CODE SIMATIC – Relatórios de BI com Dados Gerados pelo CLP SIMATIC S7-1200

Alisson da Silva Bueno, Anderson José de Souza Inácio, Everton Luiz Sausen, Fernando André de Lima, Joziel Alves de Souza

Bacharelado em Sistemas de Informação – Faculdades da Indústria IEL
São José dos Pinhais – PR – Brasil

Abstract. *This project aims to perform data recovery of a SIMATIC S7-1200 CLP. The collected data was stored in a database and through this data will be generated BI reports for the requesting company. Reports will be available in an Android application.*

Resumo. *Este projeto tem como objetivo realizar a recuperação de dados de um CLP SIMATIC S7-1200. Os dados coletados ficaram armazenados em banco de dados e através destes dados serão gerados relatórios de BI para a empresa solicitante. Os relatórios ficarão disponíveis em um aplicativo Android.*

1. Introdução

Em um mundo corporativo globalizado como o de hoje, onde a concorrência pode vir do outro lado do mundo, ter domínio do conhecimento é um dos requisitos para a sobrevivência do negócio. Entretanto ter apenas o conhecimento é insuficiente se a empresa não possui controle total sobre seus meios de produção. Nunca se precisou tanto de informação em tempo real. Saber quanto seu negócio está produzindo e quais os problemas que o atinge neste exato momento permitem que a empresa possa dar respostas imediatas a eventos não planejados, fazer projeções mais realistas sobre seus lucros e perdas, planejar estratégias com muito mais acuracidade. Os avanços tecnológicos na indústria, principalmente o uso intensivo de controladores lógicos programáveis (CLP), permitiram a automação massiva de linhas de produção, além da recuperação e acompanhamento em tempo real destes dados através da internet. Isso permitiu o gerenciamento inteligente e eficiente do negócio e ao mesmo tempo permitiu a utilização de ferramentas que iram auxiliar a tomada de decisões da gerência, como softwares de BI (*Business Intelligence*).

2. Fundamentação Teórica

A Fundamentação Teórica do projeto foi segmentada em duas áreas: Negócio e Tecnologias Empregadas.

2.1. Fundamentação do Negócio

A empresa Porto Seguro Cosméticos, que pertence ao grupo Leclair, nos propôs ajudá-la a recuperar informações de uma de suas máquinas de envase de frascos de perfumes. A máquina em específico possui um CLP (Comando Logico Programável) que é capaz de armazenar informações de operação, mas, no momento, esses dados não são utilizados para nenhum fim. Assim a Porto Seguro fica na dependência de relatórios manuais que, por muitas vezes estão indisponíveis ou são imprecisos.

Nossa parceira precisa que informações coletadas e armazenadas no CLP SIMATIC S7-1200 da máquina seja disponibilizada através de um servidor *web* próprio. Essas informações devem gerar relatórios de produção, como por exemplo, quanto tempo a máquina ficou parada em um determinado período. Esses relatórios ficaram disponíveis tanto no ambiente de intranet da empresa quanto em um aplicativo Android, para a consulta da gerência e para a alimentação de outras ferramentas de BI da empresa.

2.2. Fundamentação das Tecnologias

Para o entendimento completo do trabalho, foi preciso pesquisar e utilizar diversas tecnologias, das quais listamos abaixo.

2.2.1. Android

Android é um sistema operacional baseado no núcleo Linux e atualmente desenvolvido pela Google. O sistema operacional utiliza-se de telas sensíveis ao toque para que o usuário possa manipular objetos virtuais, embora não se limite apenas a aparelhos que possuam *touchscreem*. Atualmente, o sistema operacional móvel do Google é o mais utilizado em todo o mundo, e está presente em milhares de aparelhos.

O Android surgiu em 2003 desenvolvido pelos empresários Andy Rubin, Rich Miner, Nick Sears e Chris White e que fundaram a Android Inc. Segundo Rubin, o objetivo do Android é disponibilizar "Dispositivos móveis mais inteligentes e que estejam mais cientes das preferências e da localização do seu dono". Na época, Rubin e sua equipe ofereceram um novo meio de sistema operacional móvel baseado no Kernel Linux, ou seja, *Open Source*. Em 2005 o Google adquiriu o Android Inc, e com isso nasceu a Google Mobile Division, divisão de pesquisa em tecnologia móvel da maior empresa do mundo de tecnologia.

