



PROJETO 3E

EFICIÊNCIA ENERGÉTICA EM EDIFICAÇÕES

WWW.MMA.GOV.BR/CLIMA/ENERGIA/PROJETOS

BENCHMARKING DE EDIFÍCIOS PÚBLICOS

Relatório técnico de
desenvolvimento da plataforma
web

Maio
2017

DEO

DESEMPENHO
ENERGÉTICO
OPERACIONAL
EM EDIFICAÇÕES



CBCS
Conselho Brasileiro de
Construção Sustentável

Relatório Técnico de Desenvolvimento da plataforma *web*

Projeto: Benchmarking de desempenho energético em edifícios públicos

CBCS - Conselho Brasileiro de Construção Sustentável
Olavo Kucker - Presidente do Conselho Deliberativo
Vanderley M. John e Orestes M. Gonçalves - Diretores

PNUD: Celena Souza e Willian Zanetti
MMA: Alexandra Maciel, Camila Vasconcelos e Alessandra Silva

Equipe:

Prof. Roberto Lamberts - Coordenador do CT Energia CBCS
Edward Borgstein
Alexandre Schinazi
Braúllio Nunes
Rosane Fukuoka
Maxine Jordan
Adriano Perez
Carolina Furlanetto Mendes

São Paulo – Maio, 2017

Observação:

A divulgação, cópia, publicação, reprodução e distribuição, todas sem fins comerciais, do presente relatório pelo beneficiário do benchmarking é permitida, desde que seja feita referência expressa ao PNUD, ao Ministério do Meio Ambiente e ao GEF.

SUMÁRIO

PLATAFORMA	1
CONTEXTUALIZAÇÃO DO PROJETO	1
BENCHMARKING	1
PLATAFORMA	2
1. INTRODUÇÃO	2
2. FLUXO DE NAVEGAÇÃO	3
3. DESCRIÇÃO PLATAFORMA.....	8
4. AMBIENTES E SERVIDORES.....	8
4.1 REQUISITO MÍNIMO DE HARDWARE:.....	8
4.2 ESTRUTURAS E CÓDIGOS	9
4.3 FRONTEND E BACKEND	9
4.4 SERVICES	10
4.5 FOLHA DE ESTILOS	10
4.6 BOOTSTRAP	11
4.7 BANCO DE DADOS	11
4.8 RESTFUL API	12
5. INSTALAÇÃO DO SISTEMA - PACOTES E DEPENDÊNCIAS.....	13
6. DEFINIÇÃO DE SIGLAS E TERMOS	14

PLATAFORMA

Ferramenta web operacional descrita em documento técnico contendo: a) especificação da documentação do projeto; b) definição de siglas; c) busca de referências com especificações técnicas voltadas à orientação de equipes de Tecnologia da Informação que por ventura venham trabalhar com a calculadora e fazer ajustes futuros.

CONTEXTUALIZAÇÃO DO PROJETO

O Projeto 3E (Projeto PNUD BRA/09/G31 - “**Transformação do mercado de eficiência energética no Brasil**”) **compreende a construção de uma plataforma** online que servirá para entrada de dados de consumo energético e avaliação de edifícios através do benchmarking desenvolvido. Esta plataforma tem como objetivos:

- 1) Avaliações de estoques de edifícios, comparando os níveis de eficiência entre os diferentes edifícios da mesma organização, para determinar quais edifícios devem receber prioritariamente trabalhos de retrofit e diagnóstico energético (avaliar diversos edifícios ao mesmo tempo);
- 2) Direcionamento das estratégias de gestão de energia, com avaliações periódicas para identificação de problemas e potenciais de melhoria (avaliações regulares do mesmo edifício);
- 3) Medição e verificação de economias realizadas a partir de retrofits em determinados edifícios (avaliações pós-retrofit).

BENCHMARKING

A base do desenvolvimento do benchmark é a avaliação do comportamento do consumo energético de edifícios em diferentes condições e situações, de forma que seja possível entender os impactos de fatores externos e fatores de ocupação no seu desempenho. Para a captação de dados de consumo energético em edifícios públicos, foi realizada uma chamada pública para edifícios dos poderes executivo, legislativo e judiciário, nos planos municipal, estadual e federal, com área total acima de 500m² e tipologia de escritórios, que submeteram dados sobre os edifícios e contas de energia dos últimos 12 meses de operação. Dos edifícios que participaram da chamada pública, foram selecionados 20 para receberem medidores de energia e diagnósticos energéticos detalhados, identificando os seus níveis de desempenho, possíveis melhorias e outras ações requeridas.

