

## Alunos

Alison Pereira Ribeiro  
Frank Douglas Barros  
Cleiton Solano Soares Caetano  
Gabriel Soares Costa

**Universidade Federal de Goiás**  
**Prof: Fábio Moreira Costa**  
**Sistemas Distribuídos**  
**Documentação - Projeto de Implementação**

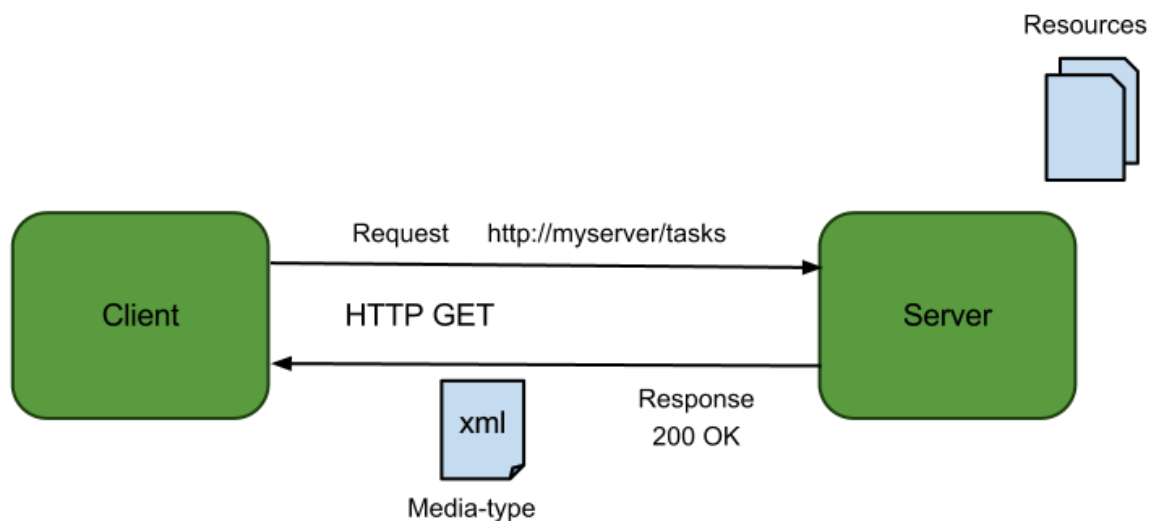
Front-end: <http://ec2-52-200-1-213.compute-1.amazonaws.com/>

Github Servidores: <https://github.com/gabrielsxp/project-building-api>

Github Front-End: <https://github.com/gabrielsxp/building-complex-fe>

## Arquitetura

A arquitetura do projeto consiste na em Restful. Vários clientes realizam várias requisições HTTP, que no caso do projeto podem ser realizadas por meio do Postman, ou da aplicação front-end escrita para o projeto.



O servidor foi escrito em Javascript por meio do Node JS. A biblioteca Express foi utilizada para realizar as requisições HTTP e o MongoDB foi a base de dados utilizada. Outra biblioteca que foi utilizada foi o Mongoose, que foi responsável por criar modelos fixos para que exista uma consistência de dados, uma vez que o MongoDB é um banco de dados especializado no armazenamento de documentos.