Mas o Android não roda somente em *smartphones* e *tablets*. A aplicação em diversos aparelhos como *netbooks*, câmeras digitais, s*mart* tv's e ainda estar à frente da revolução da internet das coisas.

Escolhemos utilizar um aplicativo desenvolvido para Android devido a grande aplicabilidade, a quantidade de documentação e de bibliotecas, além de para atender as exigências do projeto integrador, cobrindo a atual grade curricular. Será através de um aplicativo Android que exibiremos para o usuário as informações recuperadas do CLP da máquina.

2.2.2. Android Studio

Android Studio é um ambiente de desenvolvimento integrado (IDE) para desenvolver para a plataforma Android. Foi anunciado em 16 de Maio de 2013 na conferência Google I/O. Android Studio é disponibilizado gratuitamente sob a Licença Apache 2.0. A primeira compilação estável foi lançada em Dezembro de 2014, começando da versão 1.

Antigamente, o desenvolvimento para Android no Brasil era difundido utilizando a plataforma Eclipse com o Android Development Kit (ADK), fornecido pelo Google, que lançou a plataforma Android Studio. Este é baseado no IntelliJ IDEA, que é uma IDE que também oferece suporte ao Android, mas possui um custo elevado.

A IDE Android Studio possui algumas vantagens como, por exemplo, o gerenciador de dependências Gradle (vide seção *Links*), também baseado no IntelliJ, muito utilizado fora do Brasil. Este é um dos grandes trunfos do editor da plataforma,

pois oferece mais opções ao desenvolvedor na hora de compilar, já que o Eclipse utilizava o jeito clássico de compilação.

2.2.3. CLP

O CLP ou controlador lógico programável é um tipo especial de computador muito utilizado não somente na indústria, mas em controles de máquinas e processos em diferentes aplicações. Sendo um computador, este dispositivo compartilha termos comuns de um computador pois ele é composto por uma CPU (*Central Processing Unit* ou processador), memória RAM (memória de leitura e gravação) e ROM (memória de apenas leitura) e também portas de comunicação (COMs).

2.2.4. SIMATIC S7-1200

O SIMATIC S7-1200 é modular, compacto e versátil. Apresenta interface PROFINET e poderosas funções tecnológicas integradas em um projeto flexível. Isso permite uma comunicação simples e soluções eficientes para exigências tecnológicas, que são perfeitamente adequadas às necessidades de automação e uma ampla variedade de aplicações.

O SIMATIC S7-1200 além da interface Profinet também permite acesso remoto, bem como diversas funções tecnológicas integradas, além de alta flexibilidade. Isto permite comunicação simples, soluções eficientes para atividades tecnológicas e um encaixe perfeito a requisitos individuais de automação em uma grande variedade de aplicações.

2.2.5. PostgresSQL

O PostgreSQL é um poderoso sistema de banco de dados objeto-relacional de código-fonte aberto que usa e estende a linguagem SQL combinada com muitos recursos que armazenam e dimensionam com segurança as cargas de trabalho de dados mais complicadas. As origens do PostgreSQL datam de 1986 como parte do projeto POSTGRES da Universidade da Califórnia em Berkeley e têm mais de 30 anos de desenvolvimento ativo na plataforma central.

O PostgreSQL ganhou uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da comunidade de código aberto por trás do *software* para fornecer soluções eficazes e inovadoras de maneira consistente.

2.2.6. Rest

Representational State Transfer, abreviado como REST, é um dos modelos de arquitetura que foi descrito por Roy Fielding, um dos principais criadores do protocolo HTTP, em sua tese de doutorado e que foi adotado como o modelo a ser utilizado na evolução da arquitetura do protocolo HTTP.

Muitos desenvolvedores perceberam que também poderiam utilizar o modelo REST para a implementação de *webservices*, com o objetivo de se integrar aplicações pela *Web*, e passaram a utilizá-lo como uma alternativa ao SOAP.

REST na verdade pode ser considerado como um conjunto de princípios, que quando aplicados de maneira correta em uma aplicação, a beneficia com a arquitetura e padrões da própria *Web*.

