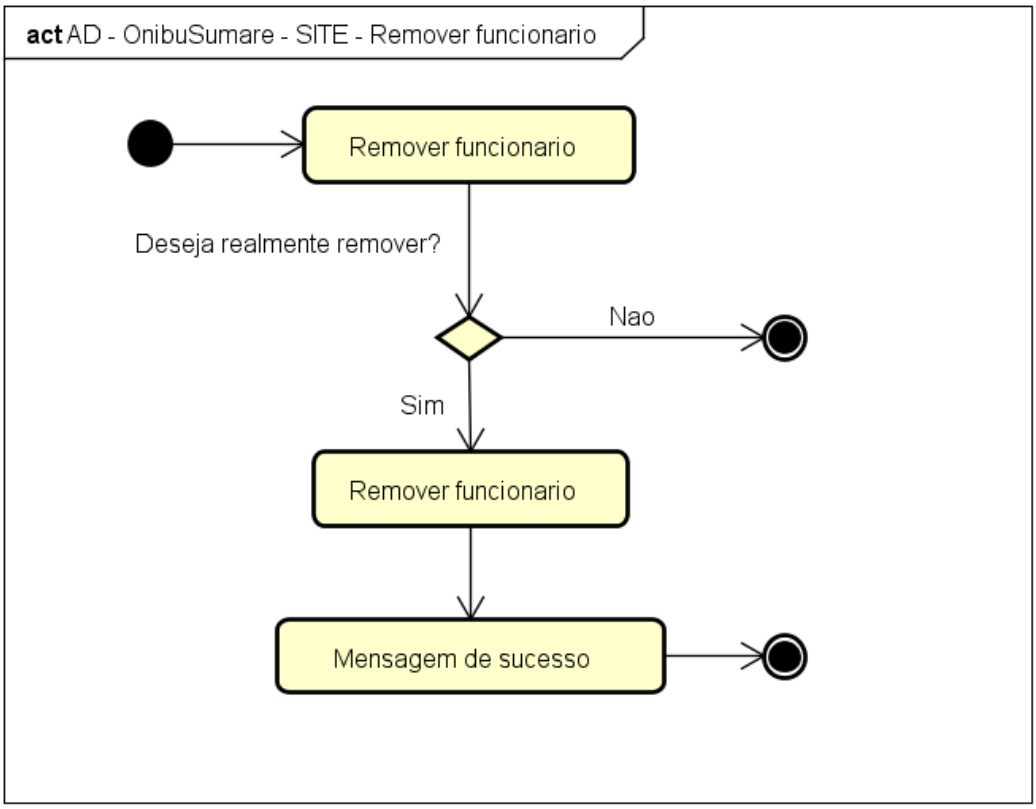

Fonte: Elaborada pelo autor.

Figura 13 – Diagrama de Atividade – SITE – Alterar ônibus



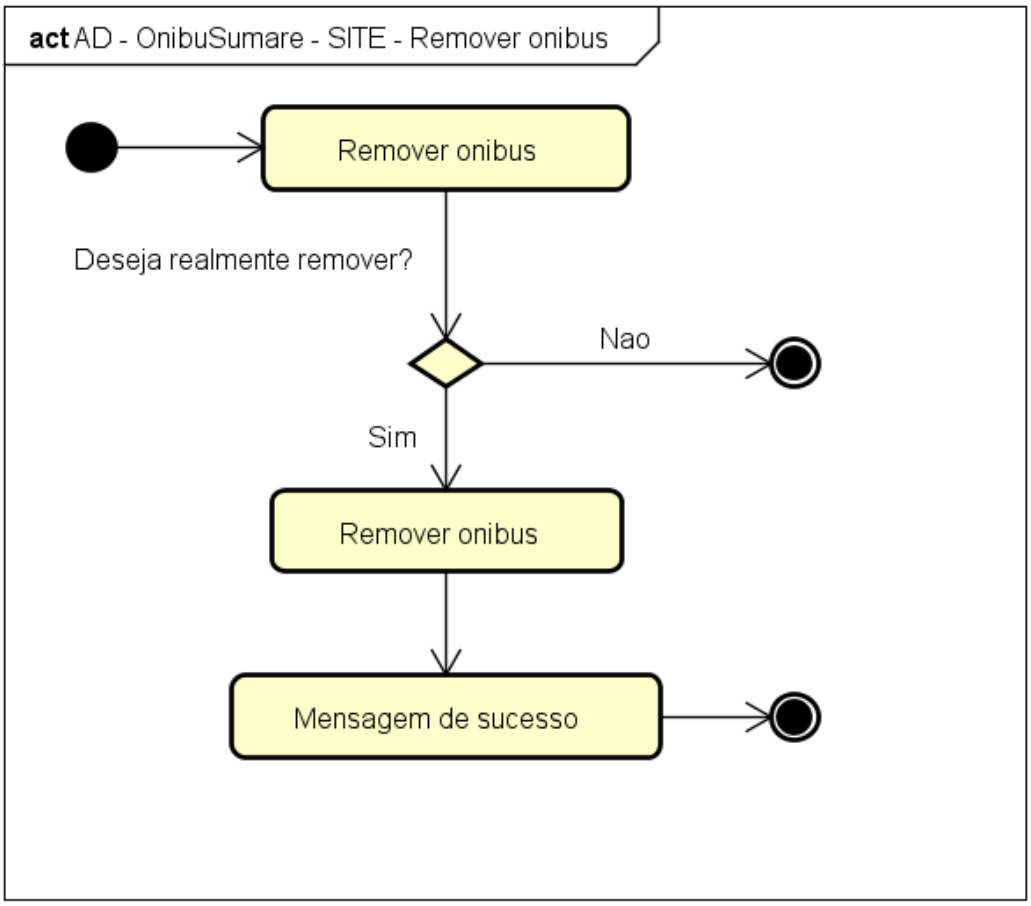

Fonte: Elaborada pelo autor.

Figura 14 – Diagrama de Atividade – SITE – Remover funcionário



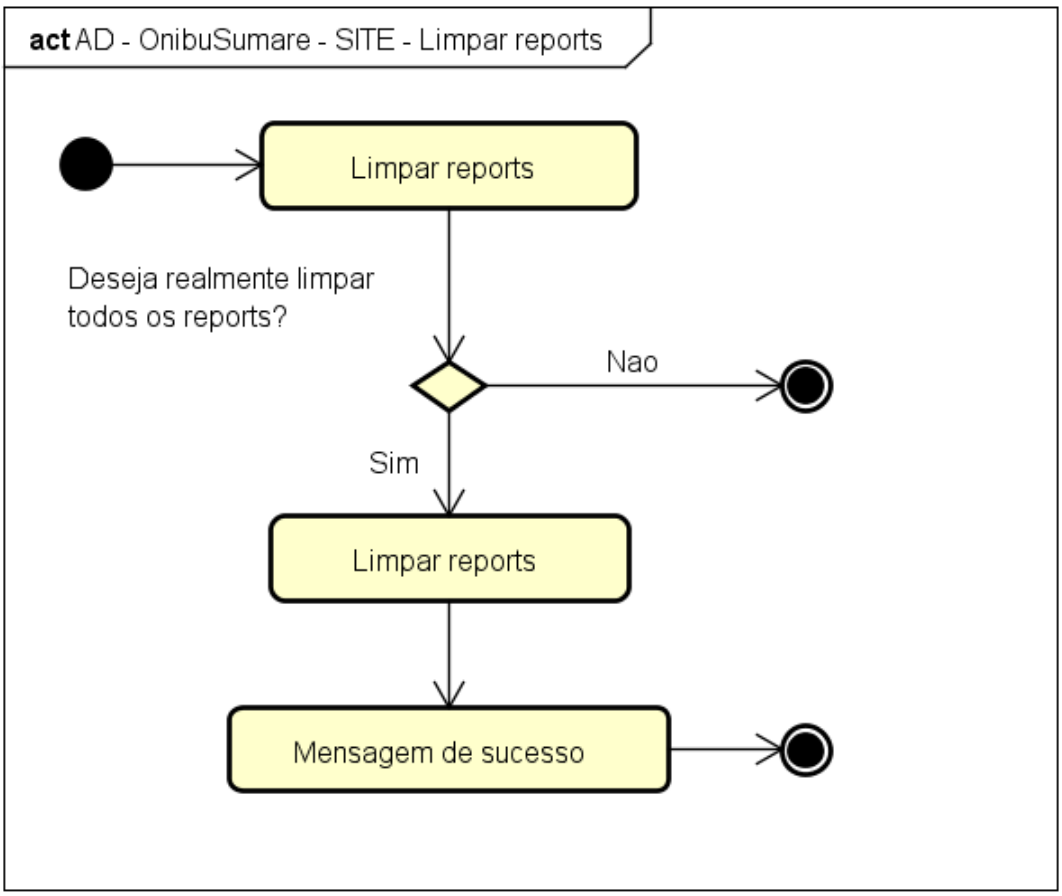

Fonte: Elaborada pelo autor.

Figura 15 – Diagrama de Atividade – SITE – Remover ônibus




Fonte: Elaborada pelo autor.