Os dados medidos de consumo e o desenvolvimento de modelos arquétipos para simulações energéticas compõem a metodologia de desenvolvimento da equação de benchmarking.

O papel do benchmark é possibilitar comparações entre edifícios a fim de identificar seus níveis de eficiência de infraestrutura física e operacional. De acordo com os resultados das avaliações realizadas nos edifícios públicos selecionados, os fatores que precisam ser levados em consideração para definição de um benchmark adequado são:

- Área útil do edifício;
- Temperatura externa real;
- Densidade de população no edifício;

- Área do edifício que é condicionada;
- Consumo de energia em datacenters;
- Área de estacionamentos cobertos;
- Área de iluminação externa; e
- Uso de grupo gerador na hora de ponta.

PLATAFORMA

É uma ferramenta online que pode ser acessada de qualquer computador com acesso à internet, sendo uma plataforma de cálculo de benchmarking para edifícios públicos, que pretende ser uma ferramenta de fácil entendimento e de uso constante por gestores destes edifícios.

Com uma base de dados sólida, a plataforma oferece comparativos de consumo de energia que consideram, dentre outros aspectos, a temperatura ambiente de cada localidade, a ocupação e o uso de cada edifício.

Rodando em um browser (software para navegação na internet) o usuário não precisa de mais nada além de acesso à internet. A partir do registro de um e-mail e senha o sistema gera um cadastro simples para o gestor, que poderá incluir dados de múltiplos edifícios e comparar variáveis de cada um deles, bem como utilizar os gráficos e relatórios gerados pela plataforma para controlar o consumo energético anual comparando o consumo de cada edifício cadastrado com a média de consumo de edificações similares.

O sistema prevê ainda a possibilidade de inclusão de outras variáveis como graus médios calculados para cada cidade, bem como a geração de um histórico de consumo energético de edificações públicas com base nos dados cadastrados pelos usuários.

1. INTRODUÇÃO

Esse documento tem o intuito de facilitar o entendimento do ponto de vista da construção técnica da plataforma de cálculo de benchmarking para edifícios públicos, do conhecimento da estrutura, linguagem de programação e banco de dados, além de **plugins** e outros itens que foram utilizados na concepção, deixando transparente para qualquer profissional habilitado a desenvolver, manter ou aprimorar os códigos e funções construídas nesta versão.

2. FLUXO DE NAVEGAÇÃO

1. O usuário acessa a plataforma:




2. Realiza seu cadastro utilizando e-mail e senha:



3. Cadastra um novo edifício, inserindo as informações estruturais pertinentes:

BENCHMARKING ENERGÉTICO - Edifícios Públicos

Novo Edifício - Região



Estado


Escolher ▼

Cidade

Escolher ▼

CANCELAR PRÓXIMA

Novo Edifício - Estrutura



Edifício Público

Nome do Edifício*

Insira o nome

Área Útil (m²)* ?

m²

Área Estacionamento(m²) ?

m²

Número de Elevadores ?

m²

Campos com * são obrigatórios.

