LEONARDO CARVALHO DE OLIVEIRA

RELATÓRIO DE ESTÁGIO: DESENVOLVIMENTO DE UM SERVIÇO WEB PARA LER E MODIFICAR CONFIGURAÇÕES DO SITEF

LAVRAS – MG

2019

LEONARDO CARVALHO DE OLIVEIRA

RELATÓRIO DE ESTÁGIO: DESENVOLVIMENTO DE UM SERVIÇO WEB PARA LER E MODIFICAR CONFIGURAÇÕES DO SITEF.

Relatório de Estágio Supervisionado apresentado ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências para a obtenção do título de Bacharel em Ciência da Computação

APROVADA em 28 de Junho de 2019.

Dra. Renata Teles Moreira	UFLA
Dra. Marluce Rodrigues Pereira	UFLA
Dr. Bruno de Abreu Silva	UFLA

Prof. Dra. Renata Teles Moreira

Orientadora

LAVRAS – MG
2019

Dedico esta monografia à minha familia.

AGRADECIMENTOS

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

Arrisque-se! Toda vida é um risco. O homem que vai mais longe é geralmente aquele que está disposto a fazer e a ousar. O barco da 'segurança' nunca vai muito além da margem.

(Dale Carnegie)

RESUMO

Atualmente, o mercado de meios de pagamento se define em grandes sistemas concentradores de transações de fundos, e esses mesmos sistemas devem ser configurados. Essas configurações ocorriam de forma local no servidores, ou seja, existiam softwares de configuração locais, que usavam de arquivos para configuração. Com o passar dos tempos, a demanda de mercado mudou e começou a existir um configurador remoto, que poderia configurar um sistema de *TEF* (Transferência eletrônica de fundos)sem estar localmente no servidor. Com isso o presente trabalho apresenta detalhes sobre o desenvolvimento de um serviço web que atende as requisições de vários aplicativos que buscam ler e modificar dados de configuração do SiTef, foram utilizados o *Play Framework* que possui a capacidade de produzir aplicações com *Scala*, *HTML*, *JavaScript*, *Docker*. O sistema está em constante evolução e já produz efeitos em produção, muitas configurações realizadas de forma remota, ja utilizam da aplicação para interface com o sistema de *TEF*.

Palavras-chave: *Play Framework. HTML. Docker. JavaScript. TEF. Scala.*

ABSTRACT

Currently, the payment medium market is defined in large systems concentrators of funds transactions, and these same systems must be configured. These settings occurred locally on the servers, that is, there were local configuration software, which used files for configuration . Over time, market demand has changed and a remote configurator has started, which could set up a ETF (Electronic Transfer of Funds) system without being locally on the server. This paper presents details about the development of a web service that answers the requests of several applications that seek to read and modify data of the SiTef configuration, the Play Framework was used that has the ability to produce applications with Scala, HTML,JavaScript, Docker. The system is in evolution and already produces production effects, many configurations carried out remotely, already use the application to interface with the ETF system.

Keywords: *Play Framework. HTML. Docker. JavaScript. ETF. Scala.*

LISTA DE FIGURAS

Figura 2.1 – Modelo Incremental 12
Figura 2.2 – Fluxo de uma transação eletrônica 14
Figura 2.3 – SiTef 15
Figura 2.4 – Fluxo do *SiTef* 15
Figura 2.5 – IntSitef 16
Figura 3.1 – FDW 20
Figura 3.2 – Exemplo HTML 25
Figura 3.3 – Exemplo HTML utilizando CSS 27
Figura 3.4 – DashBoard do GrayLog 31
Figura 4.1 – Página Home 33
Figura 4.2 – Cadastro SiTef 34
Figura 4.3 – Monitoração 36
Figura 4.4 – Status 40
Figura 4.5 – Configurador de Certificados 41

SUMÁRIO

1 INTRODUÇÃO 9
1.1 O Estágio 9
1.2 Visão geral das atividades 9
1.3 Organização do trabalho 10
2 EMPRESA 11
2.1 A empresa Software Express 11
2.1.1 Métodos de trabalho 11
2.2 Mercado de meios de pagamento 13
2.3 O Sitef 14
2.3.1 Funcionamento do SiTef 15
2.3.2 O IntSitef 16
2.3.3 Principais Funcionalidades 16
3 REFERENCIAL TEÓRICO 18
3.1 Modelagem e Estruturas 18
3.1.1 Padrão MVC 18
3.1.2 WebService 19
3.1.3 Integração de dados com Foreign Data Wrappers 20
3.2 Tecnologias *Back-end* 21
3.2.1 Play Framework 21
3.2.2 Scala 22
3.3 Tecnologias *fron-end* 23
3.3.1 HTML 24
3.3.2 CSS 25
3.3.3 JavaScript 27
3.4 Ferramentas Auxiliares e *Builders* 28
3.4.1 SBT 28
3.4.2 GrayLog 30
3.4.3 PostMan 31
3.4.4 Docker 31
3.4.5 Mercurial 32
3.4.6 Visual Studio Code 32

4 ATIVIDADES DESENVOLVIDAS 33
4.1 Cadastro de um servidor SiTef 33
4.2 Monitoração 35
4.3 Inclusão de Instituição Financeira 36
4.4 CRUD de Empresa 37
4.5 CRUD de Grupo de empresas 38
4.6 Visualizar Status 40
4.7 Configuração de Certificados 41
4.8 Aproveitamento de Logs 42
4.9 Formalização de Erros 42
5 CONSIDERAÇÕES FINAIS 43
6 REFERÊNCIAS BIBLIOGRÁFICAS 44

1 INTRODUÇÃO

O presente trabalho tem como objetivo apresentar as técnicas e as tecnologias desenvolvidas no projeto IntSiTef, em um estágio realizado na empresa Software Express, no período de maio de 2018 à Outubro de 2018.

1.1 O Estágio

Com um mercado cada vez mais competitivo, a necessidade de se ter uma qualificação profissional faz a diferença na hora de conquistar um emprego desejado. Portanto, com esta necessidade de trabalho e de demanda do mercado de *TI*, as empresas estão em busca de profissionais capacitados e aptos a aplicar os seus conhecimentos na busca de soluções de problemas, com foco no melhor para empresa.

O estágio tem como princípio viabilizar experiências profissionais aos estagiários, buscando demonstrar a prática do ensino no dia-a-dia, valorizando os momentos de experiência em situações concretas ao dia-a-dia do profissional. Neste relatório de estágio, estão descritas as atividades que foram realizadas durante o estágio obrigatório no período de maio de 2018 a outubro de 2018,com atuação na área de desenvolvimento de software, trabalhando com as novas soluções para os problemas de inconsistência de dados, falta de formalização e padronização, das configurações do SiTef (Solução Inteligente de Transferência de Fundos).

Entre muitas aplicações desenvolvidas na empresa Software Express, o que este trabalho aborda é um Sistema de Acesso das Configurações do *SiTef* , o *IntSitef*, que tem como objetivo viabilizar o acesso de dados de configuração do *SiTef*. Dessa maneira, o presente trabalho tem como objetivo apresentar as técnicas e as tecnologias utilizadas no projeto *IntSiTef*.

1.2 Visão geral das atividades

Durante o período de estágio, o estagiário atuou na equipe de Desenvolvimento do produto *SiTef* e as atividades realizadas foram:

- Criação do banco de dados.
- Implementação da comunicação com banco de dados.
- Estimação de esforço para a realização das atividades.
- Implementação de código fonte utilizando a linguagem Scala.

- Dupla validação de código.
- Resolução de problemas que foram detectados ao decorrer das atividades feitas.
- Liberação de pacote e documentação do projeto.

1.3 Organização do trabalho

Além do capítulo de introdução, este trabalho está organizado da seguinte forma: O capítulo 2 detalha como é a empresa *Software Express*, os produtos que estão presentes no projeto, e como são os métodos de trabalho da empresa. No capítulo 3 foi desenvolvido o Referencial Teórico, onde são apresentados os mais importantes conceitos, justificativas e características abordadas no trabalho. O capítulo 4 tem como objetivo mostrar e explicar as atividades desenvolvidas pelo estagiário no período de estágio e alguns exemplos em produção da aplicação *IntSitef*. O capítulo 5 remete aos resultados obtidos com todas as atividades desenvolvidas no período de estágio. E por fim no capítulo 6 é citado as referências bibliográgicas utilizadas em todo o trabalho.

2 EMPRESA

Este capítulo tem como objetivo apresentar a empresa Software Express e as aplicações importantes para o desenvolvimento da aplicação *IntSitef*.

2.1 A empresa Software Express

Software Express é uma empresa com enfoque em soluções de softwares e serviços voltados a transações eletrônicas, meios de pagamentos e comunicação de dados. A empresa preza pelo atendimento às necessidades de seus clientes e parceiros, de forma ágil, personalizada e com qualidade, criando relações de confiança e duradouras, contribuindo, assim, com o desenvolvimento sócio-econômico do mercado onde atua.

A empresa Software Express está localizada na Avenida Paulista, na região Central da cidade de São Paulo – SP, e tem atuação em soluções para transações eletrônicas de fundos, conectando as duas pontas envolvidas em uma Transferência Eletrônica de Fundos (TEF). Em uma ponta tem-se a loja e todos os mais diferentes tipos de automações comerciais (PDV, POS, Totem, um Site), além de muitas outras formas de se comunicar com o cliente. Na outra ponta tem-se, os mais diferentes tipos de Redes Autorizadoras, como por exemplo, Cielo, Redecard, FirstData, GlobalPayments, Getnet, Amex, entre outras.

Atualmente a empresa possui 403 funcionários, distribuídos em 6 setores: sendo 151 colaboradores do Desenvolvimento, 89 do Suporte, 79 da Produção, 33 do Comercial, 44 do Administrativo e 7 da Diretoria.