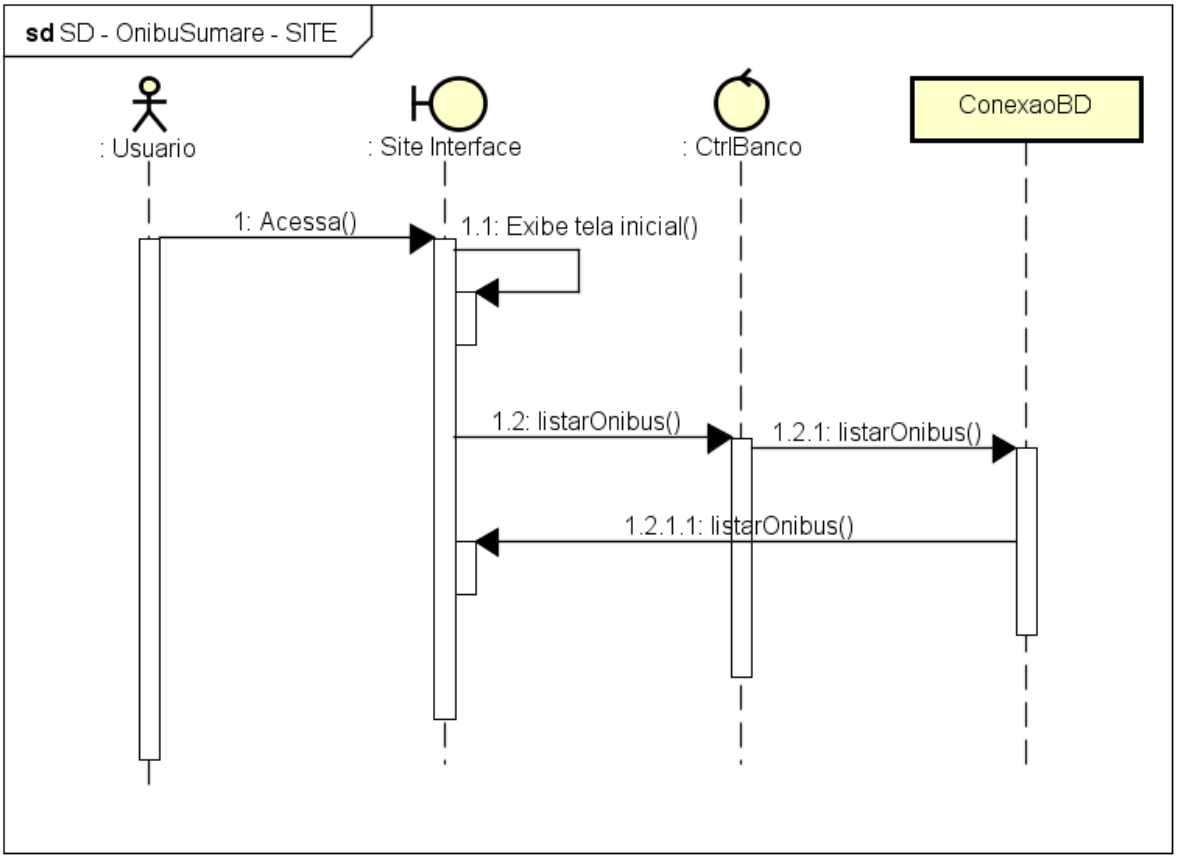
Figura 16 – Diagrama de Atividade – SITE – Limpar *reports*




Fonte: Elaborada pelo autor.

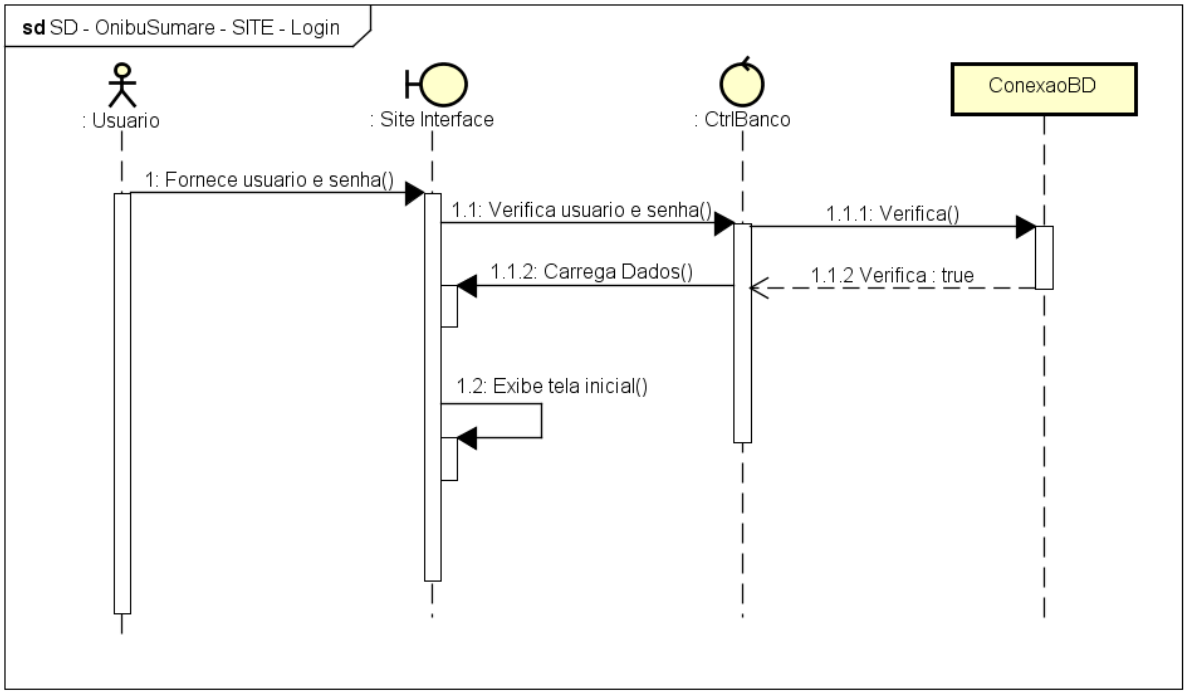
### *4.3.1.4 Diagramas de sequência*

Figura 17 – Diagrama de Sequência – SITE – Tela inicial



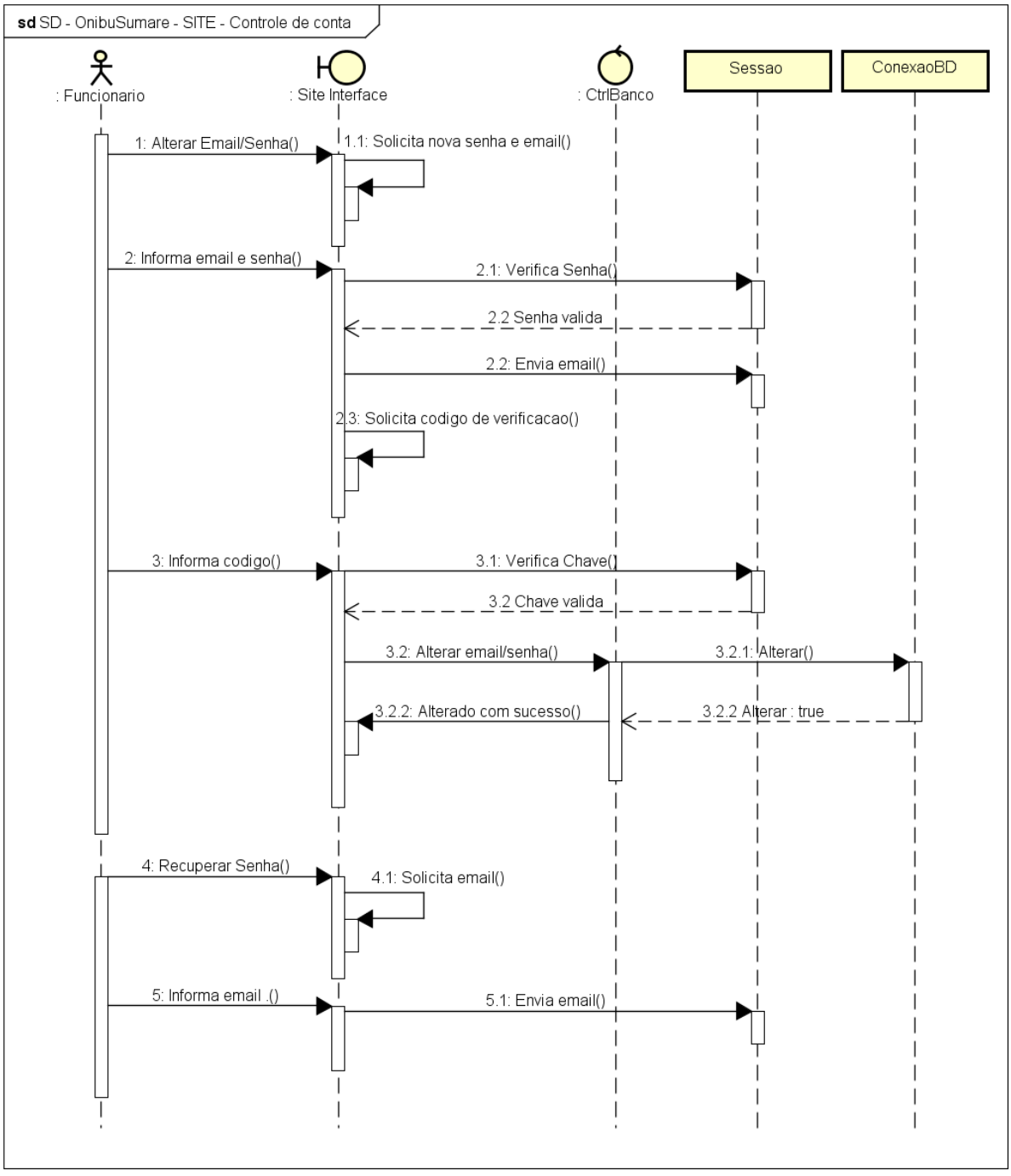

Fonte: Elaborada pelo autor.