2.2.7. Wildfly

O WildFly, que antes se chamava JBoss AS, é um servidor de aplicações *open source*, escrito em Java, baseado nos padrões definidos pela especificação Java EE e mantido pela comunidade e pela empresa Red Hat. Em 1999 foi lançada a primeira versão, com o nome EJBoss (Enterprise Java Beans in Open Source System). Era um servidor que atendia à especificação EJB 1.0. Em 2011 foi lançada o JBoss AS 7.0, quando houve a maior alteração de todas as versões. A arquitetura do JBoss foi remodelada em módulos utilizando OSGi, o que fez o servidor de aplicações se tornar extremamente mais rápido. A partir de 2013 o projeto foi renomeado para WildFly, pois já existiam vários outros projetos com o nome JBoss e também para não confundir com a versão comercial do servidor de aplicação, chamada JBoss EAP. Assim sendo, não existe JBoss AS 8, e sim WildFly 8;

2.2.8. Java8

O Java 8 é a *release* mais recente do Java que contém novas funcionalidades, aprimoramentos e correções de *bug* para aumentar a eficiência do desenvolvimento e execução de programas Java. O destaque do Java SE 8 é a implementação de expressões Lambda e funcionalidades de suporte para a linguagem de programação e plataforma Java. Algumas melhorias se destacam no Java 8, como:

- API de Data e Hora que permite os desenvolvedores tratarem data e hora de maneira mais natural, clara e fácil de entender.
- Nashhorn JavaScript Engine, uma nova implementação leve de alto desempenho do motor JavaScript foi integrada ao JDk e está disponível para aplicações Java por meio das APIs existentes.
- Maior Segurança: A lista manual existente de métodos sensíveis do chamador foi substituída por um mecanismo que identifica com precisão esses métodos e permite que seus chamadores sejam descobertos com confiança.

2.2.9. JavaEE8

O Java EE 8 apresenta a JSON Binding API (JSON-B) para mapeamento entre objetos JSON text e Java, com base na JSON Processing API (JSON-P). O *servlet* foi aprimorado com o suporte adicional para o protocolo HTTP / 2. O JAX-RS adiciona suporte a eventos enviados pelo servidor e, com base em recursos de simultaneidade incluídos no Java SE 8, uma API do cliente reativa. A nova API de segurança do Java EE fornece suporte aprimorado para autenticação e autorização em módulos da w*eb* e também apresenta APIs para acesso a armazenamentos de identidades. O Bean Validation é atualizado para refletir os aprimoramentos feitos no Java SE 8 e para estender o intervalo de objetos validados.

O Java EE 8 leva adiante o conceito de perfis. Um perfil é uma coleção de tecnologias e APIs do Java EE que abordam comunidades de desenvolvedores e tipos de aplicativos específicos.

2.2.10. HTML

HTML é uma das linguagens que utilizamos para desenvolver *web sites*. O acrônimo HTML vem do inglês e significa *Hypertext Markup Language* ou em português Linguagem de Marcação de Hipertexto. O HTML é a linguagem base da internet. Foi criada para ser de fácil entendimento por seres humanos e também por

máquinas, como por exemplo o Google ou outros sistemas que percorrem a internet capturando informação.

O criador da linguagem foi Tim Berners-Lee. Ele criou o HTML para a comunicação e disseminação de pesquisas entre ele e seu grupo de colegas. O HTML ficou bastante conhecido quando começou a ser utilizada para formar a rede pública daquela época, o que se tornaria mais tarde a internet que conhecemos hoje.

2.2.11. jQuery

jQuery é uma biblioteca JavaScript desenvolvida por John Resig, e que se tornou uma das bibliotecas JavaScript mais populares na internet, de código aberto e disponibilizada sobre as licenças MIT e GPL.

Sua criação teve como foco a simplicidade e o objetivo de facilitar o desenvolvimento de aplicações que necessitariam de muitas linhas de código para obtermos um determinado efeito, ou efetuar uma requisição Ajax. Com jQuery esse trabalho é substituído por poucas instruções, o que faz da jQuery uma ferramenta ideal para designers e desenvolvedores com pouco conhecimento em JavaScript. jQuery é uma biblioteca de funções em JavaScript que interage com o HTML, desenvolvida para simplificar os *scripts* interpretados no navegador do usuário (*client-side*). Usada por cerca de 77% dos 10 mil sites mais visitados do mundo, jQuery é a mais popular das bibliotecas JavaScript.

A sintaxe do jQuery foi desenvolvida para simplificar a navegação em documentos HTML, a seleção de elementos DOM, criar animações, manipular eventos, desenvolver aplicações AJAX e criação de *plugins* sobre ela. Permitindo aos desenvolvedores criarem camadas de abstração para interações de baixo nível de modo simplificado em aplicações *web* de grande complexidade.