Entre muitas aplicações desenvolvidas pela empresa, o que este trabalho aborda é um Sistema de Acesso das Configurações do SiTef - IntSitef, que tem como objetivo viabilizar o acesso de dados de configuração do SiTef.

2.1.1 Métodos de trabalho

A SoftwareExpress utiliza o modelo de desenvolvimento de software iterativo e incremental. O modelo incremental aplica sequências lineares de forma escalonada, sendo que cada uma das sequencias lineares gera um incremento do software. Esses incrementos são entregáveis e prontos para o cliente.

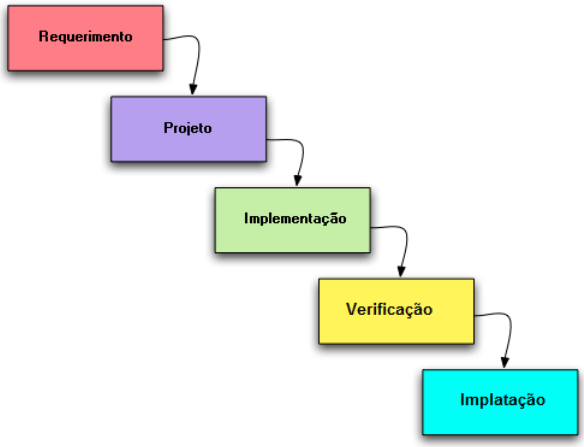
No primeiro incremento de um produto, utilizando o modelo incremental, temos apenas o essencial do produto, contendo os requisitos básicos que devem ser atendidos para o software

entrar em operação. Portanto, no primeiro incremento, muitos recursos complementares ainda não são entregues para os clientes. Após o término do primeiro incremento, o cliente utiliza e avalia esse incremento fornecendo posteriormente um resultado ou *feedback*. Com base nesse resultado fornecido pelo cliente, o próximo incremento é planejado considerando a modificação do primeiro incremento, de acordo com o *feedback* do cliente. Após a liberação de cada incremento é realizado o mesmo processo até que o produto esteja completo.

O modelo incremental de desenvolvimento de software propõe sua construção em pequenas partes operacionais, chamadas incrementos, que são usados para contínuas avaliações dos requisitos e detecções de falhas ou deficiências. No final do ciclo, um novo incremento é gerado, normalmente incorporando o incremento anterior(FUKS, 2003).

Na Software Express, as solicitações de manutenção ou desenvolvimento de algum produto são feitas pelos clientes diretamente para o setor Suporte ou para o setor comercial. Tais solicitações, por sua vez, são repassadas através da ferramenta *TRAC*[1] para os responsáveis técnicos para verificar a viabilidade da sua implementação. Após verificada a viabilidade, é criada a especificação dos requisitos da próxima versão, que é enviada para o cliente para autorização e viabilização. Após essa etapa, o documento de especificação vai para o setor de desenvolvimento para o planejamento da atividade. Após isso, o analista alocado faz uma breve análise da especificação, e aprova o esforço para a atividade, para então, o desenvolvimento da solicitação ser iniciado. Na Figura 2.1 é possível visualizar o modelo incremental utilizado na Software Express.

Figura 2.1 – Modelo Incremental



Fonte: Documentação Sitef

[1] https://trac.edgewall.org/

2.2 Mercado de meios de pagamento

Atualmente o Brasil passa por um avanço consistente do mercado de meios eletrônicos de pagamento, os valores e a quantidade de compras com pagamento com cartões de crédito e débito cresceu muito nos últimos anos. Toda a robustez e a eficiência desse mercado não surgiu por acaso, faz muito tempo que o setor tem se organizado e investido fortemente para garantir que os meios de pagamento funcionem corretamente, ofereçam segurança e tenham ampla aceitação.

Fazer pagamentos usando cartões é uma realidade cada vez mais presente em todo o mundo. A mecânica de inserir o cartão na maquininha, digitar uma senha e alguns segundos depois concluir uma compra é um movimento diário tão comum que se tornou um hábito na vida de milhões de pessoas. No processo de transação, existem três principais responsáveis por garantir que os fluxos de informações e dinheiro envolvidos em uma compra sejam concluídos, e são eles:

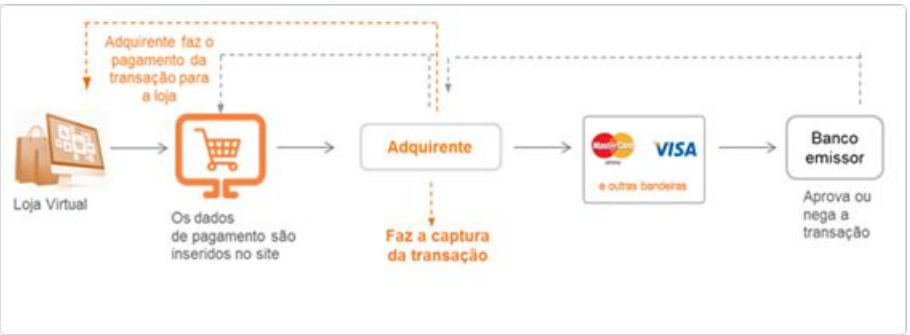
- Adquirentes: Através de cartões de débito e crédito, os adquirentes fazem a liquidação das transações financeiras. Para realizar essa conclusão, eles são responsáveis por comunicar os dados da operação às bandeiras (Mastercard, Visa, American Express, HyperCard, Elo) aos bancos emissores dos respectivos cartões. No Brasil, podemos citar algumas redes adquirentes como Rede, Cielo, GetNet, Bin e Stone. Devido ao seu importante papel nesse processo de compra, os adquirentes exercem grande influência no mercado de meios de pagamento

- Bandeira: São as empresas que conectam os adquirentes aos bancos emissores dos cartões. No ato da compra, elas se conectam ao adquirente usado no estabelecimento comercial, e em seguida acionam a instituição financeira responsável pelo cartão.

- Banco emissor: São os responsáveis por emitir os cartões, além de conceder limites que podem ser gastos aos clientes. O banco faz a autorização de uma compra, reservando o valor na conta do consumidor e capturando a transação. Quando aprovada, também é papel da instituição financeira fazer a liquidação do valor total junto ao adquirente.

A Figura 2.2 é um diagrama do contexto de uma transação eletrônica, onde tem-se o *cliente*, que envia a transação para a *adquirente*, que conecta com a *bandeira* que por sua vez conecta ao *banco* que é responsável por autorizar a transação,

Figura 2.2 – Fluxo de uma transação eletrônica



Fonte: Documentação Sitef

2.3 O Sitef

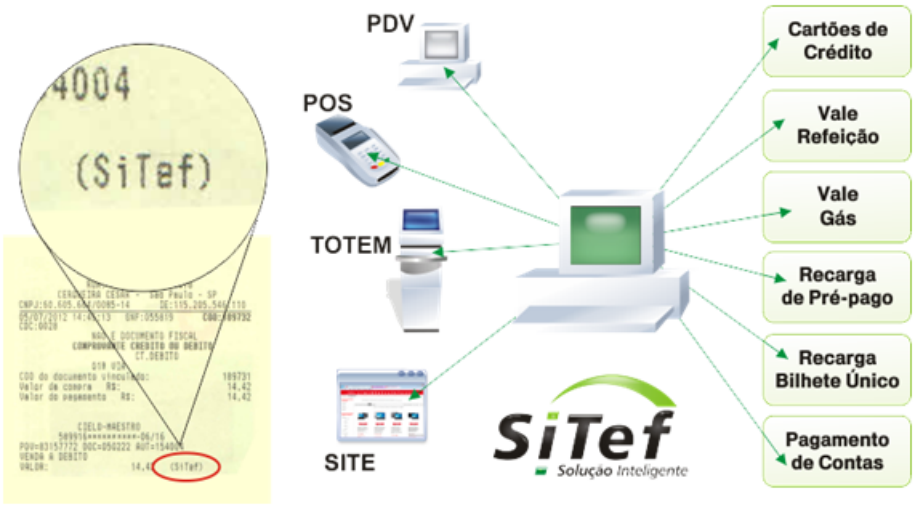
O SiTef surgiu em uma época em que poucas lojas aceitavam cartões e, quando aceitavam, normalmente era apenas de débito. Eram os meados dos anos 90, a economia do país ainda estava bastante instável e, para os estabelecimentos, não era vantajoso aceitar uma compra com cartão de crédito e receber o pagamento vários dias depois, devido justamente à inflação. Quanto a estes poucos estabelecimentos comerciais que aceitavam cartões de crédito, não o faziam de forma automatizada, mas sim em papel, usando a clássica Decalcadora de Cartões de Crédito[2].

Nesta época foi lançada a solução de PDV (“Ponto De Venda”, nada mais é do que um terminal operado, por exemplo, por um caixa de Supermercado) e desejava que esta solução aceitasse pagamento com cartão. Foi nessa época que teve-se o início do desenvolvimento de uma solução de *TEF*("Transação eletrônica de fundos"). Assim, a Software Express desenvolveu sua solução de *TEF* e, em aproximadamente 8 meses, surgiu a Solução Inteligente de Transferência Eletrônica de Fundos, ou o sistema *SiTef*. Com essa solução, durante uma compra, o cliente transfere os seus fundos ao lojista, de forma eletrônica, ou seja, paga uma compra usando um cartão. Portanto, o *SiTef* é um servidor que recebe pedidos de pagamentos com cartões e envia para o autorizador do cartão correspondente.

Atualmente o SiTef se comunica com mais de 300 redes autorizadoras, sendo que estas podem processar cartões de crédito ou débito, assim como cartões presente, cartões de refeição, de alimentação, de combustível, além de muitos outros. Na figura 2.3 pode-se visualizar este fluxo.