Figura 18 – Diagrama de Sequência – SITE - Login



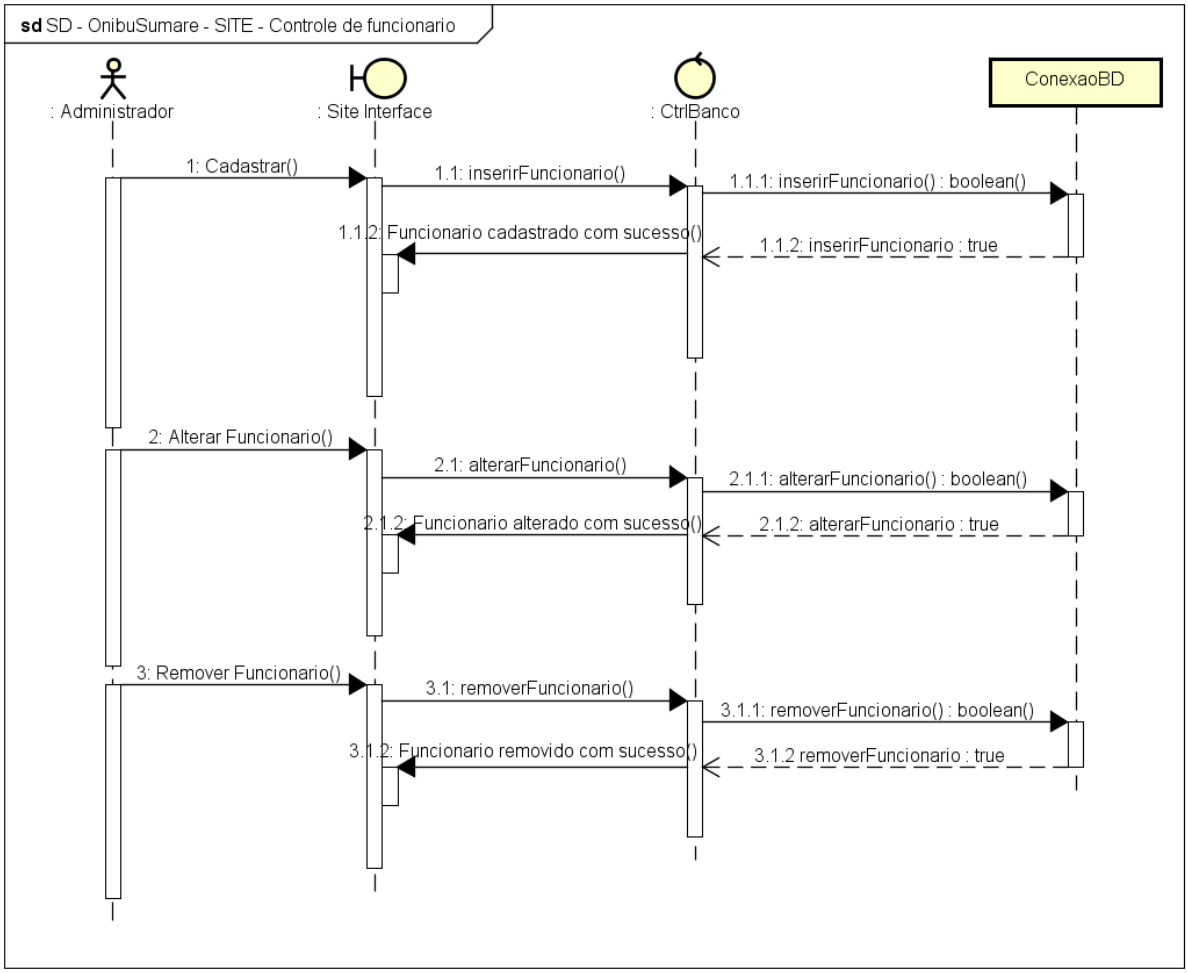

Fonte: Elaborada pelo autor.

Figura 19 – Diagrama de Sequência – SITE – Controle de conta



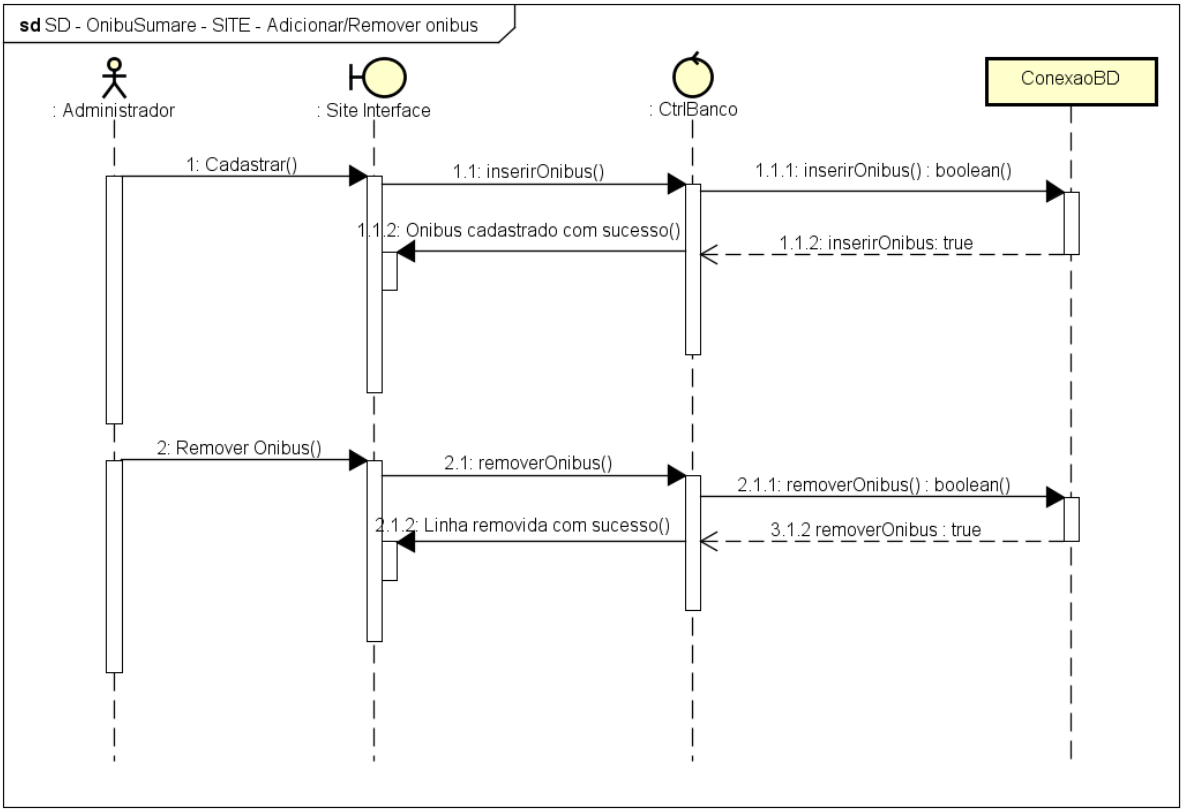

Fonte: Elaborada pelo autor.

Figura 20 – Diagrama de Sequência – SITE – Controle de funcionário



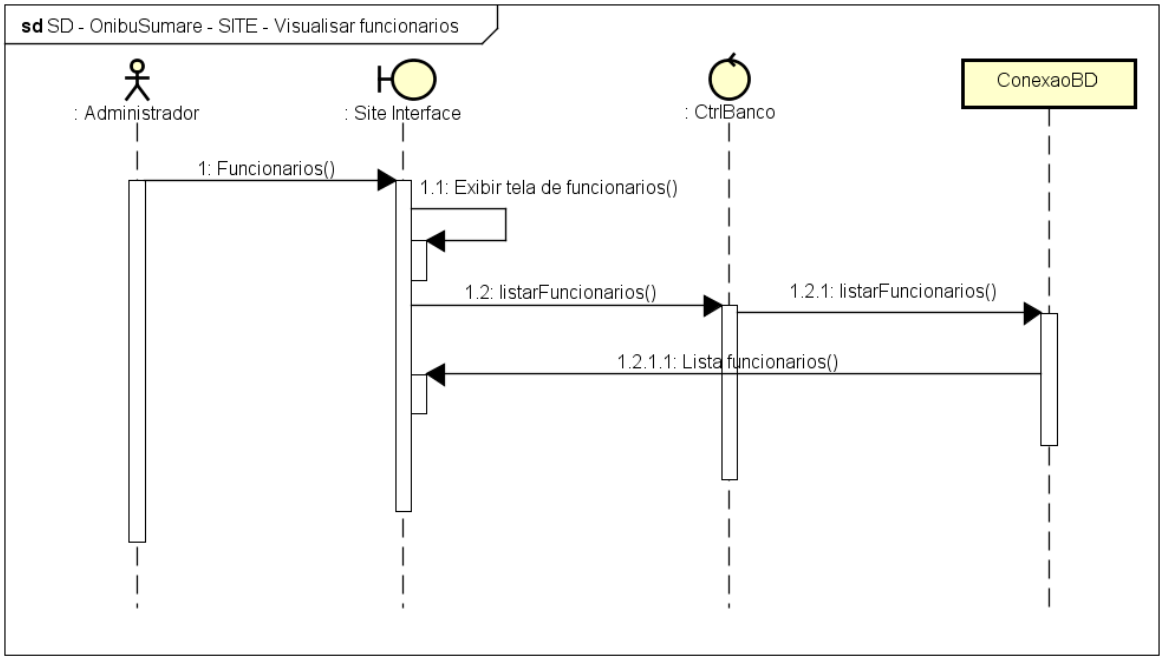

Fonte: Elaborada pelo autor.