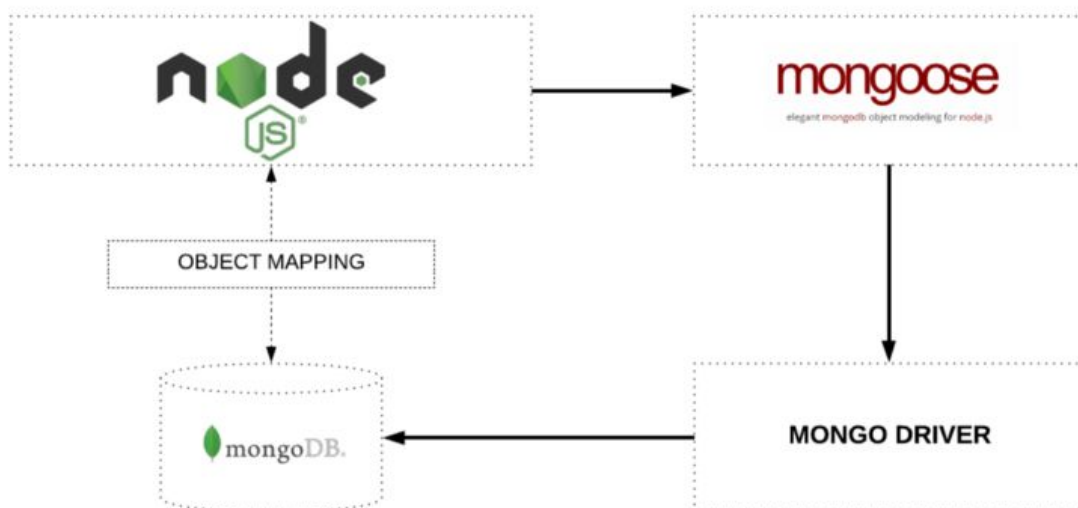
Não houve uma divisão sistemática do conjunto de ações possíveis em máquinas diferentes, ou seja, cada máquina é responsável por todas as rotas destinadas ao sistema como um todo. Para que não houvesse uma sobrecarga de requisições em uma máquina só, e consequentemente uma fila de requisições, foi implementado um round-robin. A intenção é a de que exista uma máquina servidora responsável pelo retorno desta lista de servidores antes que a requisição ao complexo seja realizada.

Essa estratégia foi implementada pois, ao implementarmos o desacoplamento das atividades, ou seja, um servidor responsável pela gerência de um prédio e vários outros servidores responsáveis pelas requisições de um andar distinto, dois problemas surgiram. O primeiro foi o tratamento de erros. O segundo foi o overhead nas respostas das requisições.

Por se tratar de uma arquitetura REST, nessa implementação desacoplada, foram necessários 3 callbacks para que o usuário fosse capaz de conseguir entrar no prédio. A primeira vista não parecia muito problemático, porém, a partir do momento que várias requisições foram feitas, o tempo necessário para que a resposta chegasse ao cliente foi ficando cada vez maior. Quanto ao tratamento de erros, foi extremamente complicado gerenciar os erros que foram surgindo a partir da primeira requisição, pois não eram detectados imediatamente e a requisição propagava para a próxima instância sem o tratamento devido. Portanto, por essas questões optamos por continuar com cada servidor responsável por todo o complexo como um todo e replicar os bancos.

## Banco de Dados

Baseando-se na descrição do projeto, os modelos salvos no banco de dados são Usuários, Andares e Prédios. Modelos são basicamente entidades, que possuem tipos e métodos específicos que podem ser utilizados uma vez que são instanciados. Esse esquema permite que o documento que será salvo no banco de dados seja especificado como um objeto.



## Replicação do Banco de dados

O banco de dados foi dividido em um conjunto de shards (pedaços). Para que a replicação funcionasse corretamente, foi necessário que cada instância do mongodbg fosse criada em máquinas diferentes. Desse modo, uma máquina teve o papel de coordenadora e as outras máquinas foram escravas.

O princípio de consistência dos dados que foi aplicada foi assíncrona, isto é, existe um período de inconsistência entre as réplicas. Foi constatado ainda que a consistência passou a ser eventual após um certo intervalo de tempo. Portanto, não foi possível estabelecer uma consistência total dos dados imediatamente após a inserção dos mesmos. Nos testes realizados, após uma requisição de inserção de prédios ser feita, a máquina coordenadora foi desligada e assim, os prédios que tinham sido inseridos antes da queda não persistiram para a réplica. Apenas os prédios que tinham inseridos a algum tempo permaneceram.