[2] Máquina utilizada para a leitura de um cartão de crédito

Figura 2.3 – SiTef



Fonte: Documentação Sitef

2.3.1 Funcionamento do SiTef

O sistema SiTef conecta as duas pontas envolvidas na Transferência Eletrônica de Fundos. Em uma ponta, na loja, estão todos os mais diferentes tipos de automações comerciais (PDV, POS, Totem, um Site na Internet, além de muitas outras formas de se comunicar com o cliente). E na outra ponta, os mais diferentes tipos de Redes Autorizadoras). Atualmente, o SiTef se comunica com mais de 200 Redes Autorizadoras, sendo que estas podem processar cartões de crédito ou débito, assim como cartões presente, cartões de refeição, de alimentação, de combustível, além de muitas outras coisas.

Figura 2.4 – Fluxo do *SiTef*

Fonte: Documentação Sitef

Na figura 2.4 pode-se visualizar o fluxo de uma venda eletrônica e onde o *Sitef* se encaixa, na figura podemos ver a 1ª perna, que seria uma solicitação de venda, e o fluxo passando pelo *SiTef* e indo para uma rede autorizadora, a 2º perna é a resposta dessa solicitação de venda por parte da rede autorizadora, e a 3ª é a confirmação da venda.

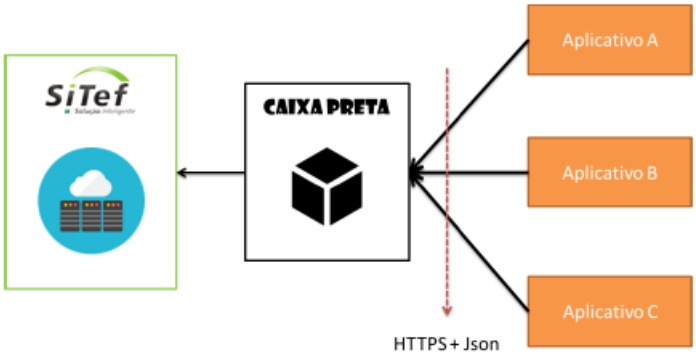
2.3.2 O IntSitef

O IntSiTef é um serviço web que atende às requisições de vários aplicativos que buscam ler e modificar dados de configuração do SiTef. As configurações de módulos no Sitef se tornou algo em que muitas aplicações precisam acessar, e com isso, desenvolveu-se uma solução que conversa com um servidor que, por sua vez, busca essas informações de forma transparente para a aplicação que faz a solicitação.

A fim de centralizar o acesso aos dados da aplicação, definiu-se utilizar em arquitetura de *WebService*.

A Figura 2.5 é um diagrama do contexto da aplicação, onde tem-se o *Sitef*, o *IntSitef*, chamado internamente de caixa preta, e os aplicativos que utilizam o *IntSitef* pra consumir configuração do *Sitef*.

Figura 2.5 – IntSitef



Fonte: Documentação IntSitef

2.3.3 Principais Funcionalidades

Devido à necessidade de acessar as configurações de servidores com o *Sitef*, os seguintes módulos foram desenvolvidos:

- **Módulo de cadastro de um servidor *Sitef***: Módulo para cadastrar as conexões de Sitefs para monitoramento de acessos à configuração;
- **Módulo de monitoração geral dos Sitefs cadastrados**: Módulo para gerenciar a movimentação de transações no IntSitef;
- **Módulo de Status dos Sitefs no ar**: Módulo para gerenciar os Sitefs que estão ou não em operação;

- **Módulo de configuração de certificados JKS**: Módulo para configuração de certificados;
- **Serviço de listagem de módulos**: Serviço que retorna todos os módulos cadastrados para determinado servidor;
- **Serviço de manipulação de instituicao financeira**: Serviço para gravar configuração de instituição financeira no servidor apontado;
- **Serviços para manipulação de empresa**: Serviços de manipulação de uma empresa no servidor *SiTef* correspondente;
- **Serviços para manipulação de um grupo empresa**: Serviços de manipulação de uma empresa no servidor *SiTef* correspondente;

3 REFERENCIAL TEÓRICO

3.1 Modelagem e Estruturas

No desenvolvimento da aplicação, tomou-se um cuidado especial com a modelagem e com as estruturas utilizadas. Esta seção apresenta as principais tecnologias utilizadas para a estruturação do projeto.

3.1.1 Padrão MVC

O *MVC* é utilizado em muitos projetos devido à arquitetura que possui, o que possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o Model, o *Controller* e a *View*, executa o que lhe é definido e nada mais do que isso.

A utilização do padrão *MVC* tráz como benefício o isolamento das regras de negócios da lógica de apresentação, a interface com o usuário. Isto possibilita a existência de várias interfaces com o usuário, que podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, proporcionando, assim, muito mais flexibilidade e oportunidades de reuso das classes.

A comunicação entre interfaces e regras de negócios é definida através de um controlador, que se torna possível a separação entre as camadas. Portanto quando um evento é executado na interface gráfica, a interface irá se comunicar com o controlador que, por sua vez, irá se comunicar com as regras de negócios.

Explicando cada um dos objetos do padrão *MVC* tem-se, primeiramente, o *Controller*, que interpreta as entradas enviadas pelo usuário e mapeia essas ações do usuário em comandos que são enviados para o *Model* e para a *View* para efetuar a alteração apropriada. Sendo assim, o *Model* gerencia um ou mais elementos de dados, responde a perguntas sobre o seu estado e responde a instruções para mudar de estado. O *Model* sabe o que o aplicativo quer fazer e é a principal estrutura computacional da arquitetura, pois é ele quem modela o problema que está se tentando resolver (DOOLEY,2011). Por fim, a *View* é responsável por apresentar as informações para o usuário. Ela não sabe nada sobre o que a aplicação está atualmente fazendo, tudo que ela realmente faz é receber instruções do *Controller* e informações do *Model* e, então, exibi-las. A *View* também se comunica de volta com o *Model* e com o *Controller* para reportar o seu estado.

Portanto, a principal ideia do padrão arquitetural *MVC* é a separação entre conceitos e código. O *MVC* é como a clássica programação orientada a objetos, criando objetos que escondem as suas informações e como elas são manipuladas e, então, apresentar apenas uma simples interface para quem quer usar. Dentre as diversas vantagens do padrão MVC estão a possibilidade de reescrita do *Controller* sem alterar o modelo, reutilização para diferentes aplicações com pouco esforço, facilidade na manutenção e adição de recursos, reaproveitamento de código, facilidade de manter o código sempre limpo.

A Listagem 3.1 apresenta um exemplo de organização da aplicação *IntSitef* utilizando o padrão MVC.

Listagem 3.1 – Exemplo da organização da aplicação utilizando o padrão MVC

```
routers
dao
models
services
forms
views
controllers
actions
filters
stuff
```

3.1.2 WebService

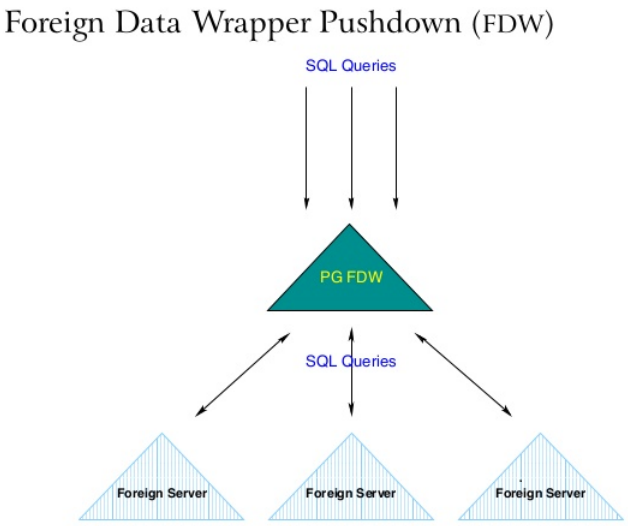
WebService é uma solução utilizada na integração de sistemas e na comunicação entre diferentes aplicações. Utilizando desta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Assim, todas aplicações continuam com suas particularidades, mas todas fazem interface com o web Service, portanto, toda comunicação entre sistemas passa a ser dinâmica e principalmente segura (COULOURIS, 2003). O intuito do WebService é fazer com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos pelo *WebService*. Esta comunicação é realizada com intuito de facilitar a integração das aplicações de uma empresa, ou seja, interoperabilidade entre a informação que circula numa organização nas diferentes aplicações. Por este motivo para o *IntSitef* foi utilizado

este conceito de estar disponível para qualquer aplicação, facilitando a integração com as outras aplicações.

3.1.3 Integração de dados com Foreign Data Wrappers

O Foreign Data Wrappers (FDW) é uma regulamentação de um padrão de configuração, desenvolvimento e aplicação de acessos a dados externos em *SGBDs* e arquivos de outros fornecedores. O conceito do *FDW* não é apenas tornar dados visíveis de várias origens em um único ponto, mas também integrar o uso desses dados, por exemplo, utilizando *joins* ou filtros de tabela externas como parâmetro para retornar dados de outra tabela de outra fonte (FOREIGN DATA WRAPPERS, 2019). Dessa forma, é possível que sejam feitas operações no acesso a dados externos, como *select*, *insert*, *update* e *delete*. Essa facilidade de manipulação de dados é um grande diferencial do *FDW*, se comparado com ferramentas que têm o mesmo propósito de integração de dados, geralmente, sem real-time ou permissão de alteração de objetos .

Figura 3.1 – FDW



Fonte: Documentação Sitef

Na Figura 3.1 é possível visualizar o fluxo do *FDW*, e como os objetos externos podem se conectar ao *PostgreSQL*, para que seja possível trabalhar com as informações dessas fontes externas. O *PostgreSQL* disponibiliza o acesso aos dados utilizando-se do conceito de tabelas externas, o que torna transparente o acesso aos dados externos, e são utilizados os mesmos procedimentos de acesso a tabelas locais.

Como o intuito da aplicação *IntSitef* é consumir configuração da aplicação *Sitef*, utilizou-se desta tecnologia para trabalharmos com dados externos provindos de arquivos (.txt, .ini, entre

outros), para serem formalizados em linguagem *SQL*, para uma melhor formalização e preparando para uma possível mudança do sistema *SiTef* para um banco de dados.