Figura 21 – Diagrama de Sequência – SITE – Adicionar/Remover ônibus



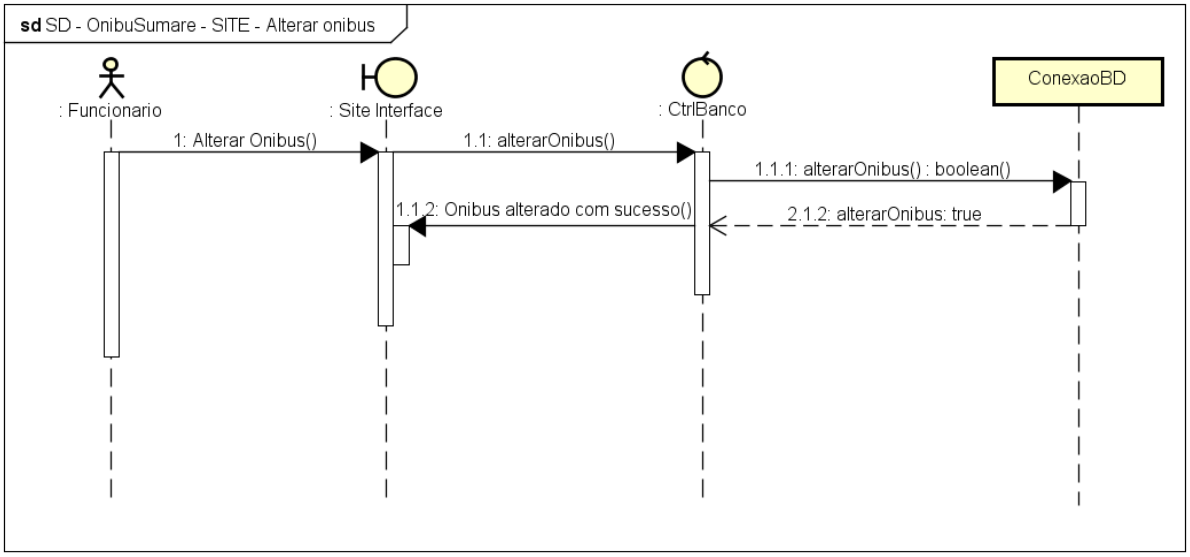

Fonte: Elaborada pelo autor.

Figura 22 – Diagrama de Sequência – SITE – Visualizar funcionários



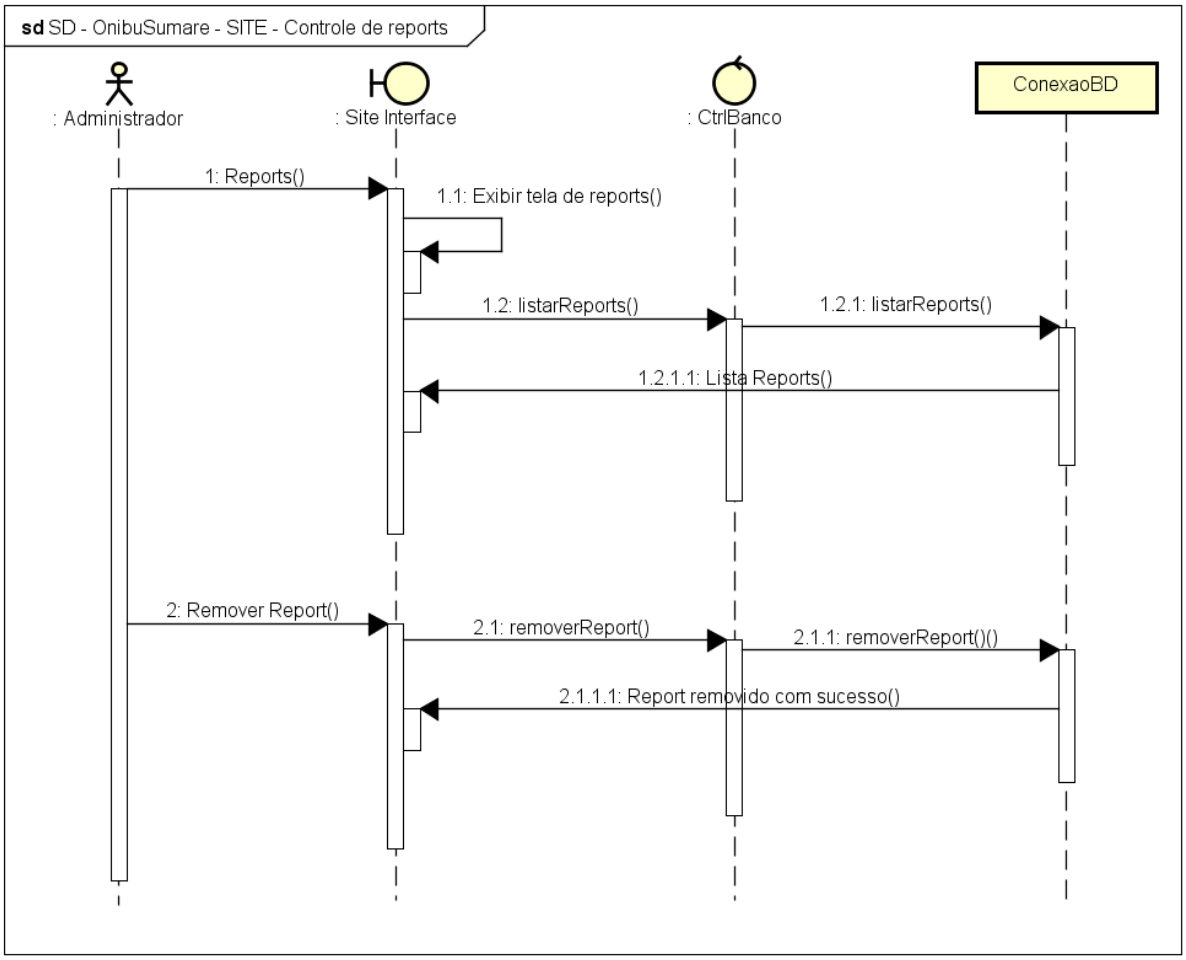

Fonte: Elaborada pelo autor.

Figura 23 – Diagrama de Sequência – SITE – Alterar ônibus




Fonte: Elaborada pelo autor.

Figura 24 – Diagrama de Sequência – SITE – Controle de reports

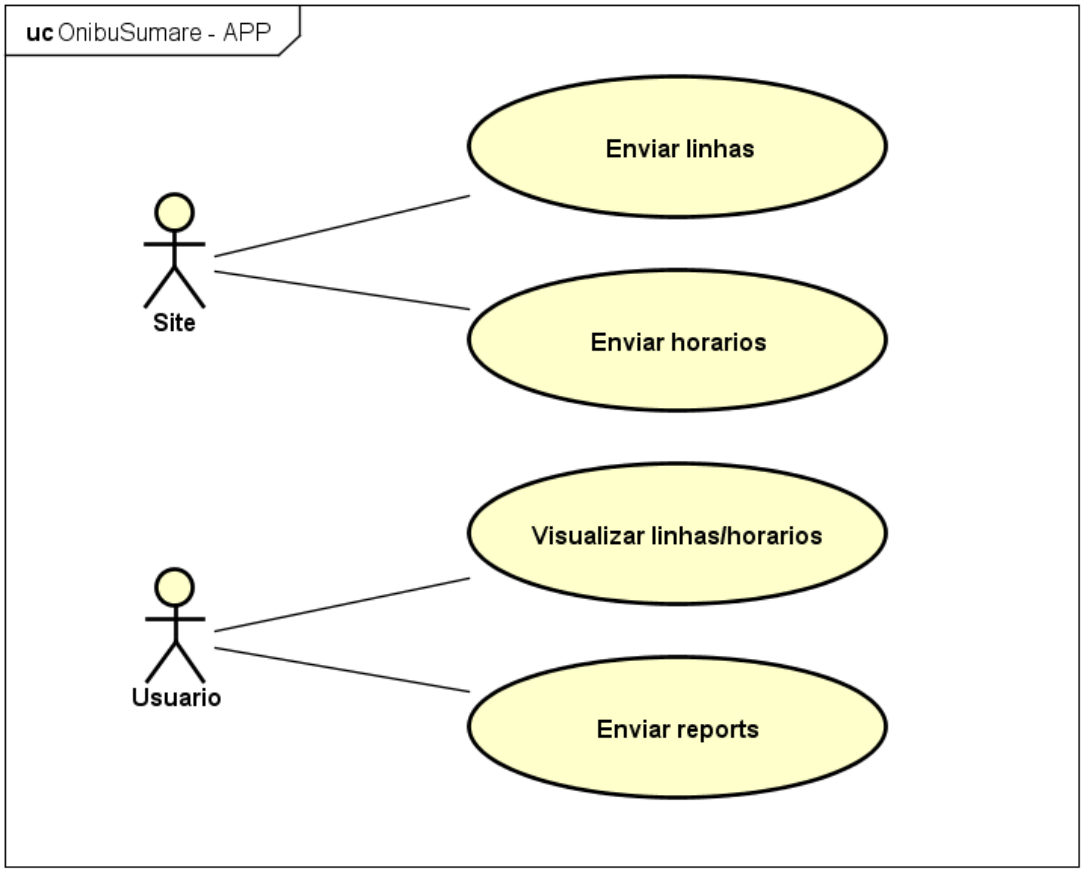



Fonte: Elaborada pelo autor.