VOLTAR PRÓXIMA

4. Insere os dados de consumo do edifício no período de 12 meses que deseja calcular:

BENCHMARKING ENERGÉTICO - Edifícios Públicos

Novo Edifício - Consumo



Edifício
Público

Período do Cálculo ?
01/2015 12/2015

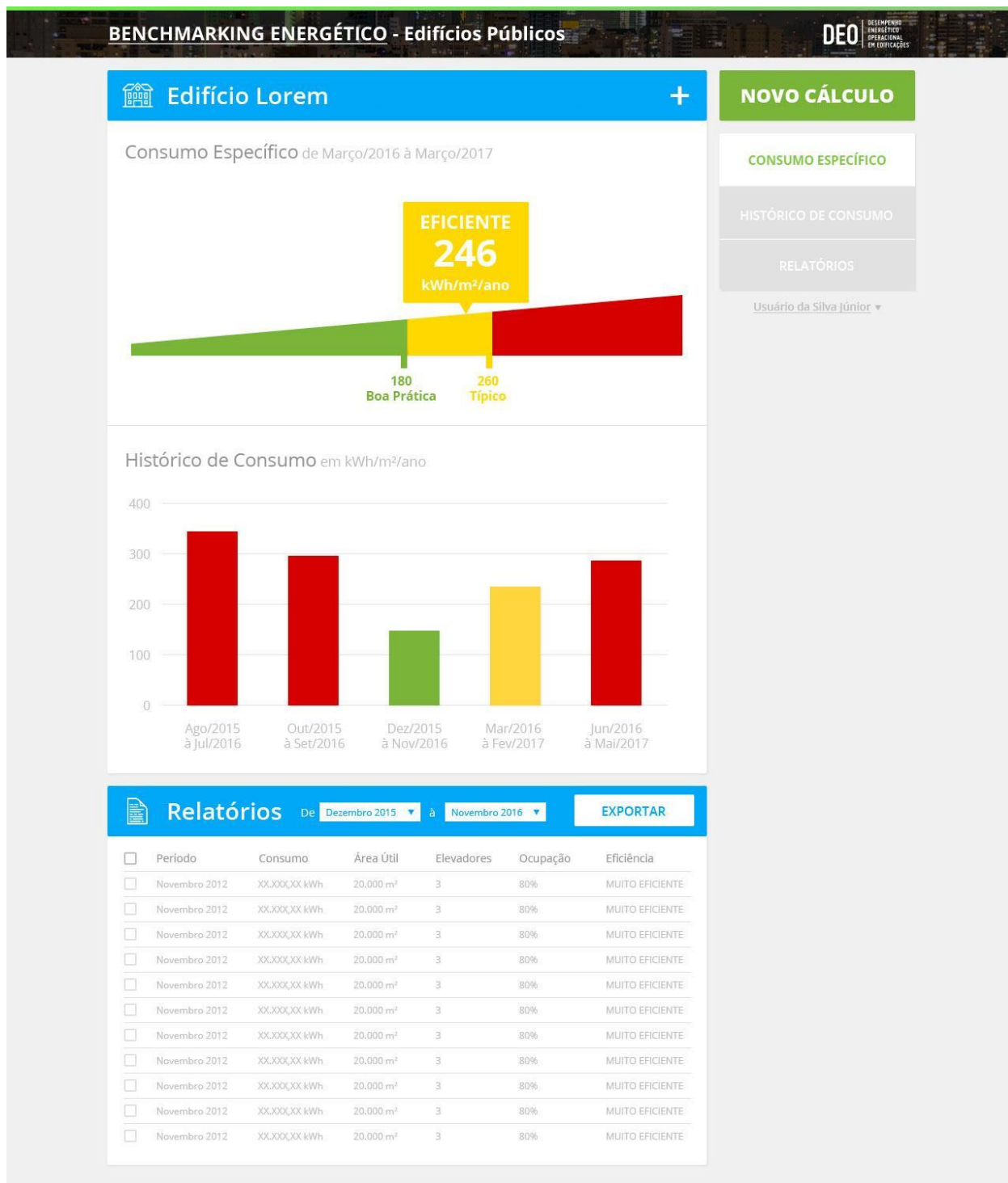
Consumo Total de Energia ?
MENSAL / **ANUAL**