2.2.12. JSON

JSON (*JavaScript Object Notation – Notação de Objetos JavaScript*) é uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar. Está baseado em um subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3a Edição-Dezembro – 1999. JSON é em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados.

JSON está constituído em duas estruturas: A primeira é coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *object, record, struct*, dicionário, *hash table, keyed list*, ou *arrays* associativas. A segunda é lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência.

Estas são estruturas de dados universais. Virtualmente todas as linguagens de programação modernas as suportam, de uma forma ou de outra. É aceitável que um formato de troca de dados que seja independente de linguagem de programação se baseie nestas estruturas.

Apesar de muito simples, tem sido bastante utilizado por aplicações *web* devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, tornando mais rápido o *parsing* dessas informações. Isto

explica o fato de o JSON ter sido adotado por empresas como Google e Yahoo, cujas aplicações precisam transmitir grandes volumes de dados.

2.2.13. Gson

O Gson é uma biblioteca Java que pode ser usada para converter objetos Java em sua representação JSON. Ele também pode ser usado para converter uma sequência JSON em um objeto Java equivalente. O Gson pode trabalhar com objetos Java arbitrários, incluindo objetos preexistentes dos quais você não possui código-fonte.

Existem alguns projetos de código aberto que podem converter objetos Java em JSON. No entanto, a maioria deles exige que você coloque anotações Java nas suas classes, algo que você não pode fazer se não tiver acesso ao código-fonte. A maioria também não suporta totalmente o uso de *Java Generics*. Gson considera ambos como metas de design muito importantes.

Entre os objetivos do Gson estão: Fornecer métodos *toJson()* e *fromJson()* métodos simples para converter objetos Java em JSON e vice-versa; Permitir que objetos não modificáveis, preexistentes sejam convertidos para e de JSON; Suporte extensivo de *Java Generics*; Permitir representações personalizadas para objetos; Suporta objetos arbitrariamente complexos (com hierarquias de herança profundas e uso extensivo de tipos genéricos).

3. Desenvolvimento

O projeto CODE SIMATIC foi criado devido à necessidade de se obter os dados gerados por uma máquina que possui o CLP S7-1200, salvá-los e gerar relatórios de BI que possam ser visualizados em um aplicativo Android nativo.

3.1. Delimitação do Projeto

O primeiro passo foi a delimitação do tema do projeto em concordância com o cliente. Foi definida qual necessidade a ser suprida e então foi realizado o levantamento das restrições do projeto.

A solução do projeto precisou ser desenhada de forma que englobasse o CLP S7, o dispositivo Android, o banco de dados e o servidor que conecta todos os componentes físicos do projeto. Abaixo segue o diagrama que possui todos os hardwares do projeto.

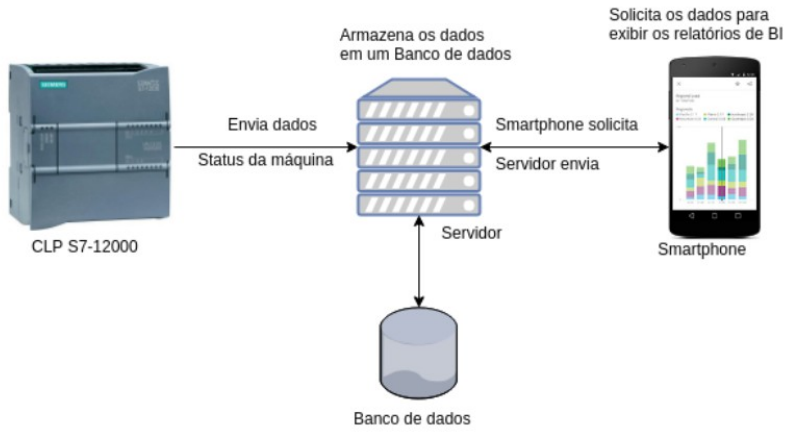

Figura 1 – Hardware do Projeto

As restrições do projeto, em partes, são decorrentes da máquina já ter um *webservice* próprio que exporta os dados em HTML dificultando a coleta e segmentação dos dados. O CLP possui tela própria que exibe os status da máquina, como, por exemplo, se a mesma está parada (STOP) ou funcionando (RUN), conforme imagem a seguir:

Figura 2 – Interface do CLP

A máquina também tem momentos de manutenção e baixa de produção dificultando os testes na coleta de dados.