Inicialmente, foi implementada a replicação síncrona, ou seja, a cada modificação na réplica coordenadora, todas as modificações eram propagadas para todas as demais. Porém, o tempo de espera era proporcional a n vezes a quantidade de réplicas dos bancos, ou seja, como usamos 5 réplicas para o banco de dados, o tempo de espera para cada requisição era 5 vezes maior. Por isso, optamos pela consistência eventual nesse caso.

## **Rotas**

O Express cria endpoints que apontam para as requisições recebidas pelo loop principal do express na porta especificada. Cada endpoint é responsável por realizar ações sobre o sistema como um todo. Um exemplo é a criação de um prédio do complexo. Existe um endpoint '/building' que atende por uma requisição HTTP POST e espera que no corpo da requisição, existam dados no formato JSON que permitam que o modelo instanciado do Mongoose seja salvo no banco de dados. Decidimos não criar Controllers para as rotas individualmente, uma vez que existem certas nuances que são tratadas individualmente dependendo da rota em que a requisição é realizada.

## **Desempenho com apenas um servidor (10 requisições simultâneas - Cria fila de requisições)**

Tempos medidos através da ferramenta de desenvolvedor da Google

### **Lista de todas os prédios (Inclui a quantidade de pessoas que podem entrar no complexo)**

263	446	389	307	517	324	297	649	472	249
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

em ms

### **10 usuários acessando o complexo simultaneamente (Inclui a quantidade de pessoas que podem entrar no complexo)**

29	74	79	37	59	84	46	71	59	43
----	----	----	----	----	----	----	----	----	----

em ms

## **Desempenho com servidores desacoplados (10 requisições simultâneas - Cria fila de requisições)**

Tempos medidos através da ferramenta de desenvolvedor da Google

### **Lista de todas os prédios (Inclui a quantidade de pessoas que podem entrar no complexo)**

521	598	604	741	547	814	466	769	652	501
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

em ms

### **10 usuários acessando o complexo simultaneamente (Inclui a quantidade de pessoas que podem entrar no complexo)**

46	24	79	64	87	42	57	95	76	64
----	----	----	----	----	----	----	----	----	----

em ms

## **Desempenho com o uso de Round-Robin (10 requisições simultâneas)**

Tempos medidos através da ferramenta de desenvolvedor da Google

### **Lista de todas os prédios (Inclui a quantidade de pessoas que podem entrar no complexo)**

242	246	260	296	177	161	141	235	264	182
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

em ms

### **10 usuários acessando o complexo simultaneamente (Inclui a quantidade de pessoas que podem entrar no complexo)**

32	28	34	47	63	31	74	27	36	39
----	----	----	----	----	----	----	----	----	----

em ms

## **Instalação**

A instalação consiste basicamente na instalação de pacotes padrão provenientes do Github. O Node JS (<https://nodejs.org/en/>) e o NPM (<https://www.npmjs.com/>) são necessários para que a instalação de ambos os pacotes (servidor e front-end) sejam instalados.

Lembrando que esta versão está configurada para acessar o banco de dados localmente (127.0.0.1:27017)

O banco de dados também deve estar instalado de forma padrão, assim como foi através desse link (<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/>).

Passos para instalar o servidor (<https://github.com/gabrielsxp/project-building-api>)

1. Clone o repositório via terminal.
2. Acesse a pasta de destino via `cd project-building-api`
3. Execute `sudo npm install`
4. \* Execute `sudo npm start`

```
PS C:\app\Node\building-complex-api> npm start
> building-complex-api@1.0.0 start C:\app\Node\building-complex-api
> node src/index.js
mongodb://127.0.0.1:27017/building-complex-api
Rodando na porta: 3001
```

\* Esta etapa funcionará apenas se o serviço do MongoDB estiver em execução

```
sudo service mongod start
```

copy

Passos para instalar o cliente (<https://github.com/gabrielsxp/building-complex-fe>) são idênticos ao servidor acima.