3.2 Tecnologias *Back-end*

São chamadas de *back-end* as tecnologias executadas do lado do servidor de aplicações. Esta seção apresenta as principais tecnologias para desenvolvimento back-end utilizadas para o desenvolvimento da aplicação.

3.2.1 Play Framework

Play Framework é uma solução que almeja entregar ao desenvolvedor de software um ambiente mais produtivo para a elaboração e manutenção de sistemas que necessitam dos requisitos exigidos pelas aplicações web atuais (PLAY FRAMEWORK, 2019). É uma ferramenta que disponibiliza formas eficazes para atender sistemas altamente distribuídos, com alta disponibilidade, que lidam com alta concorrência, com programação assíncrona,e também com os novos paradigmas de armazenamento de informação, tais quais os bancos de dados não relacionais. Além da capacidade de tornar um desenvolvedor mais eficiente durante o processo de desenvolvimento do software.

Com uma rápida curva de aprendizado, um sistema de detecção de erros robusto e uma capacidade de atualização de código em tempo real tornam o *Play* uma ferramenta de alta produtividade. A organização do projeto em uma estrutura web *MVC* e o seu modelo *RESTful* o tornam de fácil entendimento (PLAY FRAMEWORK, 2019).

O *Play* torna o desenvolvedor eficiente não só pela detecção de erros otimizada, mas também pela forma com que lida com o processamento de código-fonte simultâneo à aplicação rodando, o chamado *hot-reloading*, que permite que o desenvolvedor altere o código sem precisar reiniciar a aplicação, o que torna muito mais curto o ciclo de *feedback*.

Como a aplicação *IntSitef* necessitava de uma ferramenta que fornecesse facilidade para trabalhar com alta concorrência, utilizar o *RESTful*, possuir capacidade de grande organização e trabalhasse bem com sistemas distribuídos, foi escolhido o *Play Framework* como ferramenta principal de todo o projeto.

3.2.2 Scala

Scala é uma linguagem de programação de propósito geral, multiparadigma, projetada para expressar padrões de programação comuns de uma forma concisa, elegante e *type-safe*. Ela incorpora recursos de linguagens orientadas a objetos e funcionais (SCALA, 2019). Também é plenamente interoperável com Java. Scala é uma linguagem de programação relativamente nova, mesmo assim, nos últimos anos, conquistou empresas gigantes como o Twitter[1] e o Foursquare[2]. Uma das primeiras diferenças entre Scala e uma linguagem como Java, é que Scala também suporta o paradigma funcional. A Listagem 3.2 apresenta um exemplo de validação de CPF utilizado na aplicação *IntSitef*.

Listagem 3.2 – Exemplo de validação de CPF com Scala

```
object SE_Validations {

  def calculate(str : String) : Integer = {
     val peso : Array[Int] = Array(6,5, 4, 3, 2, 9, 8, 7, 6, 5, 4,
        3, 2)
     var soma : Integer = 0
     var digito : Integer = 0
     var indice : Integer = str.length - 1

     while (indice >= 0) {
        digito = Integer.parseInt(str.substring(indice,indice+1))
        soma = soma +  digito * peso(peso.length - str.length +
           indice)
        indice = indice - 1
     }
     soma = 11 - (soma % 11)
     if (soma > 9) 0 else soma
  }

  def isValidCNPJ(cnpj : String) : Boolean = {
```

[1] Twitter é uma rede social, que permite aos usuários enviar e receber atualizações pessoais de outros contatos, por meio do website.

[2] Foursquare é uma rede geossocial que permite ao utilizador indicar onde se encontra, e procurar por contatos seus que estejam próximo desse local.

```
        if (cnpj.length() != 14)
            false

        val digito1 : Integer  = calculate(cnpj.substring(0,12))
        val digito2 : Integer  = calculate(cnpj.substring(0,12) +
            digito1)
        cnpj.equals(cnpj.substring(0,12) + digito1.toString + digito2.
            toString) match{
            case (true) => true
            case (false) => false
        }
    }
}
```

Scala é também uma linguagem funcional,permitindo que funções sejam aninhadas. As classes Scala e seu suporte interno para *pattern matching* modela tipos algébricos usados em muitas linguagens de programação funcionais. Estas características, fazem com que o Scala seja ideal para o desenvolvimento de aplicações como *web services*. A Listagem 3.3 apresenta a implementação de um loop.

Listagem 3.3 – Exemplo de loop

```
 def whileLoop(condition: => Boolean)(command: => Unit) {
  if (condition) {
    command; whileLoop(condition)(command)
  } else ()
}
```

Na aplicação *IntSitef* foi escolhido trabalhar com a linguagem *Scala*, por ser suportada no *Play Framework* e ser de fácil manuseio, podendo, assim, criar uma solução escalável.

3.3 Tecnologias *fron-end*

São chamadas de *front-end* as tecnologias executadas do lado do cliente das aplicações. Esta seção apresenta as principais tecnologias para desenvolvimento *front-end* utilizadas na aplicação.

3.3.1 HTML

A HTML é uma linguagem de marcação utilizada para estruturar todo o conteúdo das páginas web, definindo elementos como parágrafos, títulos, imagens, etc. Sendo assim, essa linguagem é utilizada na construção de qualquer página web e, portanto, deve ser conhecida por todo programador que trabalha com esse tipo de aplicação. Todo documento HTML possui *tags*, que são os comandos de formatação da linguagem. Um elemento é formado por um nome de marcador, atributos, valores e filhos. Esses atributos modificam os resultados padrões dos elementos e os valores caracterizam essa mudança (SAMY, 2015). Na Listagem 2.4 é apresentado um exemplo de código HTML. As *tags* utilizadas no exemplo são descritas a seguir:

- <!DOCTYPE html>: permite a declaração do tipo do documento.
- <html>: define o início de um documento HTML e indica ao navegador que todo conteúdo posterior deve ser tratado como uma série de códigos HTML.
- <body>: define o conteúdo principal, o corpo do documento. Esta é a parte do documento HTML que é exibida no navegador. No corpo podem-se definir atributos comuns a toda a página, como cor de fundo, margens, e outras formatações.
- <title>: define o título da página, que é exibido na barra de título dos navegadores
- <h1>, <h2>, ... <h6>: Títulos que variam de tamanho dentro das prioridades (sua aparência pode ser alterada com CSS - *Cascade Style Sheet* - Folhas de Estilo em Cascata).
- <p>: Parágrafo.
-
: quebra de linha.
- <b>, <i>, <u> e <s>: negrito, itálico, sublinhado e riscado, respectivamente.

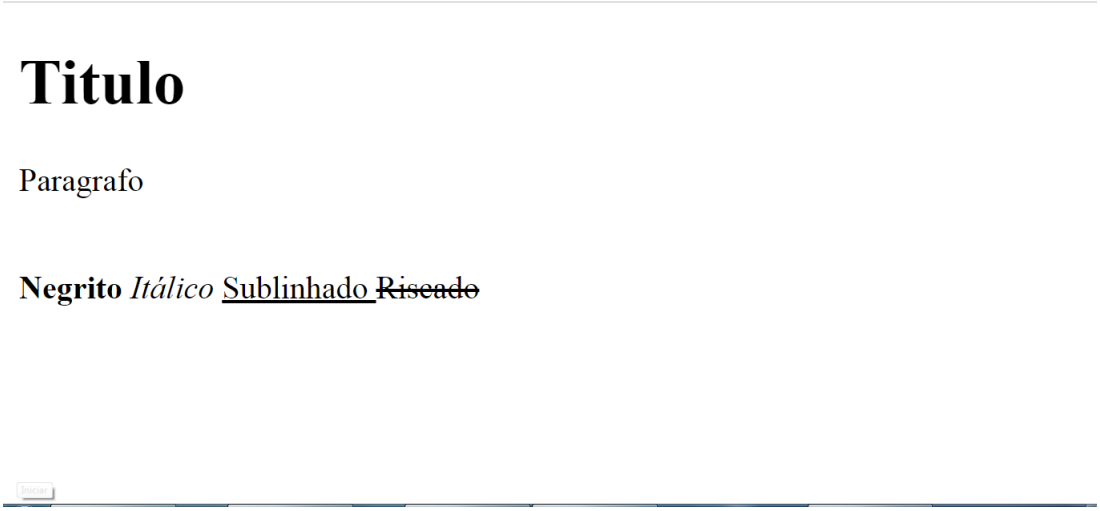
Listagem 3.4 – Exemplo de código HTML

```
<!DOCTYPE html>
<html>
        <title> Titulo da pagina </title>
        <body>
                <h1> Titulo </h1>
                <p>Paragrafo </p>
```

```
                </br>
                <b> Negrito </b>
                <i> It lico </i>
                <u> Sublinhado </u>
                <s> Riscado </s>
        </body>
</html>
```

O código do exemplo da Listagem 3.4 irá gerar contéudo que pode ser visualizado no navegador conforme apresentado na Figura 3.2.

Figura 3.2 – Exemplo HTML



Na aplicação *IntSitef*, foi utilizado o HTML para estruturar as interfaces *front-end* de todo o projeto.

3.3.2 CSS

CSS é chamado de linguagem *Cascading Style Sheet* e é usado para estilizar elementos escritos em uma linguagem de marcação como HTML. O CSS consegue separar o conteúdo da representação visual do site, portanto, utilizando o CSS é possível alterar o modo de visualização de qualquer elemento HTML.

CSS foi desenvolvido pelo *W3C (World Wide Web Consortium)* pois o HTML não foi projetado para ter tags que ajudariam a formatar a página (SAMY, 2015).

A diferença entre um site que implementa CSS e outro que não o usa é gigantesca e notável, com o seu uso as possibilidades de personalização são quase infinitas. Hoje em dia, isso está se tornando mais uma necessidade do que um simples recurso.