### *4.3.2 Aplicativo*

#### *4.3.2.1 Casos de uso*

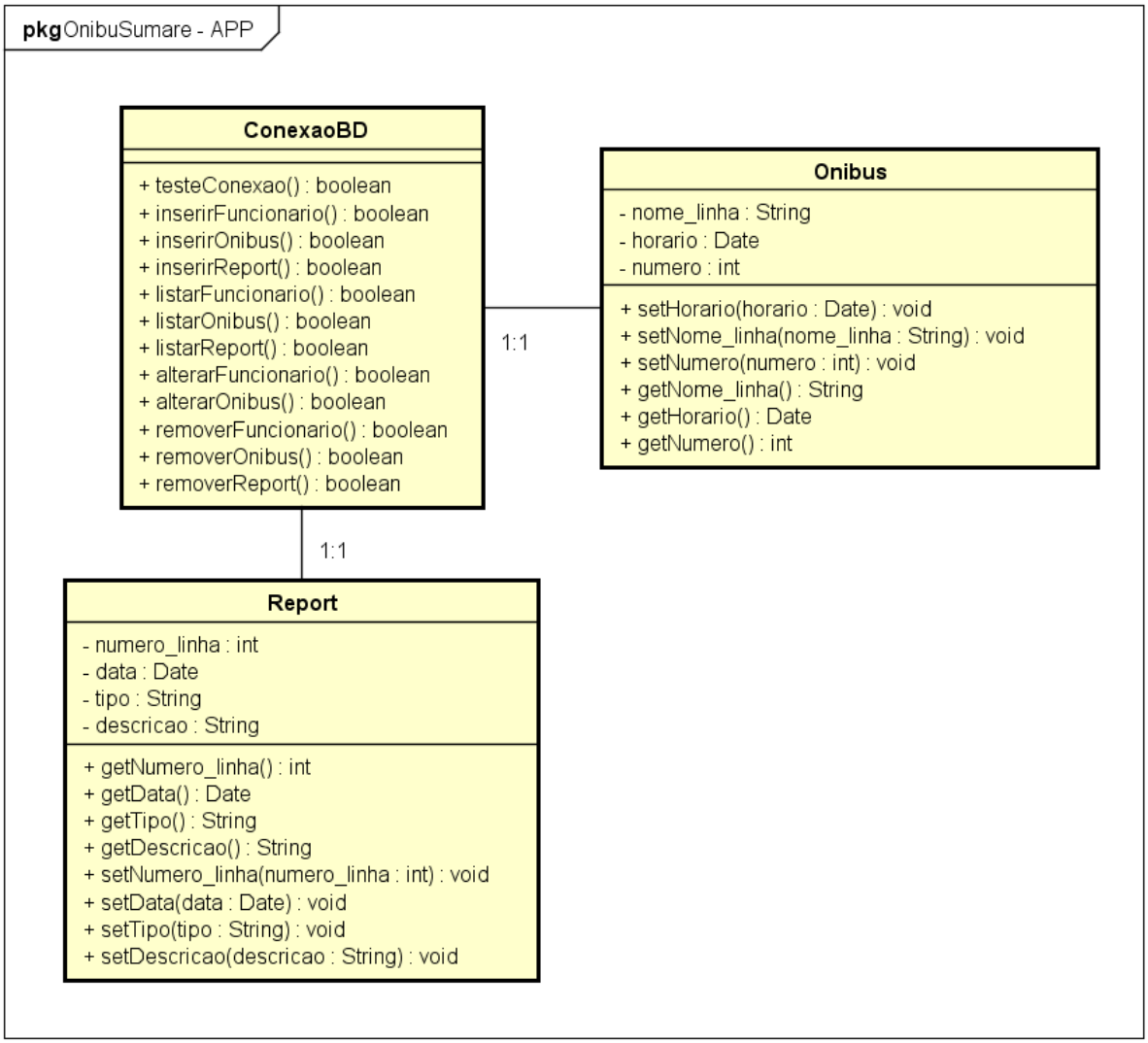
Figura 25 – Casos de Uso – APP




Fonte: Elaborada pelo autor.

*4.3.2.2 Diagrama de classes*

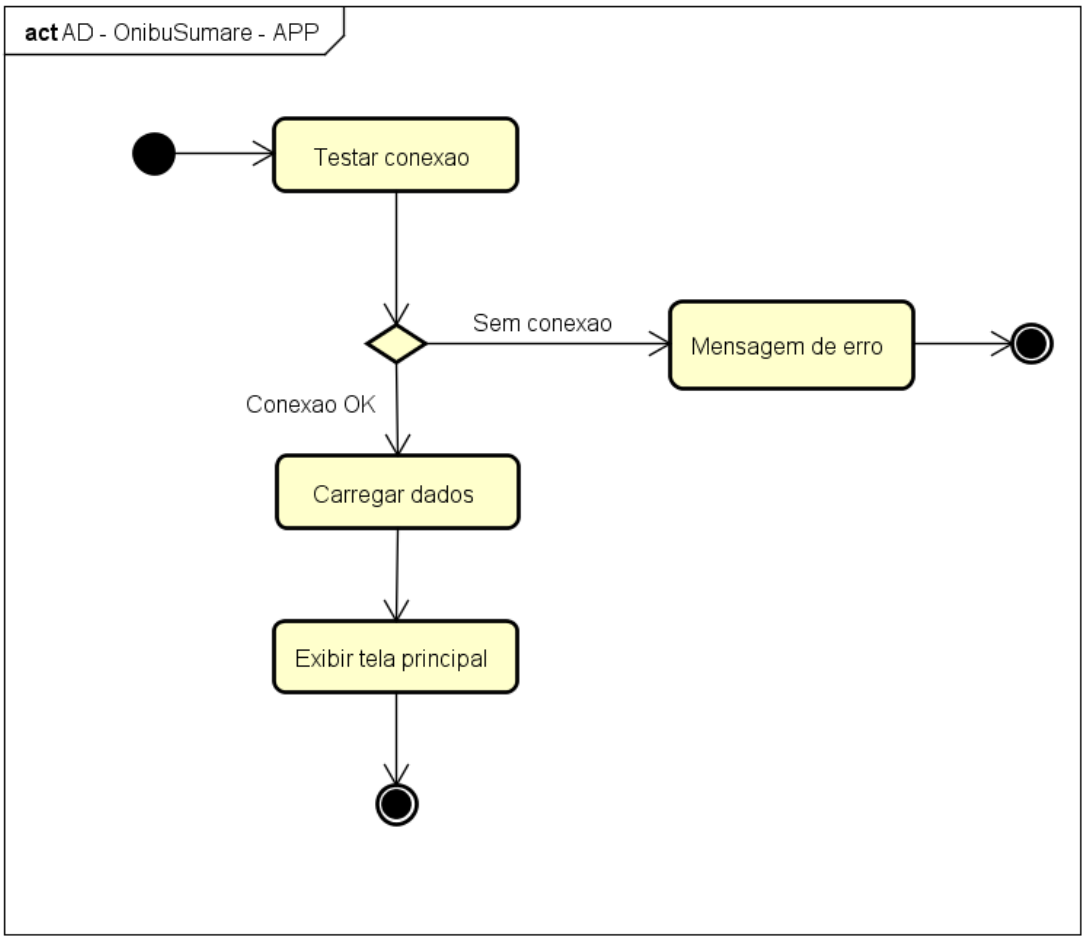
Figura 26 – Diagrama de Classes – APP




Fonte: Elaborada pelo autor.