Uma vez instalados, basta abrir o cliente pelo navegador em (<http://127.0.0.1:3000/>).

```
You can now view building-complex in the browser.

Local: http://localhost:3000/
On Your Network: http://192.168.25.30:3000/

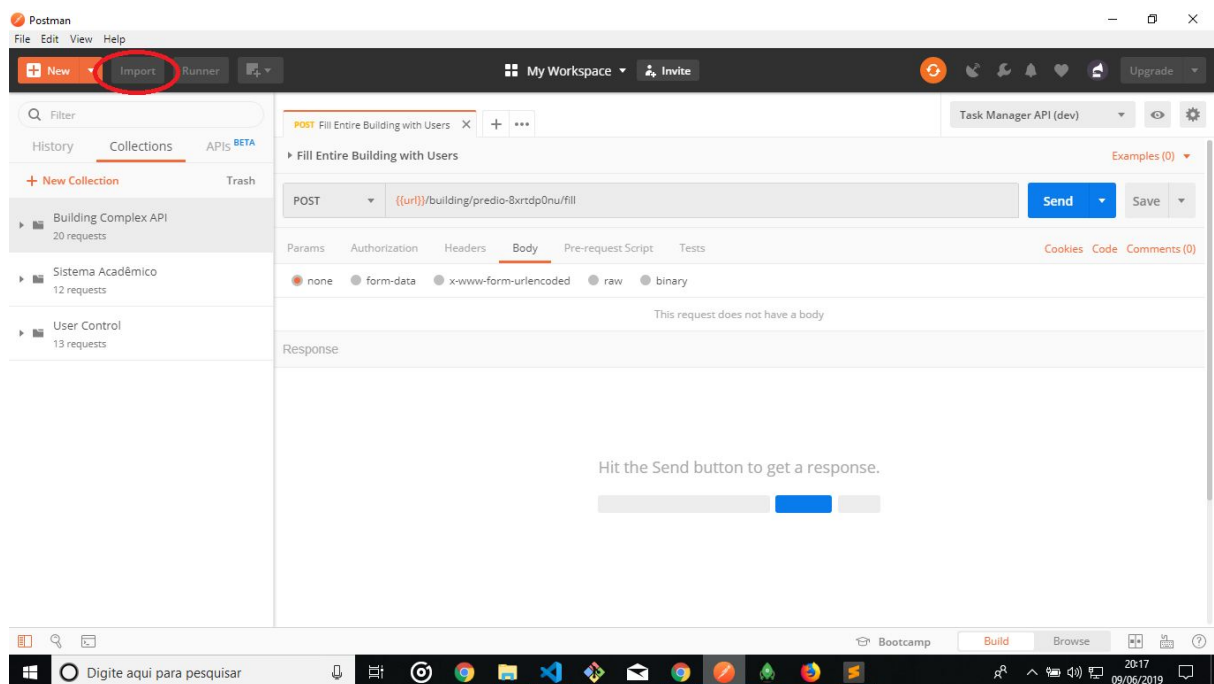
Note that the development build is not optimized.
To create a production build, use npm run build.
```

Acesse a sessão de testes para navegar sobre o conteúdo da aplicação.