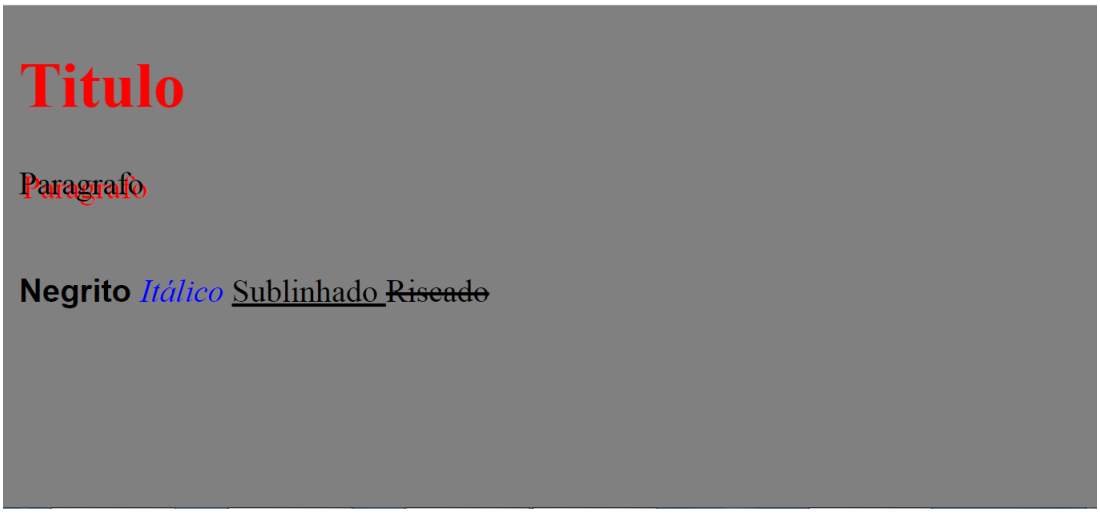
A Listagem 3.5 apresenta um exemplo de código CSS e, na Figura 3.3 é mostrado como a Figura 3.2 irá ficar com a estilização do CSS da listagem 3.5.

Listagem 3.5 – Exemplo de código CSS

```
<style>
body{
        background-color: #808080;
}
h1{
        color:red;
}
p{
        text-shadow: 2px 2px #ff0000;
}
b{
        font-family: Arial, Helvetica, sans-serif;
}
i{
        color:blue;
}
</style>
```

Na aplicação *IntSitef*, foi utilizado o *CSS* em toda a estrura de *front-end* para a criação dos estilos das páginas web.

Figura 3.3 – Exemplo HTML utilizando CSS



3.3.3 JavaScript

JavaScript é uma linguagem de programação interpretada de alto nível, caracterizada também, como dinâmica, fracamente tipada, *prototype-based* e multi-paradigma. Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da *World Wide Web*. JavaScript permite páginas da Web interativas e, portanto, é uma parte essencial dos aplicativos da web. A grande maioria dos sites usa esta linguagem, e todos os principais navegadores têm um mecanismo JavaScript dedicado para executá-lo.

Como uma linguagem multi-paradigma, o JavaScript suporta estilos de programação orientados a eventos, funcionais e imperativos. Possuí *APIs* para trabalhar com texto, matrizes, datas, expressões regulares e o *DOM*[3](GOMES, 2006).

Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido. Porém os mecanismos JavaScript agora estão incorporados em muitos outros tipos de software host, incluindo servidores e bancos de dados da Web e em programas que não são da Web, como processadores de texto e PDF, e em tempo de execução ambientes que disponibilizam JavaScript para escrever aplicativos móveis e de desktop, incluindo widgets de área de trabalho.

Na Listagem 3.6 é apresentado um exemplo de JavaScript que soma dois números e insere o HTML em uma tag com o *id* especificado.

[3] O Document Object Model ou simplesmente DOM é utilizado pelo navegador Web para representar a sua página Web

Listagem 3.6 – Exemplo de código JavaScript

```
<script>
var p1 = 5; // declarando vari vel 1
var p2 = 6; // declarando vari vel 2
var total = p1 + p2;
document.getElementById("tag_html").innerHTML =
"The total is: " + total;
</script>
```

Na aplicação *IntSitef*, o JavaScript foi o responsável por manipular dados no *front-end* e por integrar a aplicação, entre *front-end* e *back-end*.

3.4 Ferramentas Auxiliares e *Builders*

Neste capítulo são apresentadas as ferramentas auxiliares e os *builders* utilizados para o desenvolvimento da aplicação *IntSitef*.

3.4.1 SBT

SBT é um gerenciador de pacotes que tem como objetivo juntar todas as dependências que necessita para o projeto em um só lugar, assim, facilitando na manutenção de novas versões. Utilizando o SBT não é necessário passar para outro ambiente de desenvolvimento todas as dependências, pois já estão organizadas neste arquivo, bastando instalá-las para que o sistema funcione em nível de desenvolvimento (SBT, 2019). No projeto *IntSitef*, o SBT foi utilizado para levar todas as dependências do projeto para qualquer outro ambiente de desenvolvimento, portanto, foi possível concentrar todas as dependências do projeto e todas as configurações necessárias. Na Listagem 3.7 é mostrado um exemplo do arquivo de *build* do *SBT*.

Listagem 3.7 – Exemplo de build

```
// Sample build.sbt.
// Nota  es:
//   Linhas em branco precisam separar as instru  es.
```

```
//   := significa que voc  est  definindo o valor para essa chave
//   += significa que voc  est  adicionando aos valores dessa chave

name := "MyProject"

version := "0.1"

organization := "MyMegaCorp"

scalaVersion := "2.9.2"

sbtVersion := "0.13"

// As depend ncias est o no formato Maven, com % separando as
   partes.
// Observe o bit extra "test" no final do JUnit e ScalaTest, que
// significa que    apenas uma depend ncia de teste.
//
// O %% significa que ele adicionar  automaticamente a vers o
   espec fica do Scala ao nome da depend ncia.
// Por exemplo, isso ir  realmente baixar scalatest_2.9.2

libraryDependencies += "org.scala-lang" % "scala-swing" % "2.9.2"

libraryDependencies += "org.scalatest" %% "scalatest" % "1.9.1" % "
   test"

libraryDependencies += "com.github.benhutchison" % "scalaswingcontrib
   " % "1.3"

libraryDependencies += "junit" % "junit" % "4.8.1" % "test"

// Comandos iniciais a serem executados no seu REPL. Poss vel
   importar v rias ferramentas espec ficas do projeto aqui.
```

```
initialCommands := """
    import myproject.stuff._;
    import myproject.other.stuff._;
  """
```

Na aplicação *IntSitef*, foi utilizado o *SBT* para carregar todas as dependências do projeto, juntamente com todas as configurações necessárias para *build* do projeto.

3.4.2 GrayLog

O Graylog é uma ferramenta que permite o fácil gerenciamento de logs de dados estruturados e não estruturados.

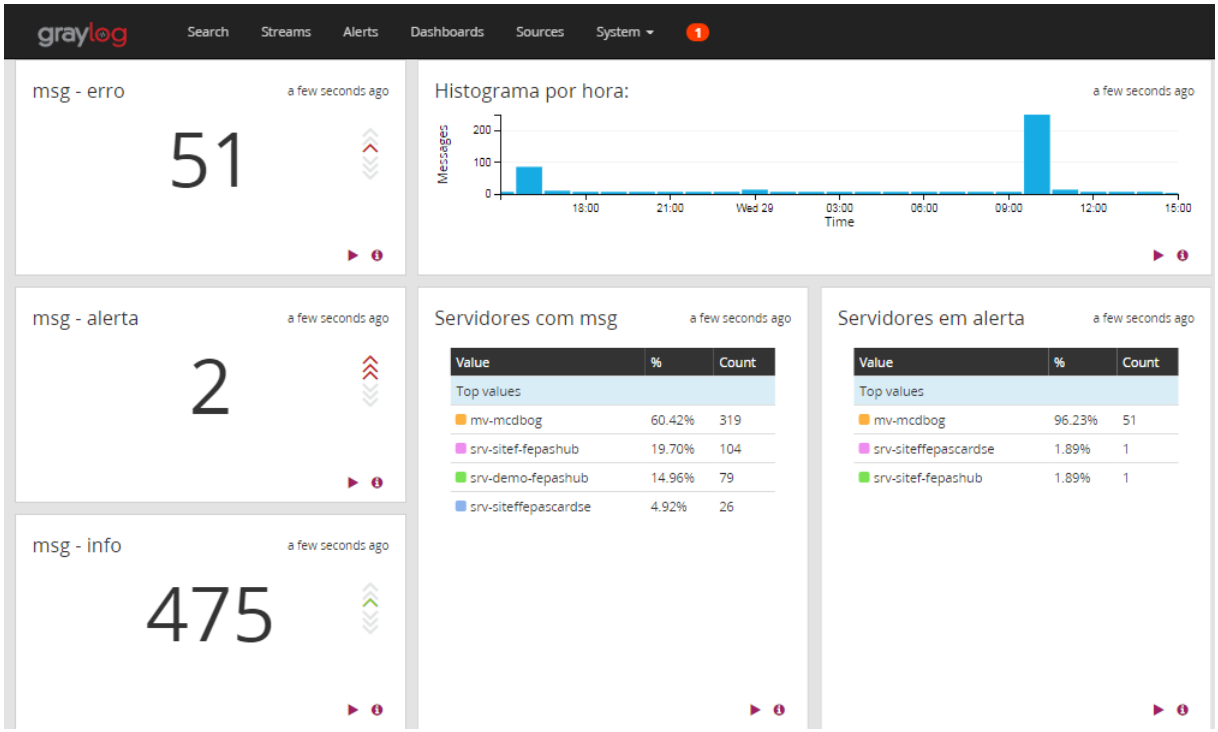
Além disso, o Graylog possui um servidor principal que recebe dados de seus clientes instalados em diferentes servidores e, contém um dashboard para visualizar os dados permitindo trabalhar com logs agregados através de um único servidor.