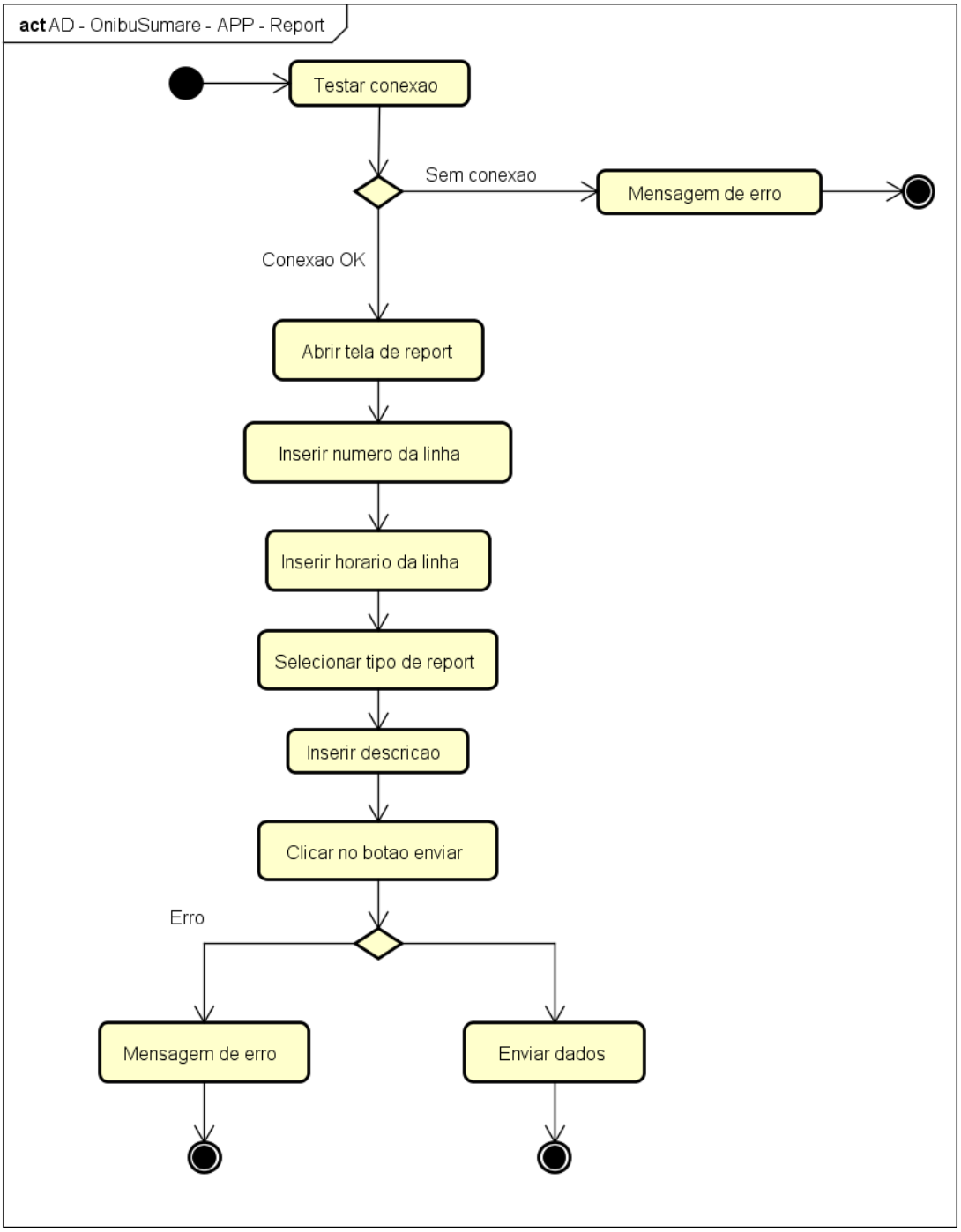
### *4.3.2.3 Diagramas de atividade*

Figura 27 – Diagrama de Atividade - APP - Início




Fonte: Elaborada pelo autor.

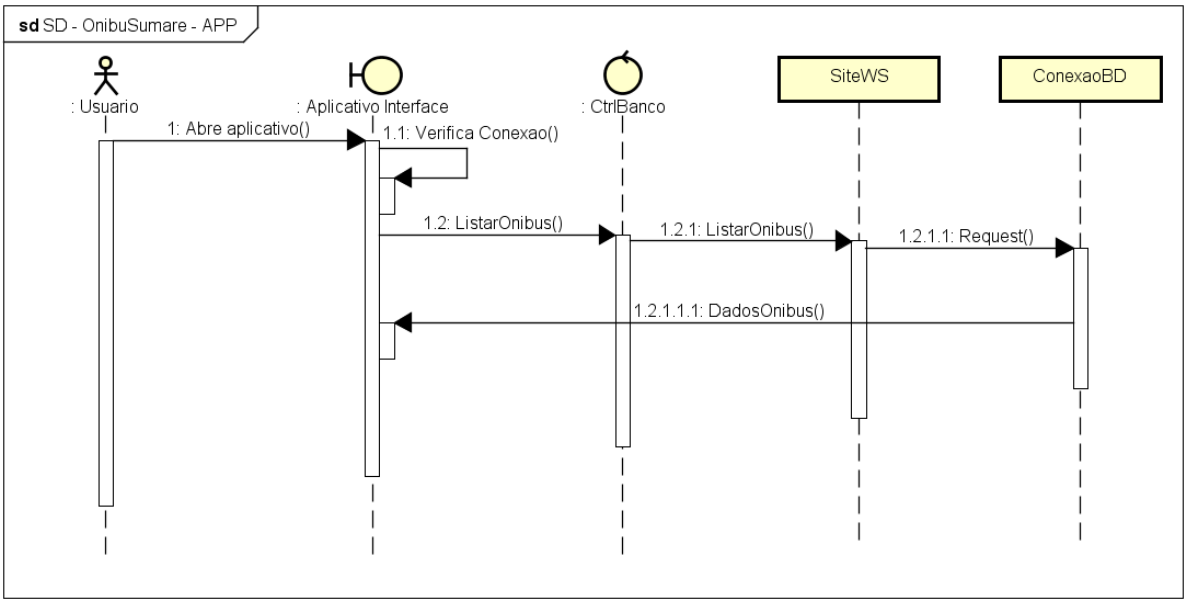
Figura 28 – Diagrama de Atividade – APP - *Report*




Fonte: Elaborada pelo autor.

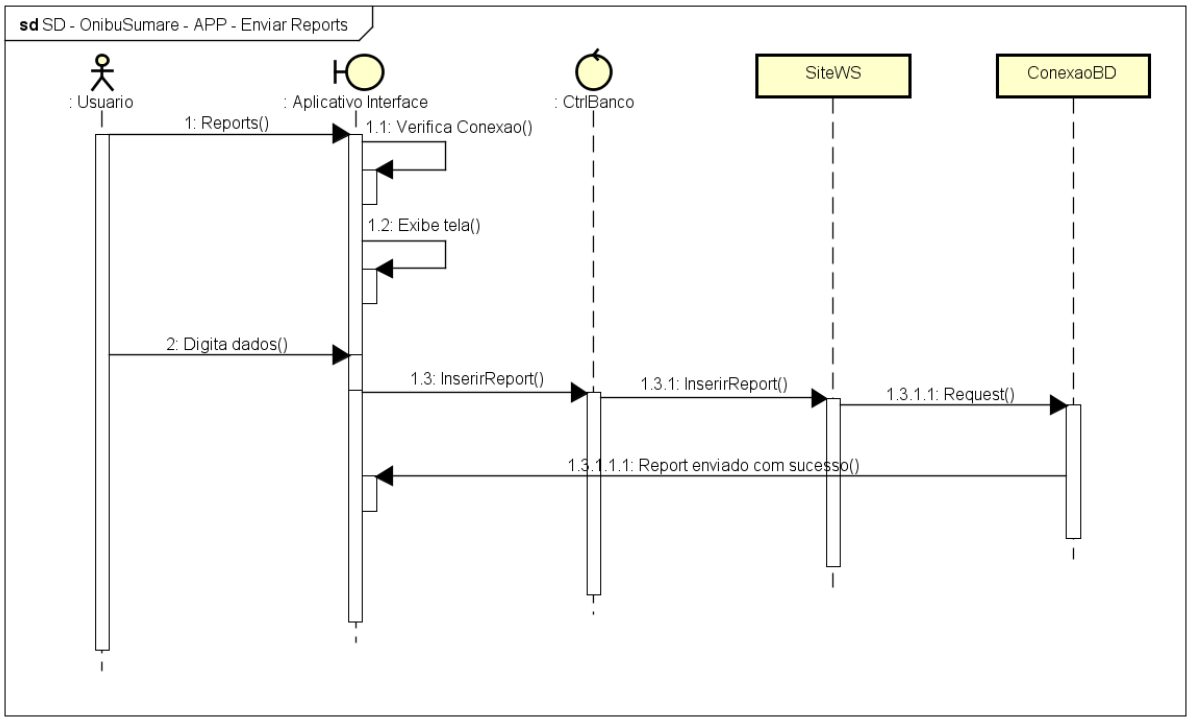
### *4.3.2.4 Diagramas de sequência*

Figura 29 – Diagrama de Sequência – APP - Início




Fonte: Elaborada pelo autor.

Figura 30 – Diagrama de Sequência – APP – Enviar *reports*



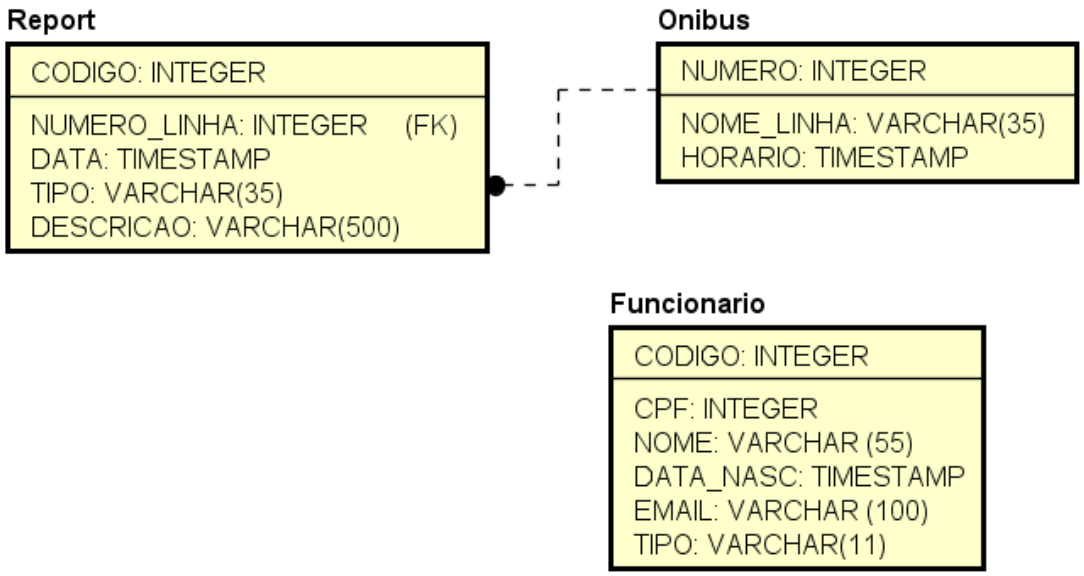

Fonte: Elaborada pelo autor.