kWh

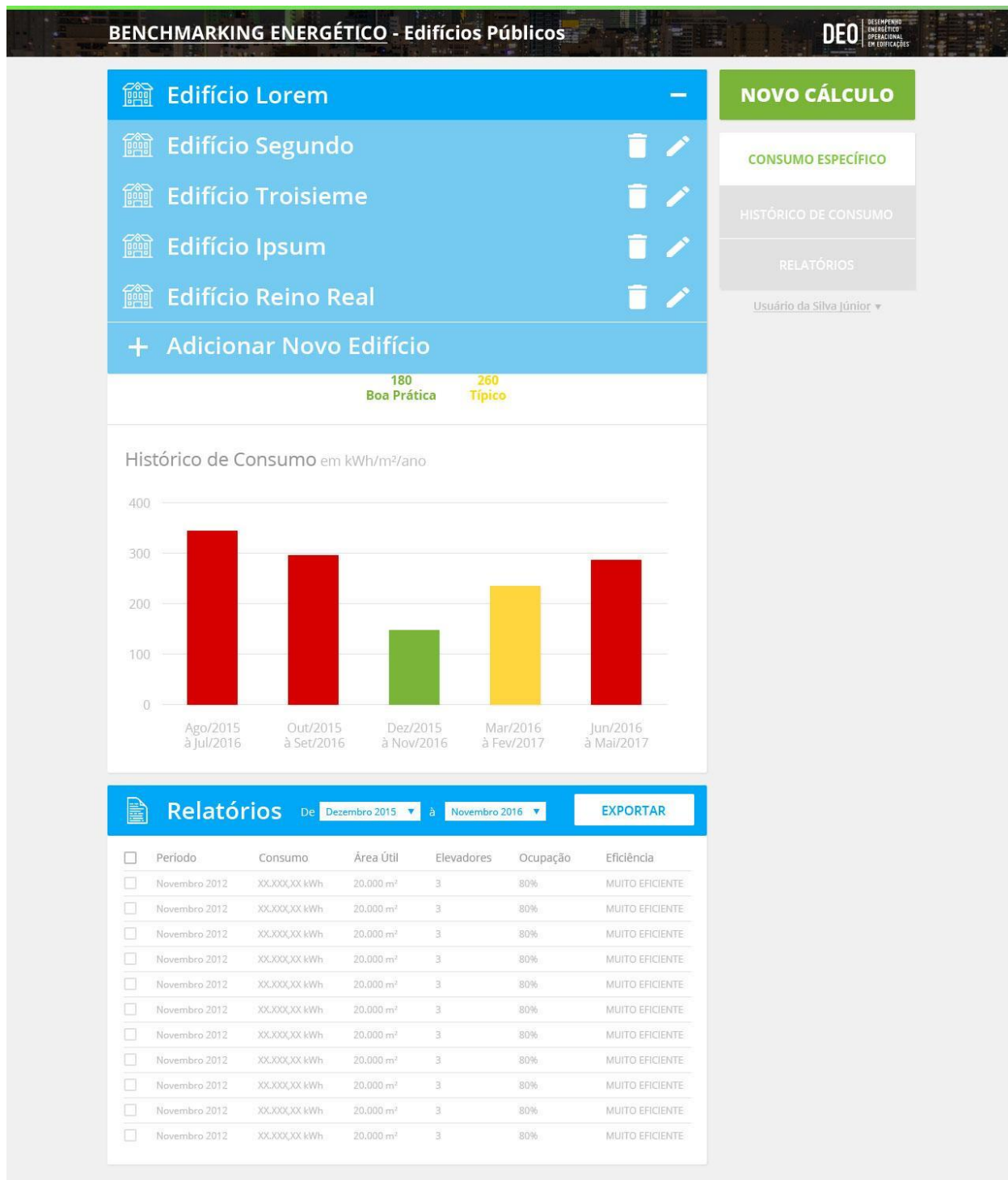
Taxa de Ocupação ?
%

VOLTAR CALCULAR

5. Após a inserção dos dados o usuário chega em um **dashboard**, que lhe apresenta o resultado do cálculo realizado e lhe permite visualizar cálculos anteriores:



6. A partir deste **dashboard** o usuário poderá realizar um novo cálculo, visualizar diferentes edifícios e adicionar novos, bem como exportar para uma planilha em excel os dados inseridos e os resultados obtidos:



3. DESCRIÇÃO PLATAFORMA

Como todo sistema baseado na internet, a plataforma de cálculo de benchmarking foi concebida com o pensamento 100% online, utilizando ambiente e servidores web em **cloud** (nuvem)

De forma macro, a plataforma é dividida em quatro camadas:

- FrontEnd: telas visuais acessadas pelo usuário que explicitam todas as regras de negócio aplicadas e devem guiar o usuário final na utilização do sistema sem acesso a códigos e sem a necessidade de conhecimento das técnicas de desenvolvimento.
- BackEnd: telas visuais acessadas pelo administrador do sistema, que possibilita e configuração de variáveis e outras informações necessárias para o funcionamento do sistema de forma visual sem a necessidade de acesso a códigos ou técnicas de desenvolvimento.
- Servidor ou Server: Sistema que recebe todos os inputs do FrontEnd e BackEnd e processa a informação conforme as regras de negócio e rotinas especiais criadas para tal.
- Banco de Dados: Onde o sistema grava e armazena os dados, tanto de configuração do sistema como dos inputs e resultados dos usuários. O banco de dados tem uma modelagem lógica criada especialmente para a plataforma de cálculo de benchmarking e é explicada detalhadamente nas **seções posteriores**.

4. AMBIENTES E SERVIDORES

Para manter os sistemas em operação, é utilizado um **cloud** (servidores interligados na nuvem de dados) na versão atual. Atualmente o sistema está hospedado no Cloud Amazon EC2 da empresa desenvolvedora, mas pode ser instalado em qualquer ambiente compatível conforme descrito abaixo:

4.1 REQUISITO MÍNIMO DE HARDWARE:

- 4 Gb de memória RAM;
- 500 Gb de espaço físico (HD);
- Acesso a internet moderado;
- Firewall para controle de acesso a portas nota 1;
- Espelhamento ou rotinas de backup real-time;
- Sistema operacional Linux Ubuntu de 64Bits versão 14.6 ou compatível;
- Servidor Node.js com pacotes especiais descritos nos próximos capítulos;
- Servidor Apache2;
- Banco de dados Mongo DB versão 4.0 ou superior.