Outra característica do Graylog é que ele permite pesquisa personalizada nos logs usando consultas estruturadas. Quando integrado corretamente com um aplicativo da Web, o Graylog pode ajudar as equipes de segurança, desenvolvimento e infraestrutura a analisar o comportamento do seu sistema (GRAYLOG, 2019).

O Graylog foi desenvolvido, especificamente, para oferecer a melhor coleta, armazenamento, enriquecimento e análise de registros. A simplicidade na pesquisa, exploração e visualização torna o Graylog uma solução completa.

Na aplicação *IntSitef*, o Graylog apoiou a equipe de infraestrutura e segurança, pois fornece um design que permite uma análise de dados mais rápida. Com a implantação do graylog foi possível analisar melhor o tráfego de dados na aplicação. Pode-se analisar pela Figura 3.4 o *dashboard* gerado pelo *GrayLog* no ambiente do *IntSitef*, onde tem-se a representação das mensagens de alerta, informação e de erro.

Figura 3.4 – DashBoard do GrayLog



Fonte: GrayLog Server

3.4.3 PostMan

Postman é uma ferramenta que tem como objetivo testar serviços RESTful (Web APIs) por meio do envio de requisições HTTP e da análise do seu retorno. Com ele é possível consumir facilmente serviços locais e na internet, enviando dados e efetuando testes sobre as respostas das requisições. O uso de Web APIs vem crescendo e o Postman auxilia nos testes desse tipo de projeto, bem como permite a desenvolvedores analisarem o funcionamento de serviços externos, a fim de saber como consumi-los (POSTMAN, 2019). Essa ferramenta foi de grande importância durante o desenvolvimento e durante a manutenção do sistema, pois com ele foi possível disparar qualquer requisição de forma simplificada, auxiliando no tempo de desenvolvimento.

3.4.4 Docker

O Docker possibilita o empacotamento de uma aplicação ou ambiente inteiro dentro de um container e, a partir desse momento, o ambiente inteiro torna-se portável para qualquer outro Host que contenha o Docker instalado.

Isso reduz drasticamente o tempo de *deploy* de alguma infraestrutura ou, até mesmo, da aplicação, pois não há necessidade de ajustes de ambiente para o correto funcionamento do serviço o ambiente é sempre o mesmo, configure-o uma vez e replique-o quantas vezes quiser (DOCKER, 2019).

Para a aplicação *IntSitef*, o docker foi de muita importância apoiando o ajuste e empacotamento da aplicação, evitando o problema de implantação por parte da produção.

3.4.5 Mercurial

Com a necessidade de controlar as versões que foram lançadas para o cliente e as que estão em desenvolvimento, foi utilizado *Mercurial*, que possibilita o desenvolvimento paralelo (MERCURIAL, 2019). Na *Software Express*, o *Mercurial* foi utilizado como ferramenta de controle de versão, auxiliando na criação de versões de produção, homologação, ramos de desenvolvimento e outros.

3.4.6 Visual Studio Code

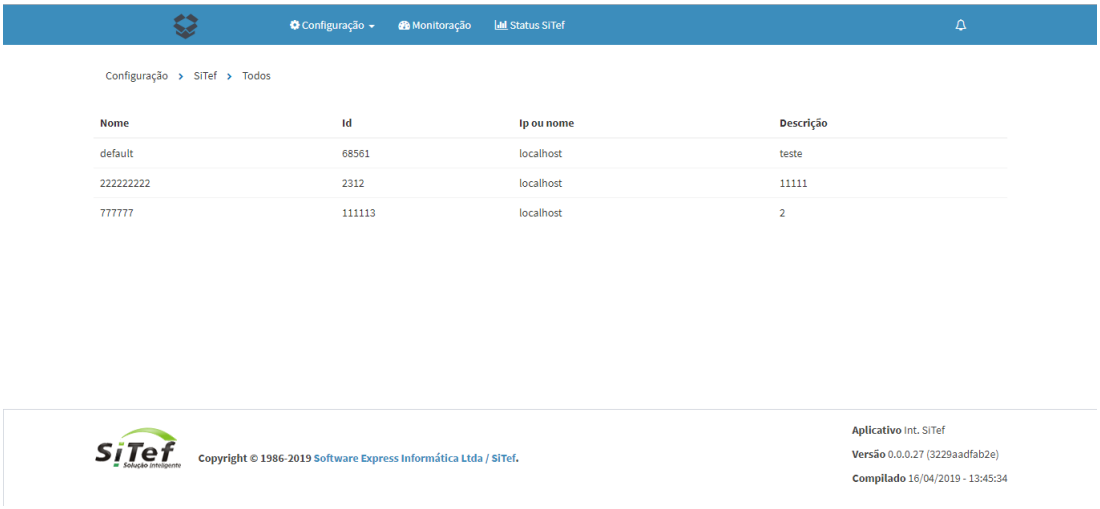
Visual Studio Code é uma versão gratuita de *Visual Studio*, que é um aplicativo para formatação de códigos. Nele existem extensões que facilitam o desenvolvimento, pré compilando o código e mostrando as falhas antes da execução deste mesmo (VSC, 2019). O Visual Studio Code foi utilizado no desenvolvimento da aplicação, pela facilidade de desenvolvimento com maior agilidade.

4 ATIVIDADES DESENVOLVIDAS

Este capítulo descreve o desenvolvimento das principais funcionalidades executadas pelo estagiário no sistema IntSitef durante o estágio, no período de maio de 2018 à Outubro de 2018.

A Figura 4.1 representa a página inicial do sistema *IntSitef* mostrando uma tabela com o nome, id, IP[1] e descrição de alguns servidores de testes do *SiTef* configurados no *IntSitef*.

Figura 4.1 – Página Home



Fonte: IntSitef

4.1 Cadastro de um servidor SiTef

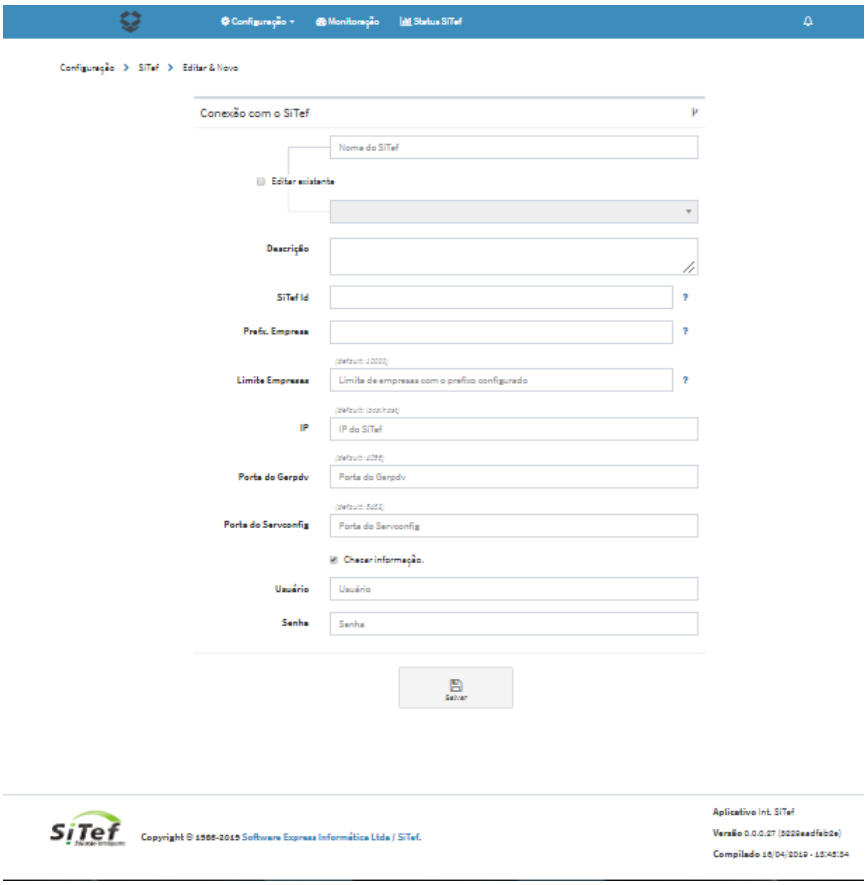
Neste módulo, o estagiário trabalhou no desenvolvimento do cadastro de um servidor SiTef. O cadastro de um servidor *SiTef*, dentro da aplicação, é importante, pois é a partir de um servidor *SiTef* que surgiu a necessidade do desenvolvimento do *IntSitef* para configurar e consumir da configuração de um servidor. A principal atividade desenvolvida pelo estagiário nesse módulo foi o desenvolvimento de uma interface genérica que forneceu a possibilidade de trabalhar as configurações tanto localmente, quanto em um banco de dados .

Neste módulo, foi realizado, também, o serviço de teste da conexão quando inserido um servidor novo e um serviço para buscar os servidores já existentes, para não se repetirem, tendo como campo de busca o nome do servidor e suas devidas validações, garantindo a consistência dos dados cadastrados no sistema.

[1] IP significa "Internet Protocol"e é um número que identifica um dispositivo em uma rede

Na figura 4.2 é possível visualizar a interface utilizada para o cadastro do servidor *SiTef* na aplicação *IntSitef*.

Figura 4.2 – Cadastro SiTef



Fonte: IntSitef

Para o cadastro de um novo servidor *Sitef* é necessário preencher os seguintes campos:

- Nome do *SiTef*, nome de identificação de um servidor *SiTef*.
- Descrição, descrição de um servidor *SiTef*.
- *SiTef* Id, identificador de um servidor *SiTef*.
- Prefixo Empresa, um prefixo que será colocado no nome da empresa quando a mesma é criada, para um servidor *SiTef*.
- Limite Empresas, limite do número de empresas para um servidor *SiTef*.
- IP, *ip* para um servidor *SiTef*.