3.2. Definição das Telas

Apos restringir o projeto, iniciou-se o planejamento de quais tipos de gráficos seriam possíveis gerar na aplicação. Também iniciou-se a definição de quais as telas básicas que o sistema deveria ter. Finamente o grupo foi capazes de definir como seria apresentado a aplicação para o usuário final

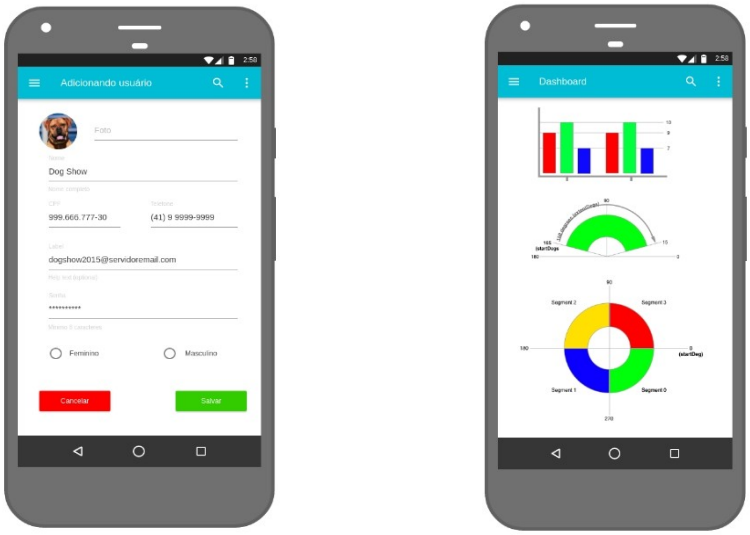

Figura 3 – Protótipo das Telas da Aplicação

O componente de Dashboard mostrando de forma visual as informações geradas pela SIMATIC, buscando facilitar a compreensão. Desenvolvemos todos os protótipos antes de definitivamente recriá-los no Android Studio. Foi desenhado telas de interação básicas do usuário, como login de acesso e cadastro de usuário, até exemplos de como poderiam ser exibidos os gráficos gerados.

3.3. Desenvolvimento dos Códigos

Com a definição do projeto e os requisitos levantados, iniciou-se o desenvolvimento do backend da aplicação.

3.3.1. Descrição da Arquitetura

A primeira etapa do backend do projeto foi entender e definir o fluxo da aplicação. Somente após foi possível desenvolver o código em si. O fluxo se divide em dois.

O fluxo do projeto de inicia no CLP, que gera a informação e grava em um banco de dados interno. Os dados passam então para o *webserver* do próprio S7-1200, um código javascript gera o arquivo json com as informações do CLP. O próprio *webserver* envia estes json para um servidor *web* que ira gravar as informações contidas no arquivo em outro banco de dados.

O segundo fluxo se inicia com a requisição de dados feita pelo aplicativo Android. O servidor então busca os dados no banco e envia-os novamente para a aplicação Android. A aplicação processa os dados, gera os gráficos e exibe-os para o usuário final.