4.2 ESTRUTURAS E CÓDIGOS

O sistema utiliza um MVC (**model, view, controller**) para a aplicação FrontEnd e BackEnd, além de um RESTful API para receber as chamadas no lado do servidor.

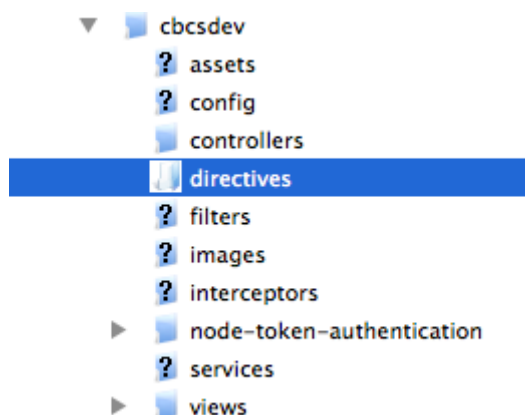
A aplicação BackEnd e FrontEnd foi desenvolvida 100% em Angular JS e utiliza uma estrutura de pastas descrita nos próximos capítulos.

Ainda no FrontEnd e BackEnd, foram utilizados **plugins** e **frameworks** para facilitar o desenvolvimento das interfaces e melhorar o UX (user experience), esses **plugins** serão descritos nos próximos capítulos.

O Server foi construído em Node.js utilizando Express e outros pacotes que também serão descritos nos próximos capítulos.

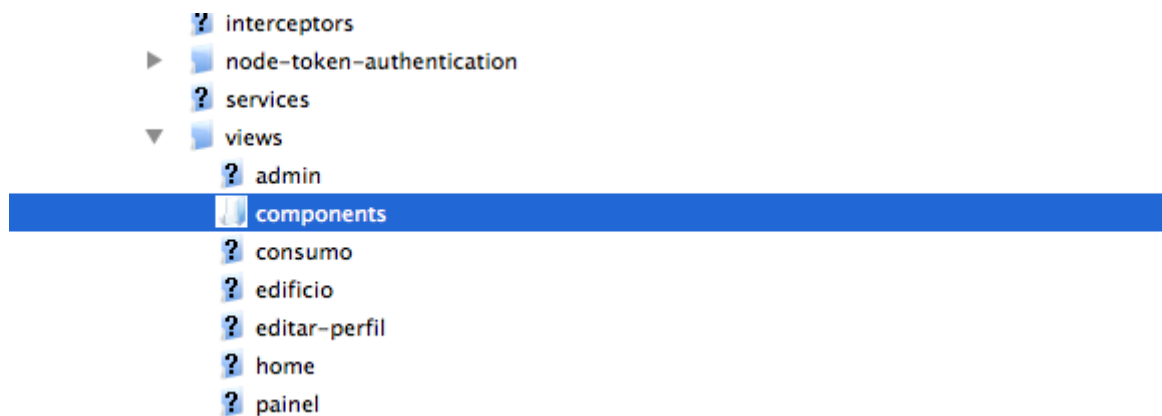
4.3 FRONTEND E BACKEND

Construído em Angular JS, utiliza um MVC (**module, view, controller**) além de diretivas e componentes construídos de forma estruturada para fácil manutenção e evolução do código, conforme exemplificado nas pastas abaixo:

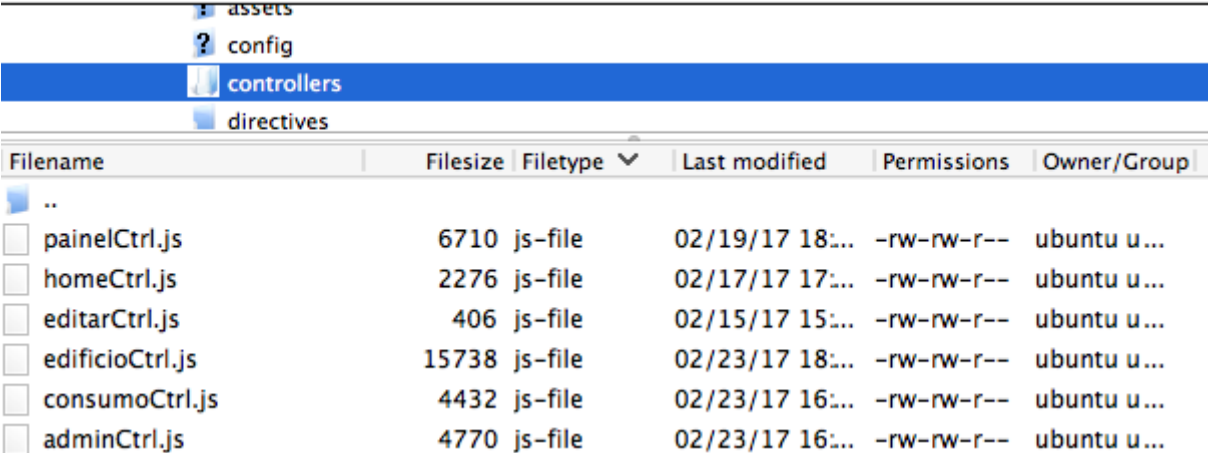


Com isso os HTMLs são criados na pasta “views”, os **controllers** na pasta “controllers” e assim por diante.