## Postman

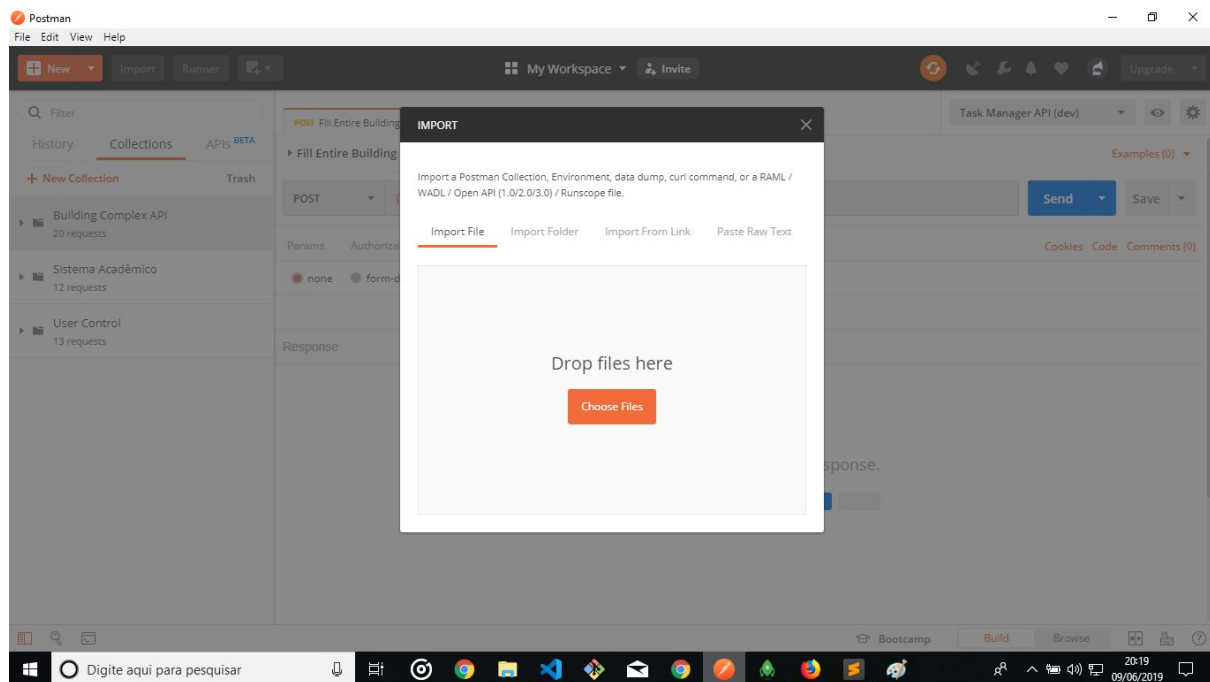
No pacote .zip, existe uma coleção do Postman com rotas testadas para realizar as requisições. Basta importar o arquivo e salvar a coleção.