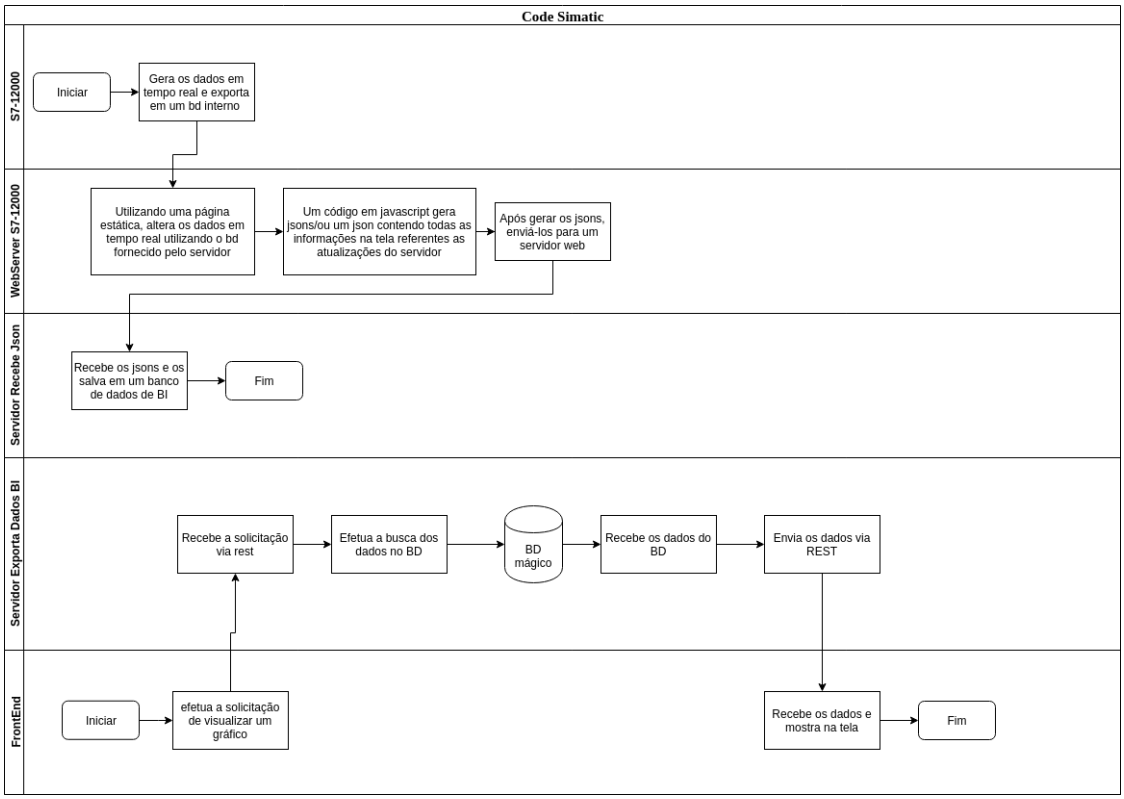

Figura 4 – Fluxo da aplicação

3.3.2. Padrões e Mecanismos Arquiteturais

Toda a aplicação foi desenvolvida utilizando o padrão MVC (*Model, View* e *Control*), um padrão de arquitetura de software, separando a aplicação em 3 camadas. A camada de interação do usuário (*view*), responsável pela exibição dos dados. A camada de manipulação dos dados (*model*), responsável pela leitura e escrita de dados, e também de suas validações. A camada de controle (*controller*), responsável por receber todas as requisições do usuário.

3.4. Obtenção das Informações

Durante o levantamento de requisitos, recebemos um projeto gerado no software SIMATIC STEP 7 Professional, conhecido também como TIA Portal, nele efetuamos a análise dos parâmetros que estavam configurados e conseguimos compreender a relação detes com as informações mostradas no HTML que está hospedado dentro do *Web Server* do controlador.

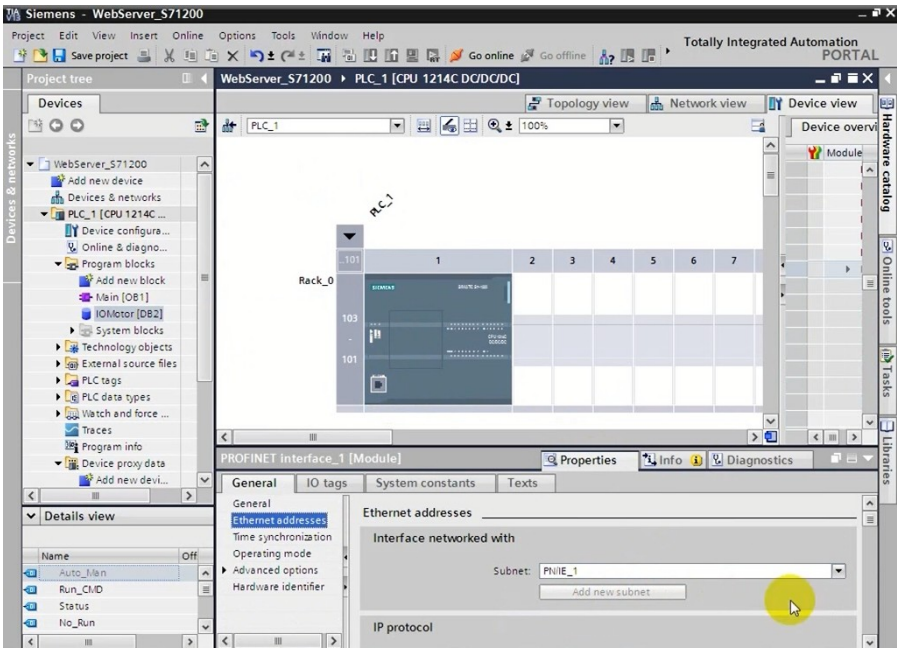

Figura 5– Software TIA Portal

- Os parâmetros encontrados foram:
- Velocidade de rotação da máquina, encontrado como Speed_PV;
- Potência da máquina, encontrado como Power;
- Contador de vezes em que a máquina ficou parada, encontrado como No_Run;
- Identificação de funcionamento automático, sem a possibilidade de alterar a potência, encontrado como Auto_Man;
- Identificação de funcionamento manual, no qual é possível alterar a potência da máquina manualmente, encontrado como Run_CMD;
- Identificação do status da máquina de ligado ou desligado, encontrado como Status.