Importante atentar para o fato de que foram utilizados os componentes para as **views**, assim o código estruturado para os HTMLs ficaram componentizados conforme imagem abaixo:



Os **controllers** utilizam nomenclatura clara, o que facilita o entendimento. Cada seção do sistema tem um **controller** próprio e a URL deixa explícito o **controller** a ser utilizado.



Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
painelCtrl.js	6710	js-file	02/19/17 18:...	-rw-rw-r--	ubuntu u...
homeCtrl.js	2276	js-file	02/17/17 17:...	-rw-rw-r--	ubuntu u...
editarCtrl.js	406	js-file	02/15/17 15:...	-rw-rw-r--	ubuntu u...
edificioCtrl.js	15738	js-file	02/23/17 18:...	-rw-rw-r--	ubuntu u...
consumoCtrl.js	4432	js-file	02/23/17 16:...	-rw-rw-r--	ubuntu u...
adminCtrl.js	4770	js-file	02/23/17 16:...	-rw-rw-r--	ubuntu u...

4.4 SERVICES

Os **services**, responsáveis pela conexão ao ServerSide (lado do servidor), são configurados para cada EndingPoint de forma explícita, e enviam parâmetros conforme a descrição da API.

Veja exemplo de acesso ao servidor via REST, utilizando um **service**:

```
"cbcs.service("edificioAPI", function(con g, $http){ this.getEdi cios = function (){  
    return $http.get(con g.baseUrlAPI+"api/edi cios");  
});"
```

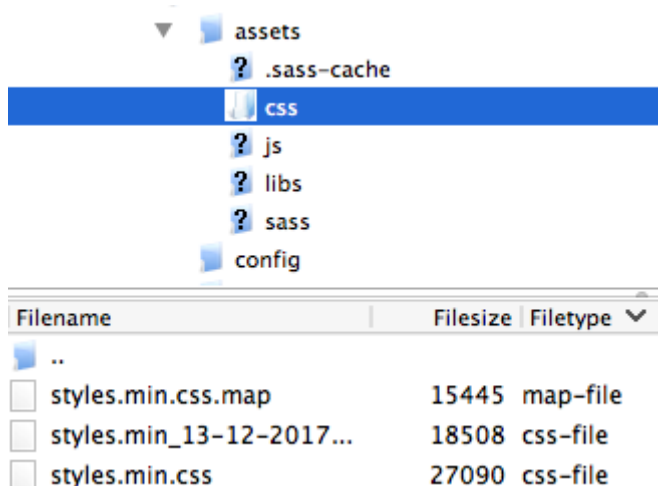
Nesse caso, o **service** devolverá um Escopo contendo dados dos edifícios cadastrados no sistema, permitindo o acesso aos dados no **controller**, possibilitando a livre manipulação na **View**.

O sistema utiliza um arquivo de configuração, onde as variáveis como "config.baseUrlAPI" são configuradas e podem ser utilizadas em todos os serviços. Este arquivo fica na pasta:

/config/configValue.js

4.5 FOLHA DE ESTILOS

O CSS (**cascade stylesheet system**) está localizado na pasta "assets/css" conforme figura abaixo, sendo utilizado em um formato minimizado. A versão completa do CSS pode ser encontrada na mesma pasta:



4.6 BOOTSTRAP

Ainda no quesito visual, foi utilizado o **bootstrap framework** do Twitter para agilizar o processo de desenvolvimento do FrontEnd, sendo empregada a versão 3 CDN.

4.7 BANCO DE DADOS

O banco utilizado é um banco de dados não relacional MONGO DB, o banco não relacional mais utilizado no mundo. A modelagem está explícita dentro da pasta apps/models, possuindo cada coleção de dados seu próprio arquivo .js.