### *4.3.3 DER*

O Diagrama Entidade Relacionamento representa graficamente o banco de dados, facilitando o trabalho da equipe e mostrando vários detalhes importantes, tanto para quem analisa o sistema quanto para quem desenvolve.

#### 4.3.3.1 DER do Projeto

Figura 31 – Diagrama Entidade Relacionamento do Projeto




*Fonte: Elaborada pelo autor.*

Os *reports* serão ligados aos ônibus através do campo que armazena o número de linha dos ônibus. A tabela de funcionário tem como função armazenar os dados dos funcionários, e quando efetuado o *login*, o site verificará pelo tipo de funcionário quais funções serão habilitadas.

### *4.3.4 Layouts*

Nessa parte estão alguns dos esboços dos layouts para se utilizar como base. É importante lembrar que esses layouts são esboços, portanto não são como o software irá se parecer em sua versão final.

#### *4.3.4.1 Site*

##### *4.3.4.1.1 Tela inicial*

Figura 32 – *Layout* – SITE – Início




Fonte: Elaborada pelo autor.

Na tela inicial é possível ver a barra de *login* e senha no topo da página. No corpo há informações sobre as linhas de ônibus, irrestritas a qualquer usuário.

*4.3.4.1.2 Telas de controle de funcionários, ônibus e reports*

Figura 33 – *Layout* – SITE – Controle de funcionário

OnibuSumare

Bem vindo, Administrador! Sair

Funcionários
Adicionar Funcionário
Ônibus
Adicionar Ônibus
Reports
Alterar Email/Senha

**Funcionários:**

| Codigo | Nome | CPF | Data de nascimento | Email | Tipo |
|---|---|---|---|---|---|
| 01 | NomeR | 123456789-45 | 04/08/1967 | teste@teste.com | Funcionario |
| 02 | NomeO | 123456789-45 | 28/06/1968 | teste@teste.com | Administrador |
| 03 | NomeB | 123456789-45 | 13/06/1969 | teste@teste.com | Funcionario |
| 04 | NomeN | 123456789-45 | 07/11/1969 | teste@teste.com | Funcionario |
| 05 | NomeO | 123456789-45 | 02/10/1970 | teste@teste.com | Funcionario |
| 06 | Nomel | 123456789-45 | 05/11/1971 | teste@teste.com | Funcionario |
| 07 | NomeG | 123456789-45 | 02/06/1972 | teste@teste.com | Funcionario |
| 08 | NomeL | 921617234-05 | 24/03/1973 | teste@teste.com | Administrador |
| 09 | NomeS | 123456789-45 | 12/09/1975 | teste@teste.com | Administrador |
| 10 | NomeZ | 123456789-45 | 21/01/1977 | teste@teste.com | Funcionario |
| 11 | NomeE | 123456789-45 | 30/11/1979 | teste@teste.com | Administrador |
| 12 | NomeO | 123456789-45 | 21/03/1983 | teste@teste.com | Funcionario |
| 13 | NomeH | 123456789-45 | 07/09/1987 | teste@teste.com | Funcionario |
| 14 | Nomel | 123456789-45 | 28/03/1994 | teste@teste.com | Administrador |
| 15 | NomeE | 123456789-45 | 07/11/2014 | teste@teste.com | Funcionario |

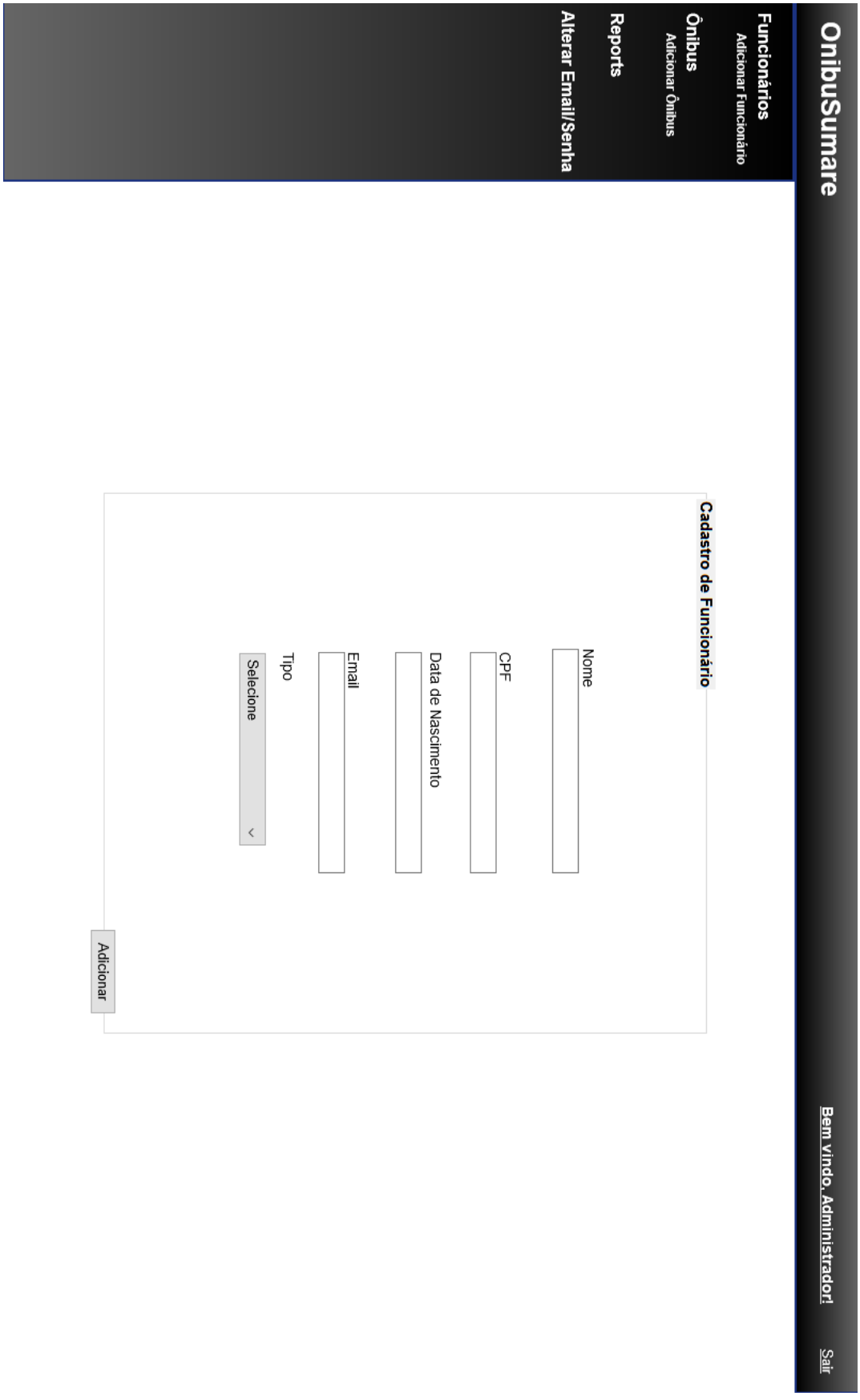
Adicionar Funcionário

Fonte: Elaborada pelo autor.

Nessa tela é possível ver a tabela com os funcionários, onde uma linha carrega as informações sobre cada funcionário, e ao final das linhas há botões para alterá-los ou excluí-los. Na parte inferior há um botão para adição de funcionário. Há também um menu lateral com algumas funções exclusivas do administrador do sistema. As telas de administração dos ônibus e de *reports* funcionam da mesma maneira.