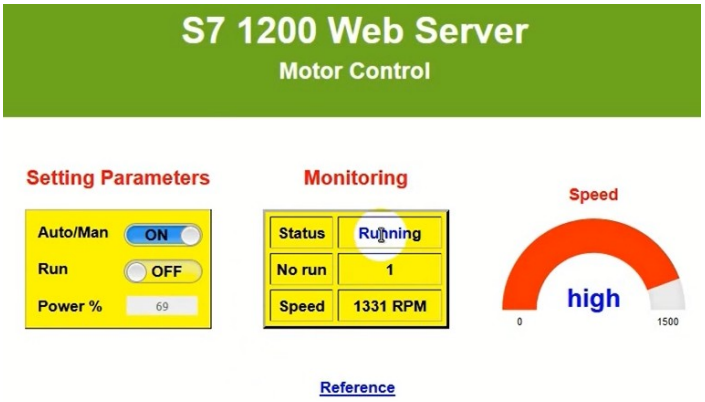

Figura 6 – HTML com dados parametrizados com o CLP

4. Resultados

Simulado a geração de dados idênticos aos dados esperados pela máquina, foi possível enviar estes dados e armazená-los no banco de dados. Estes mesmos dados foram então recuperados do banco e enviadas para a API responsável por gerar os gráficos no aplicativo Android.

Ao acessar o aplicativo, o usuário deve selecionar qual tipo de informação ele deseja ter. Existem três opções disponíveis de relatórios:

- Porcentagem Ligado e Desligado: mostra o percentual de tempo que a máquina ficou ligada e desligada.
- Funcionamento Manual e Automático: moatra o percentual de tempo que a máquina ficou em cada um dos modos disponíveis, manual e automático.
- Potencia Utilizada na Maquina: mostra o percentual de tempo que a máquina operou em cada faixa de potência indo de 25% até 100% (máquina funcionando na potência máxima, 1500 RPM)

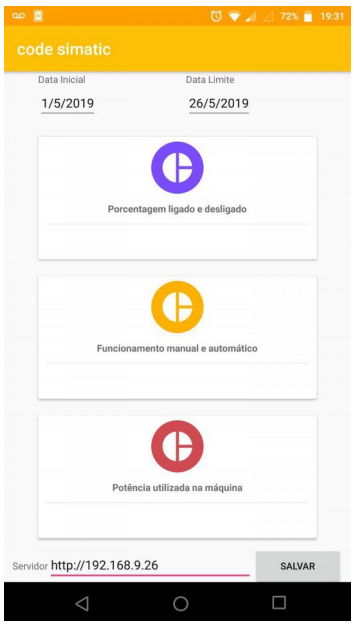

Figura 7 – Tela de Seleção do Gráfico

A API aguarda que seja selecionado, no aplicativo Android, um range (intervalo) de período para a avaliação. Esse período é selecionado a partir de um calendário exibido na tela do aplicativo.

Figura 8 – Tela de Seleção do Range de Datas

Os gráficos foram gerados e atualizados em tempo real da aplicação, conforme foi previsto no início do projeto.

As informações são exibidas sempre em gráficos tipo pizza, desta forma melhorando o entendimento do usuário.