Para a conexão foi utilizado o Mongoose e cada **model** com seu respectivo formato de dados como por exemplo:

```
"var mongoose = require('mongoose');
var Schema = mongoose.Schema;
module.exports = mongoose.model('Edificios', new Schema({
nome id_cidade      : String,
endereco           :String,
area_util ....
```

O MongoDB utiliza basicamente entradas e saídas de dados em formato **JSON** e todo o sistema também utiliza este formato em todos os scripts de manipulação de dados.

Exemplo de saída de dados após consulta ao REST por edifícios cadastrados:

```
[
{ "_id": "58af9f565858e521fc0a823d",
  "grupo_gerador": 1,
  "area_util_iluminada": 28,
  "area_acondicionada": 3333.33,
    "potencia_cpd": 0,
      "consumo_cpd": 240128,
  "numero_elevadores": null,
  "numero_ocupantes": 287,
  "sistema_ar_condicionado":
    "Chiller a água e fancoils", "numero_andares": 9, "area_estacionamento_coberto": 259,
      "area_util": 3333.33,
  "endereco": "Av Brigadeiro Faria Lima, 1793",
  "id_cidade": "58af6d94af0e31154050e133",
  "nome": "Edi cio novo",
  "__v": 0,
  ... "
```

4.8 RESTFUL API

Representational State Transfer (REST) ou RESTful Web Services são uma forma de fornecer interoperabilidade entre sistemas de computadores na Internet¹.

O Servidor foi todo construído para gerar um RESTful com todos os **endpoints** (url de acesso a dados do RESTful) necessários para o funcionamento completo do sistema.

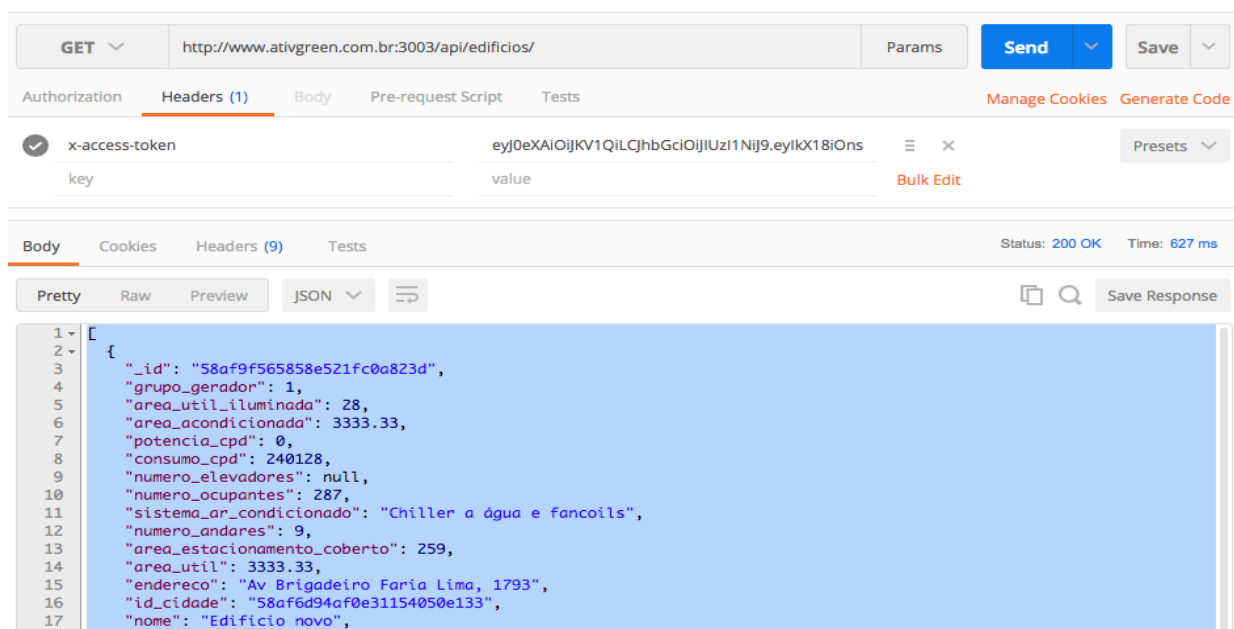
A segurança está na "Tokenização", geração de conjuntos de caracteres criptografados, que possuem validade de 2 horas.

A cada acesso feito pelo usuário, na versão final, o sistema valida no banco de dados do usuário e a senha fornecida para o **login**. Estando estas informações corretas, retorna com um conjunto de caracteres criptografados (**token**) que servirá de chave de acesso ao RESTfull. A partir daí o

¹ https://en.wikipedia.org/wiki/Representational_state_transfer

FrontEnd e o BackEnd passam a acessar os métodos de consulta, informando sempre esse **token**, que garantirá a veracidade da origem e a criptografia dos dados de acesso.