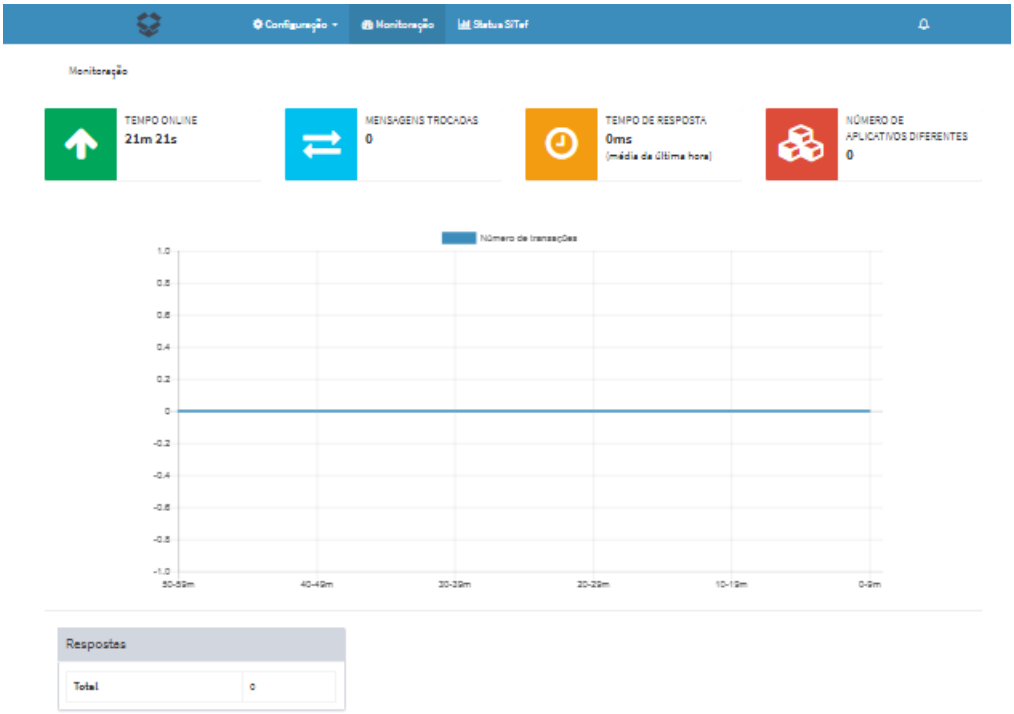
- Porta do *Gerpdv*, porta para o *gerpdv*, que é um aplicativo interno de um servidor *SiTef*.
- Porta do *Servconfig*, porta para o *servconfig*, que é um aplicativo interno de um servidor *SiTef*.
- Usuário, usuário para o acesso de um servidor *SiTef*.
- Senha, senha para o acesso de um servidor *SiTef*.

4.2 Monitoração

O módulo de monitoração, oferece estatísticas de como o servidor está trabalhando, possibilitando ao administrador um controle maior das atividades que o servidor está executando. Nestas estatísticas estão, a quantidade de mensagens trocadas entre o servidor *IntSitef* e o aplicativos que buscam as configurações, a quantidade em que cada recurso é utilizado e qual aplicativo que foi o responsável por tais requisições. Neste módulo, o estagiário foi responsável pelo desenvolvimento de atividades referentes ao back-end, utilizando as ferramentas *postman* para realizar as requisições e linguagens *Scala* e banco de dados *postgreSQL*.

Na Figura 4.3 são visualizadas as informações estatísticas através da interface criada na aplicação *IntSitef*.

Figura 4.3 – Monitoração



Fonte: IntSitef

Na figura 4.3 é possível visualizar o tempo online em que a aplicação *IntSitef* está *online*, as mensagens trocadas e o tempo de espera entre o *IntSitef* e os aplicativos que buscam as configurações do *SiTef* e, o número de aplicativos que estão se conectando na aplicação *IntSitef*.

4.3 Inclusão de Instituição Financeira

Neste módulo, o estagiário foi responsável, também, pelo desenvolvimento de atividades referentes ao *back-end*.

A primeira atividade realizada nesse módulo foi a inserção de uma *rota* que faz a comunicação cliente/servidor e servidor/servidor para obtenção de informações estabelecendo rotas intuitivas, facilitando o consumo de informações necessárias do *IntSitef*. Com isso, foram criadas validações para garantir consistência nos dados vindos de um *JSON*, possibilitando criar os tratamentos necessários no *Controller* e, consequentemente, na *DAO* da aplicação.

Com essa inclusão, foi possível inserir um componente "Instituição Financeira"dentro da configuração de uma empresa. Na Tabela 4.1 pode-se analisar a estrutura do *JSON* para a

inclusão de uma "Instituição Financeira". A estrutura do *JSON* possui um método *POST* para inclusão, com os campos de código e descrição de tipos *numérico* e *string* respectivamente.

Tabela 4.1 – JSON para inclusão de Instituição Financeira

Método POST	Campo	Tipo	Descri ção
	codigo	Num	Código da Instituição
	descricao	Str	Descrição

4.4 CRUD de Empresa

Nesta atividade, o estagiário desenvolveu atividades referentes ao *back-end* da aplicação.

A primeira atividade desenvolvida foi a inserção de várias *rotas* para que as informações necessárias pudessem ser enviadas ao *IntSitef*. Com isso foram criadas validações para garantir consistência nos dados vindos de um *JSON*, possibilitando criar os tratamentos necessários no *Controller* e, consequentemente, na *DAO*. O tratamento de empresa é mais complexo e utilizado em praticamente todos os fluxos da aplicação. Nesse caso, tem-se a ideia de um *CRUD* que são as quatro operações básicas (criação, consulta, atualização e destruição de dados).

Com essa inclusão, foi possível inserir, ler, excluir e atualizar um componente "empresa"dentro da configuração de um Sitef e, possibilitando, assim, tratar inúmeras outras finalidades.

A Tabela 4.2 mostra a estrutura do *JSON* para a inclusão de uma empresa, possuindo um método *PUT* para inclusão, com os campos de estab (código de estabelecimento), dado complementar (descrição), cnpj, prefixo empresa, rede, instituição financeira e com os tipos *string*, *string*, *string*, *string*, *instituição financeira* e *int*, respectivamente.

A Tabela 4.3 podemos analisar a estrutura do *JSON* para a Atualização de uma empresa, possuindo um método *POST* para atualização, com os campos de estab (código de estabelecimento), dado complementar (descrição), cnpj, prefixo empresa, rede, instituição financeira. As colunas Tipo e Descrição mostram o tipo e a descrição de cada campo, respectivamente..

A Tabela 4.4 mostra a estrutura do *JSON* para a captura de uma empresa, possuindo um método *GET* para o retorno de informações, passando o parâmetro *"empresa"* na *URL*.

A tabela 4.5 mostra a estrutura do *JSON* para a excluir uma empresa, possuindo um método *DELETE* para informar a ação de exclusão, passando o parâmetro *"empresa"* na *URL*.

Tabela 4.2 – JSON para inserir de uma empresa

Método PUT	Campo	Tipo	Descri ção
	estab	Str	Código estabelecimento
	dadocomplementar	Str	Descrição
	cnpj	Str	CNPJ configurado para a empresa
	prefixoempresa	Str	Prefixo da empresa
	rede	int	Código da rede
	instituicaofinanceira	IF	Objeto IF

Tabela 4.3 – JSON para atualizar uma empresa

Método POST	Campo	Tipo	Descri ção
	estab	Str	Código estabelecimento
	dadocomplementar	Str	Descrição
	cnpj	Str	CNPJ configurado para a empresa
	prefixoempresa	Str	Prefixo da empresa
	rede	int	Código da rede
	instituicaofinanceira	IF	Objeto IF

Tabela 4.4 – JSON para capturar uma empresa

Método	GET
URL	/pdap/empresa/empresa/<empresa>

Tabela 4.5 – JSON para excluir uma empresa

Método	DELETE
URL	/pdap/empresa/empresa/<empresa>

4.5 CRUD de Grupo de empresas

Nesta atividade, o estagiário desenvolveu atividades referentes ao *back-end* do sistema.

Nesta atividade foi desenvolvida a inserção de várias *rotas* para que as informações necessárias pudessem ser enviadas ao *IntSitef*. Com isso, foram criadas validações para garantir consistência nos dados vindos do *JSON*, possibilitando criar os tratamentos necessários no *Controller* e, consequentemente, na *DAO*. Nesse caso também foi desenvolvido um *CRUD* para um grupo de empresas.

Com essa inclusão, foi possível inserir, consultar, atualizar e excluir um componente "empresa"dentro de um grupo de empresas e criar em si o "grupo de empresas", podendo, dessa forma, tratar inúmeras outras finalidades.

A Tabela 4.6 mostra a estrutura do *JSON* para a inclusão de um grupo de empresas, possuindo um método *PUT* para inclusão, com os campos de cnpj do tipo *string*.

A Tabela 4.7 mostra a estrutura do *JSON* para a atualização de um grupo de empresas, possuindo um método *POST* para atualização, com os campos de cnpj e com o tipo *string*.

A Tabela 4.8 mostra a estrutura do *JSON* para a captura de um grupo de empresas, possuindo um método *GET* para o retorno de informações, passando o parâmetro *grupo* na *URL*.

A Tabela 4.9 mostra a estrutura do *JSON* para a exclusão de um grupo de empresas, possuindo um método *DELETE* para a exclusão de informações, passando o parâmetro *grupo* na *URL*.