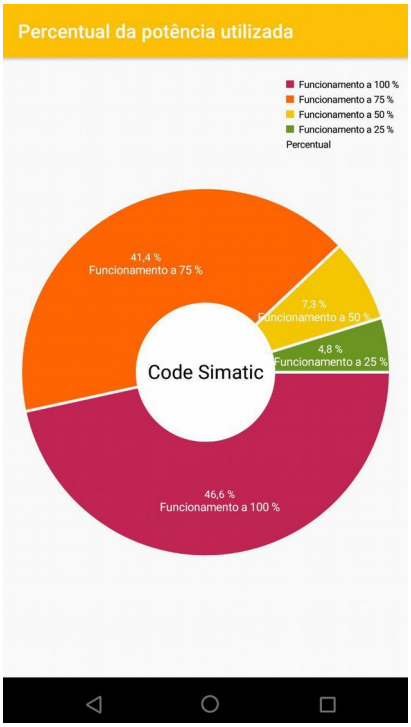

Figura 9 – Gráfico Gerado pela API no Aplicativo

5. Considerações Finais

Este projeto se mostrou um verdadeiro desafio para o grupo. Seu desenvolvimento englobou diversos conceitos visto tanto nos semestres anteriores quanto neste semestre. Foi preciso estudar e utilizar novas ferramentas, até então não vistas por nenhum dos membros do grupo, como por exemplo o Android Studio.

Tivemos inúmeras dificuldades e percalços durante o desenvolvimento. Problemas com a máquina, que ficou parada devido a manutenções não programadas, falta de acesso direto aos dados e Indisponibilidade do principal contato do grupo dentro da fábrica, fez com que fossemos obrigados a procurar soluções alternativas para desenvolver a solução do projeto proposto.

Todos os problemas enfrentados, são problemas recorrentes em qualquer indústria, o que deu a equipe uma experiência completa sobre como é gerir este tipo de projeto. Atingimos boa quantidade de objetivos levantados no início e com isso podemos afirmar que o desenvolvimento do projeto ao longo do semestre teve um bom aproveitamento e resultados dentro o esperado.

6. Referencias

Crockford, Douglas. Introdução ao JSON. JSON, 2019. Disponível em: <https://www.json.org/json-pt.html>. Acesso em: 24 de maio de 2019.

Eis, Diego. O básico: O que é HTML?. Tableless, 21 de jan. de 2011. Disponível em: <https://tableless.com.br/o-que-html-basico>. Acesso em: 24 de maio de 2019.

Ferreira, Rodrigo. REST: Princípios e boas práticas. Caelum, 23 de set. de 2017. Disponível em: <https://blog.caelum.com.br/rest-principios-e-boas-praticas>. Acesso em: 04 de maio de 2019.

Gson. GitHub, 2019. Disponível em: <https://github.com/google/gson>. Acesso em: 24 de maio de 2019.

Informações do Java 8. Oracle, 2019. Disponível em: <https://https://www.java.com/pt_BR/download/faq/java8.xml>. Acesso em: 24 de maio de 2019.

MEYER, MAXIMILIANO. A história do Android [Atualizado Android Pie 9.0]. Oficina da Net, 05 de out. de 2018. Disponível em: <https://www.oficinadanet.com.br/post/13939-a-historia-do-android >. Acesso em: 04 de maio de 2019.

RAMOS, ALLAN. O que é MVC?. Tableless, 26 de fev. de 2015. Disponível em: <https://tableless.com.br/mvc-afinal-e-o-que/>. Acesso em: 01 de junho de 2019.

ROMANATO, ALLAN.Tutorial de Android Studio. devmedia, 2016. Disponível em: <https://www.devmedia.com.br/tutorial-de-android-studio/34003>. Acesso em: 24 de maio de 2019.

SCHMIDT, ADRIANO. WildFly – Do básico ao ambiente de produção. Devmedia, 2015. Disponível em: <https://www.devmedia.com.br/wildfly-do-basico-ao-ambiente-de-producao/33653 >. Acesso em: 24 de maio de 2019.

Silveira, Cristiano Bertulucci. Saiba Tudo Sobre CLP. CitiSystems, 2019. Disponível em: <https://www.citisystems.com.br/clp/ >. Acesso em: 24 de maio de 2019.

SIMATIC S7 – 1200 [Manual]. Siemens, 2019. Disponível em: <https://w3.siemens.com.br/automation/br/pt/automacao-e-controle/automacao-industrial/simatic-plc/s7-cm/s7-1200/Documents/ Brochura_SIMATIC_S7_1200_portugues.pdf >. Acesso em: 20 de abril de 2019.

Sobre o PostgreSQL. PostgreSQL, 2019. Disponível em: <https://www.postgresql.org/about/ >. Acesso em: 24 de maio de 2019.

TEIXEIRA, JOSÉ RICARDO. jQuery Tutorial. Devmedia, 2013. Disponível em: <https://www.devmedia.com.br/jquery-tutorial/27299>. Acesso em: 24 de maio de 2019.