Na figura abaixo, demonstramos como realizar uma chamada ao RESTfull pelo POSTMAN (programa altamente difundido e conhecido para realizar chamadas via JSON) incluindo o **token**:



5. INSTALAÇÃO DO SISTEMA - PACOTES E DEPENDÊNCIAS

O servidor tem alguns pacotes necessários para seu funcionamento, além dos pré-requisitos de hardware e software já descritos nos capítulos anteriores. É necessária também a instalação dos pacotes especiais descritos abaixo, sendo que para facilitar sua instalação, sugerimos a utilização do NPM, que gerencia pacotes e pode ser utilizado via SSH.

Com o NPM instalado, utilize para a instalação o “package.json”, o qual está na raiz da plataforma.

REFERÊNCIAS DE PACOTES E PLUGINS NECESSÁRIOS

Os pacotes necessários e suas referências estão listados abaixo, devendo sempre ser utilizadas suas versões mais recentes.

SERVIDOR:

body-parser V. 1.15.0 - <https://www.npmjs.com/package/body-parser-json>

express V 4.13.4 - <http://expressjs.com/>

jsonwebtoken V ^5.7.0 - <https://jwt.io/introduction/>

mongoose V 4.4.5 - <http://mongoosejs.com/>

morgan V 1.7.0 <https://github.com/expressjs/morgan>

BANCO DE DADOS

Mongo Db - <https://www.mongodb.com/>

FRONTEND E BACKEND

ANGULAR JS - <https://angularjs.org/>

JQUERY - <https://jquery.com/>

MOMENT.JS - <https://momentjs.com/>

BOOTSTRAP - <http://getbootstrap.com/>

24

6. DEFINIÇÃO DE SIGLAS E TERMOS

As siglas utilizadas em todos os documentos são de controle universal pela comunidade mundial de desenvolvedores WEB, abaixo estão descritas as mais utilizadas:

CSS: **Cascading Style Sheets** (CSS) é uma linguagem de folhas de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML.

HTML: (abreviação para a expressão inglesa **HyperText Markup Language**, que significa Linguagem de Marcação de Hipertexto) é uma linguagem de marcação utilizada na construção de páginas na Web. Documentos HTML podem ser interpretados por navegadores. A tecnologia é fruto da junção entre os padrões HyTime e SGML.

RESTful: A **Representational State Transfer** (REST), em português Transferência de Estado Representacional. Pode-se caracterizar os **web services** como “RESTful” se eles estiverem em conformidade com as restrições descritas na seção restrições arquiteturais.

QUERY: Processo de extração de dados de um banco de dados e sua apresentação em uma forma adequada ao uso.

ROOT: O processo de **rooting** permite ao usuário:

- Remover, alterar ou substituir aplicativos de sistema ou pré-instalados de fábrica - liberando, desta forma, espaço na memória interna do dispositivo;
- Utilizar aplicativos especializados que requerem permissões administrativas
- Realizar operações de outra forma inacessíveis a um usuário comum.
- Substituição do sistema operacional original do dispositivo por outro.

SSH: **Secure Shell** (SSH) é um protocolo de rede criptográfico para operação de serviços de rede de forma segura sobre uma rede insegura. O exemplo de aplicação mais conhecido é para **login** remoto a sistemas de computadores pelos usuários.

PING: Ping ou latência como podemos chamar, é um utilitário que usa o protocolo ICMP para testar a conectividade entre equipamentos. É um comando disponível praticamente em todos os sistemas operacionais. Seu funcionamento consiste no envio de pacotes para o **equipamento de destino e na “escuta” das respostas. Se o equipamento de destino estiver ativo, uma “resposta” (o “pong”, uma analogia ao famoso jogo de ping-pong) é devolvida ao computador solicitante.**

FRONTEND: O **frontend** é responsável por coletar a entrada do usuário em várias formas e processá-la para adequá-la a uma especificação em que o **backend** possa utilizar.

Comitê Temático de Energia – CBCS

energia.benchmarking@cbcs.org.br



PROJETO 3E

EFICIÊNCIA ENERGÉTICA EM EDIFICAÇÕES

WWW.MMA.GOV.BR/CLIMA/ENERGIA/PROJETOS

Coordenação:

**MINISTÉRIO DO
MEIO AMBIENTE**



Implementação:

Financiamento:



*Empoderando vidas.
Fortalecendo nações.*



Realização:

DEO | DESEMPENHO
ENERGÉTICO
OPERACIONAL
EM EDIFICAÇÕES


CBCS
Conselho Brasileiro de
Construção Sustentável