*4.3.4.1.3 Tela de adição de funcionário*

Figura 34 – *Layout* – SITE – Adição de funcionário

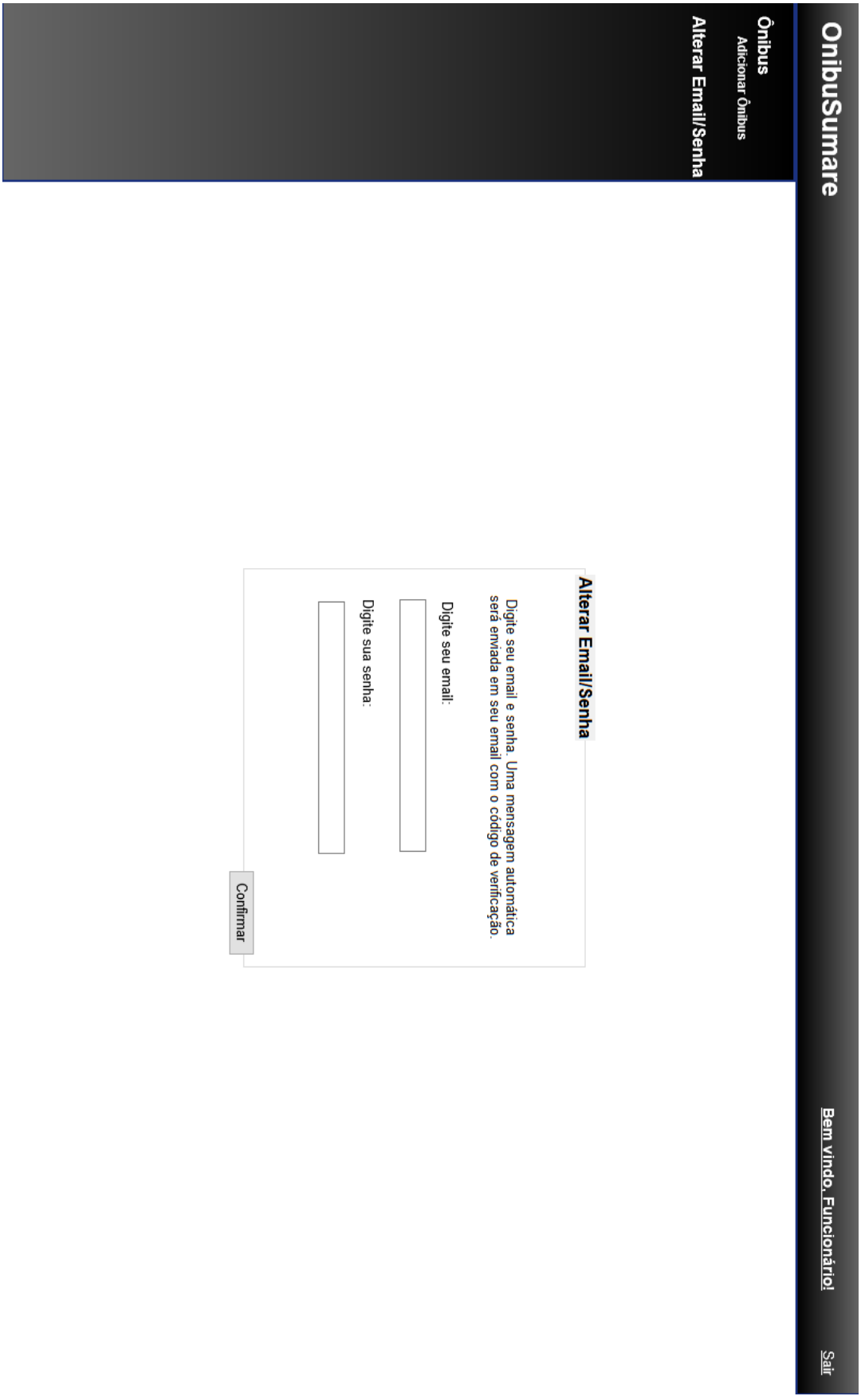



Fonte: Elaborada pelo autor.

Nessa figura é exibida a tela de adição de funcionários. Para adicionar um funcionário é preciso escrever as informações sobre ele, como o nome, CPF, entre outros. Após digitar as informações, é selecionado se o novo funcionário terá permissão de administrador ou não.

*4.3.4.1.4 Tela de alteração de e-mail/senha*

Figura 35 – *Layout* – SITE – Alteração de e-mail/senha




Fonte: Elaborada pelo autor.

Nessa tela se faz a alteração de e-mail/senha. É preciso digitar o e-mail e a senha, e após o sistema verificar se a senha é válida, envia um e-mail com um código de verificação para o e-mail fornecido pelo funcionário.

### *4.3.4.2 Aplicativo*

#### *4.3.4.2.1 Tela inicial*

Figura 36 – *Layout* – APP – Início


Fonte: Elaborada pelo autor

*Na tela inicial ficam listadas as linhas, e pressionando sobre uma linha é possível consultar seus horários.*

#### *4.3.4.2.2 Tela de reports*

Figura 37 – *Layout* – APP - *Report*


Fonte: Elaborada pelo autor.

Na tela de *report* (ou denúncia), o usuário fornece as informações sobre a linha de ônibus e descreve o motivo. A denúncia é enviada ao site, onde o administrador poderá consultá-la.

# 5 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi desenvolver o projeto de um site e aplicativo para a consulta de linhas urbanas de ônibus em Sumaré-SP, visto que apesar de ser uma cidade com mais de 200.000 habitantes, não tem meio eletrônico para verificação desses horários. A ferramenta não só disponibiliza os horários das respectivas linhas de ônibus, como também oferece um sistema de denúncias para os usuários do aplicativo.

O trabalho atingiu seu objetivo e o desenvolvimento do projeto foi concluído sem muitos problemas. A principal dificuldade foi no aprendizado sobre aplicativos mobile e a conexão entre aplicativo e site através de *Web Service*, justamente o motivo pelo qual esses elementos foram escolhidos para este projeto.

## 5.1 Sugestões para projetos futuros

Durante o desenvolvimento do projeto foram verificadas algumas possibilidades para projetos futuros:

• Implantação de API para localização de ônibus em tempo real utilizando GPS;

• Persistência de dados em modo *off-line*;

• Sistema de feedback aos *reports*;

• Notificação de veículo quebrado;

• Otimização dos processos e estrutura do projeto;

• Desenvolvimento do software;

• Adição de mapa com a rota das linhas, ou lista de lugares em que a linha passa;

• Integração ao site da empresa.

**REFERÊNCIAS**

AMERICAN DIALECT SOCIETY. “App” voted 2010 word of the year by the American Dialect Society (UPDATED) – 2011. Disponível em: <http://www.americandialect.org/app-voted-2010-word-of-the-year-by-the-american-dialect-society-updated>. Acesso em: 17 Jun. 2015 18h36min.

FALBO, Ricardo de Almeida. Engenharia de Software: Notas de Aula – 2005. Disponível em: <http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf >. Acesso em: 18 Jun. 2015 19h36min.

FARIA, Fernanda B. et al. Evolução e Principais Características do IDE Eclipse – 2010. Disponível em: < http://www.enacomp.com.br/2010/cd/artigos/completos/enacomp2010_23.pdf>. Acesso em: 14 Out. 2015 22h14min.

IBGE. Infográficos: dados gerais do município – 2015. Disponível em: <http://www.cidades.ibge.gov.br/painel/painel.php?lang=&codmun=355240&search=sao-paulo%7Csumare%7Cinfograficos:-dados-gerais-do-municipio>. Acesso em: 14 Jun. 2015 21h36min.

IDC. Smartphone OS Market Share, 2015 Q2 – 2015. Disponível em: http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em: 24 Set. 2015 15h35min.

JACYNTHO, Mark Douglas de Azevedo. Processos de Desenvolvimento para Aplicações Web – 2009. Disponível em: <ftp://ftp.inf.puc-rio.br/pub/docs/techreports/09_23_jacyntho.pdf> Acesso em: 03 Set. 2015 19h47min.

LECHETA, Ricardo R. **Google Android:** Aprenda a criar aplicações para dispositivos móveis com o Android SDK. 3.ed. São Paulo: Novatec Editora, 2013.

MELLO, Emerson Ribeiro de. et al. Segurança em serviços web – 2006. Disponível em: <http://www.researchgate.net/profile/Emerson_Mello/publication/229019068_Segura

na_em_Servios_Web/links/02e7e522f4b7e99245000000.pdf>. Acesso em: 16 Jun. 2015 16h45min.

MILANI, André. **Programando para iPhone e iPad:** Aprenda a construir aplicativos para iOS. São Paulo: Novatec Editora, 2012.

OURO VERDE. 2015. Disponível em: <http://www.ouroverde.com.br>. Acesso em: 07 Jun. 2015 21h20min.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço da. **Android Para Desenvolvedores.** Rio de Janeiro: Brasport, 2009.

PHONE ARENA. Android's Google Play beats App Store with over 1 million apps, now officialy largest – 2013. Disponivel em: http://www.phonearena.com/news/Androids-Google-Play-beats-App-Store-with-over-1-million-apps-now-officially-largest_id45680>. Acesso em: 10 Set. 2015 20h04min.

PRESSMAN, Roger S. **Engenharia de Software:** Uma Abordagem Profissional. 7. ed. Porto Alegre: AMGH Editora Ltda, 2011. 778p.

SWEBOK V3. Guite to the Software Engineering Body of Knowlegment - 2014. Disponível em: < http://www.computer.org/web/swebok/v3 >. Acesso em: 16 Jun. 2015 19h54.

TAROUCO, Fabricio. A Metrópole Comunicacional e a Popularização dos Apps para Dispositivos Móveis – 2013. Disponível em: < http://coral.ufsm.br/sipecom/2013/wp-content/uploads/gravity_forms/1-997169d8a192ed05af1de5bcf3ac7daa/2013/09/A-metropole-comunicacional-o-e-a-popularizacao-dos-apps.pdf> Acesso em: 17 Jun. 2015 17h47min.

TAVARES, Ana Lúcia de Oliveira. et al. Engenharia de Software: uma visão geral – 2007. Disponível em: <http://drc.objectis.net/faculdade-anhanguera-de-anapolis/pos-graduacao/engenharia-de-software/material/07-artigo-eng-sof.pdf>. Acesso em: 18 Jun. 2015 18h40min.

TECNOBLOG. Android supera 80% das vendas de smartphones e Windows Phone continua avançando – 2013. Disponível em:

<https://tecnoblog.net/145067/vendas-smartphones-terceiro-trimestre-2013/>. Acesso em 10 Set. 2015 20h09min.

ZENETI JUNIOR, Luis Antonio; VIDAL, Antonio Geraldo da Rocha. Construção de sistemas de informação baseados na Tecnologia Web – 2006. Disponível em: <www.rausp.usp.br/download.asp?file=V4103232.pdf>. Acesso em: 16 Jun. 2015 17h25min.