Tabela 4.6 – JSON para inclusão de um grupo

Método PUT	Campo	Tipo	Descri ção
	cnpj	Str	CNPJ do grupo

Tabela 4.7 – JSON para atualização de um grupo

Método POST	Campo	Tipo	Descri ção
	cnpj	Str	CNPJ do grupo

Tabela 4.8 – JSON para consulta de um grupo

Método	GET
URL	/pdap/grupo/grupo/<grupo>

Tabela 4.9 – JSON para excluisão de um grupo

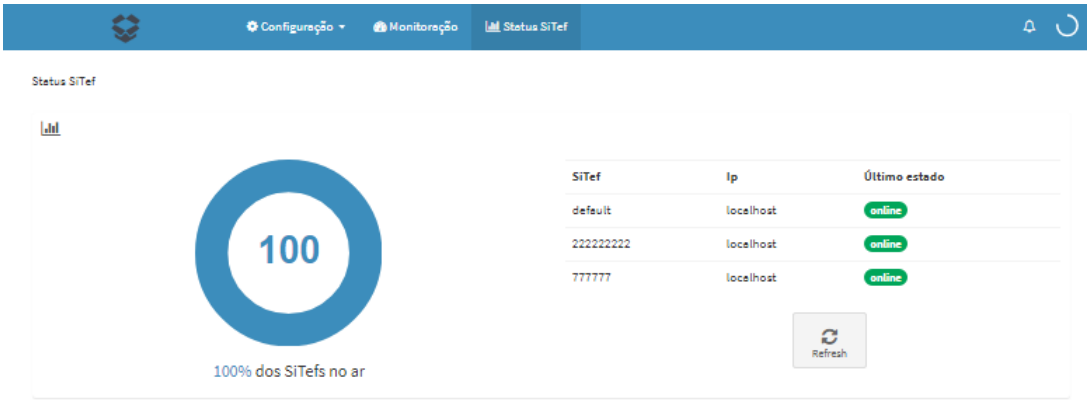
Método	DELETE
URL	/pdap/grupo/grupo/<grupo>

4.6 Visualizar Status

Neste módulo, o estagiário atuou no desenvolvimento do visualizador de status dos servidores SiTef. Este módulo possibilita controlar quais os servidores Sitef estão funcionando. A tela principal do módulo mostra um gráfico geral de disponibilidade dos servidores, que possiblita ao administrador do servidor saber sobre o status do sistema, dando uma maior controle para o administrador do servidor.

Na Figura 4.4 é mostrado o módulo de Status no ambiente da aplicação *IntSitef*.

Figura 4.4 – Status



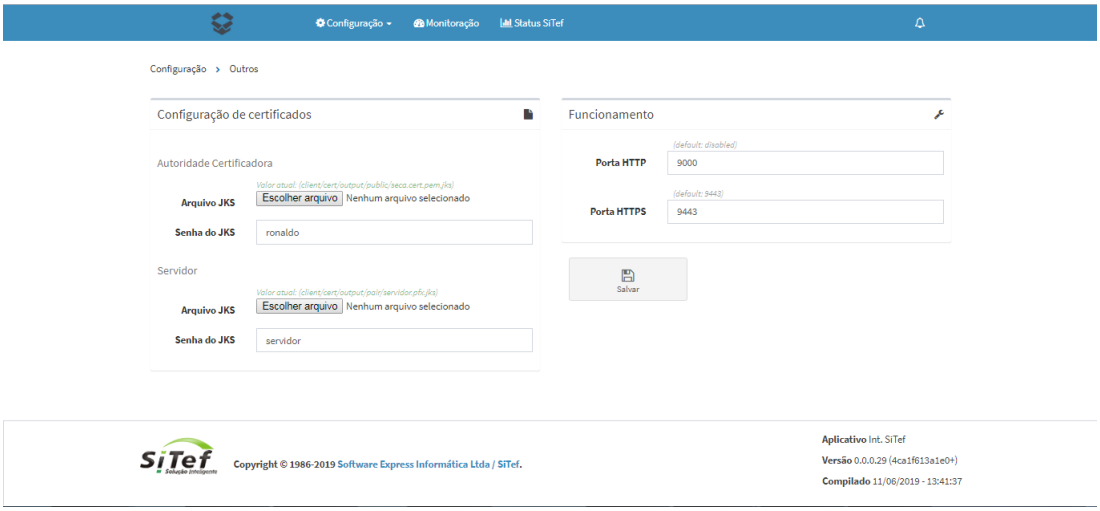
Fonte: IntSitef

4.7 Configuração de Certificados

Neste módulo, o estagiário atuou no desenvolvimento do configurador de certificados. Neste módulo é possível configurar os certificados *JKS* utilizados para o acesso das informações dos servidores, e as portas utilizadas para a comunicação com o *IntSitef*

Na Figura 4.5 é mostrado o módulo de configuração de certificados no ambiente da aplicação *IntSitef*.

Figura 4.5 – Configurador de Certificados



Fonte: IntSitef

Para a configuração de certificados existe os seguintes campos:

- Arquivo *JKS*, arquivo de certificação configurado tanto para o servidor, quanto para o cliente.
- Senha do *JKS*, senha do arquivo de certificação configurado tanto para o servidor, quanto para o cliente.
- Porta *HTTP*, configuração da porta *HTTP* (HyperText Transfer Protocol) da aplicação *IntSitef*.
- Porta *HTTPS*, configuração da porta *HTTP* (Hyper Text Transfer Protocol Secure) da aplicação *IntSitef*.

4.8 Aproveitamento de Logs

Para otimizar o entendimento e o funcionamento da aplicação *IntSitef*, foi desenvolvido um módulo de visualização de logs, sendo possível verificar todos os eventos, erros e avisos que ocorreram em um determinado período de tempo.

Esse módulo, permite a visualização separada por tipo e serviço, fazendo com que os logs fiquem organizados e facilitando uma visão analítica, o que possibilita consultas rápidas entre diversos servidores.

4.9 Formalização de Erros

Em todos os fluxos de mensagens da aplicação foi aplicado um tratamento para erros, com o intuito de formalizar e padronizar os erros com o aplicativo que faz requisição. Dessa forma, pode-se mapear, quais os possíveis erros, ajudando, assim, na detecção de problemas entre as pontas.

Para o desenvolvimento desta atividade foram utilizadas estratégias de validação em todos os campos de todas as requisições que não necessitam de interface, para que se fosse obtido um melhor resultado nas validações.

5 CONSIDERAÇÕES FINAIS

Em ambiente de homologação, o sistema apresentou resultados satisfatórios, deixando melhorias para atender toda a demanda que não estava em especificação anteriormente. Com o sistema em produção, foi possível filtrar um grande fluxo de requisições tratadas pelo IntSitef. O sistema obteve sucesso em todas as suas tarefas, se mostrando robusto e de fácil usabilidade, se mostrando uma aplicação de grande importância dentro do escopo global.

Os resultados obtidos pelo estagiário foram o crescimento profissional e pessoal. O crescimento profissional se deu pela oportunidade de estagiar em uma empresa líder de mercado, com profissionais experientes que, de alguma forma, mostraram os atalhos para um bom trabalho. Além disso o estágio possibilitou participar de uma equipe, participar de análise de especificação, de trabalhar em grupo, de tratar com os clientes, e principalmente, na resolução de problemas. Tudo isso contribuiu para o desenvolvimento tanto pessoal quanto profissional do estagiário.

Iniciando o estágio com um breve conhecimento na linguagem Scala, Orientação a Objetos e em sistema gerenciador de banco de dados, o estagiário obteve sucesso no aprendizado dessas tecnologias colocando em prática o que foi aprendido na parte teórica de matérias como Engenharia de Software, Linguagem de Programação, Programação Orientada a Objetos, Estruturas de dados, Banco de Dados, Processos de Software, entre outras das quais foi possível aplicar na prática o conhecimento adquirido em sala de aula.

Foi de grande aprendizado, a preocupação com boas práticas de programação utilizadas durante o projeto, a utilização correta do modelo MVC e da realização de testes. A partir dessas técnicas, foi possível a criação de um produto robusto e de fácil manutenção, fixando a ideia de qualidade de software, pois não basta conseguir resolver o problema e sim resolver da melhor forma possível tendo como base toda a documentação existente para a linguagem.

Além disso, o conhecimento do mercado de meios de pagamentos, e a convivência com grandes profissionais fizeram parte do conhecimento mais valioso no período de estágio, abrindo horizontes do mercado de trabalho e da importância do trabalho em equipe.

6 REFERÊNCIAS BIBLIOGRÁFICAS

COULOURIS, George; et al.**Concepts and Design.3rd edition**, 2003.

DOCKER. **Documentation**. Disponível em:**<https://docker.com/ >**. Acesso em: 12 abril 2019.

DOOLEY, J. **Software Development and Professional Practice**. Apress, 2011.

FOREIGN DATA WRAPPERS. Disponível em:
.**<https://wiki.postgresql.org/wiki/Foreign_data_wrappers>**. Acesso em: 20 de maio de 2019.

FUKS, H.; et al. **Do Modelo de Colaboração 3C à Engenharia de Groupware**, 2003.

GOMES, A.**JavaScript – Aplicações Interativas para a Web**, 2006.

GRAYLOG. **GrayLog Documentation**. Disponível em: **<http://docs.graylog.org/en/3.0/>**.
Acesso em: 12 abril 2019.

MERCURIAL. **Documentation**. Disponível em: **<https://www.mercurial-scm.org/guide>**.
Acesso em: 12 de abril de 2019.

PLAY FRAMEWORK. **Documentation**. Disponível em:
<https://playframework.com/documentation >. Acesso em: 12 abril 2019.

SAMY, M. **Construindo Sites com CSS e HTML**, 2015.

SBT. **Documentation**. Disponível em: **<https://www.scala-sbt.org/1.x/docs/index.html>**.
Acesso em: 12 de abril de 2019.

SCALA. **Documentation**. Disponível em: **<https://docs.scala-lang.org**. Acesso em: 12 maio 2019.

VSC. **Documentation**. Disponível em: **<https://code.visualstudio.com/docs>**.
Acesso em: 12 de abril de 2019.