### Clique em Import

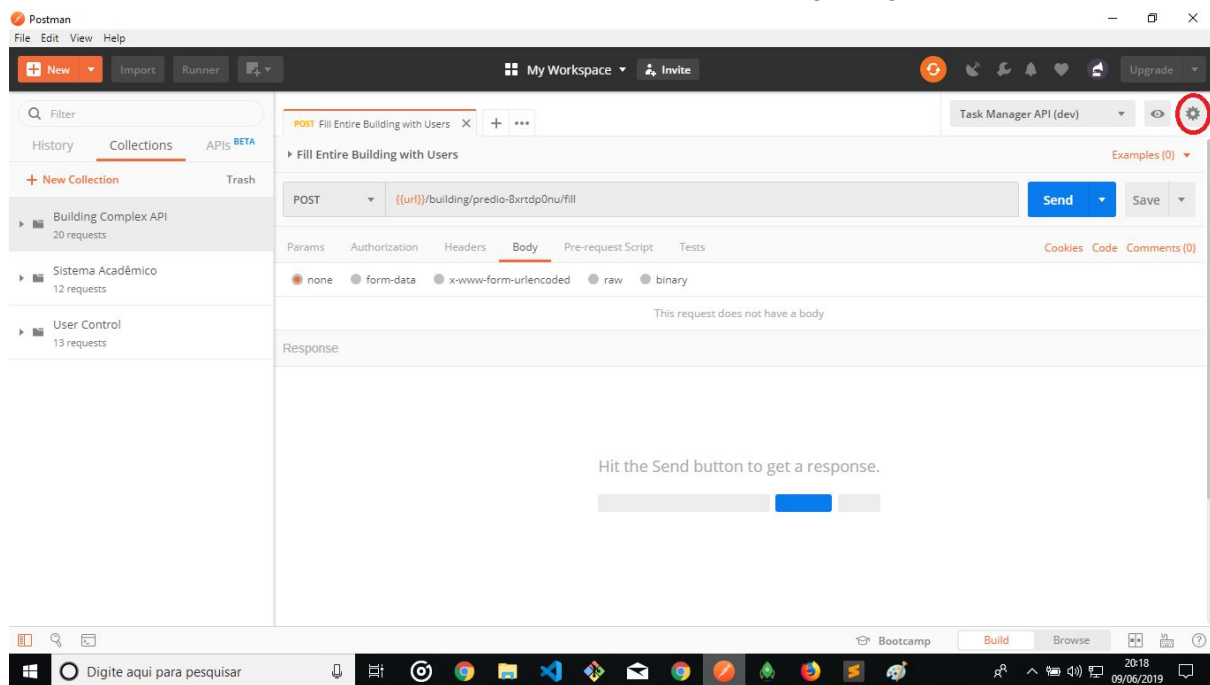


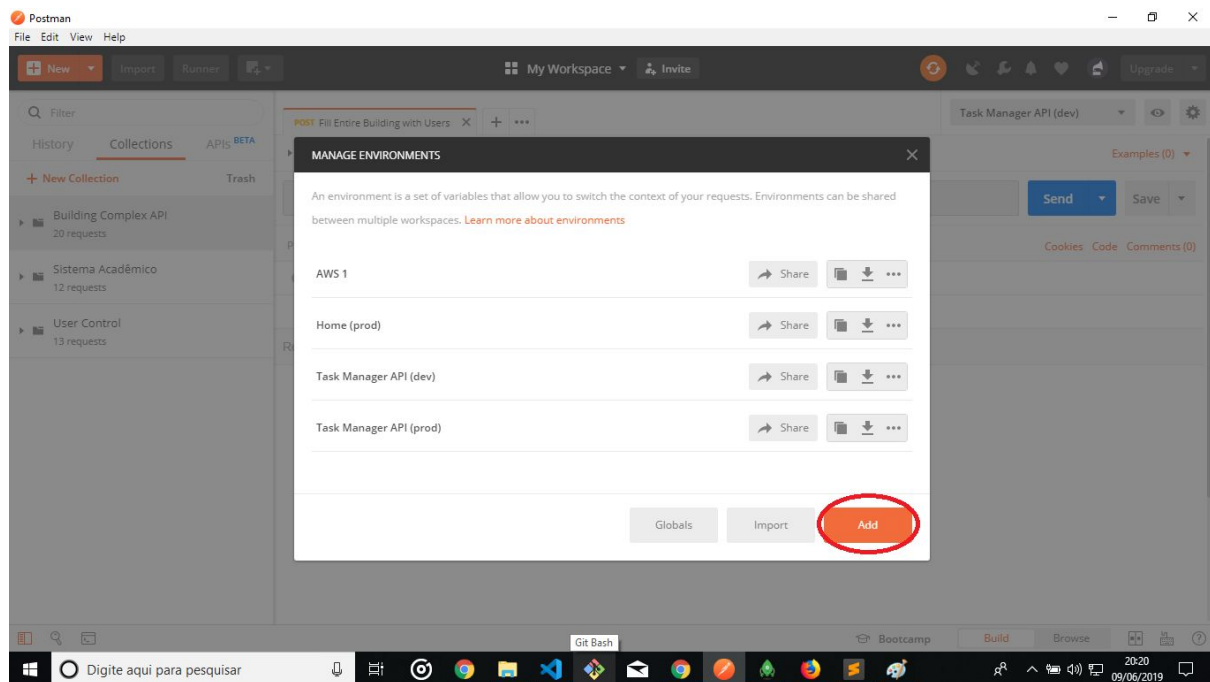
Escolha o arquivo e salve a coleção.

Obs: As requisições em lista circular não funcionam no Postman, apenas na aplicação front-end

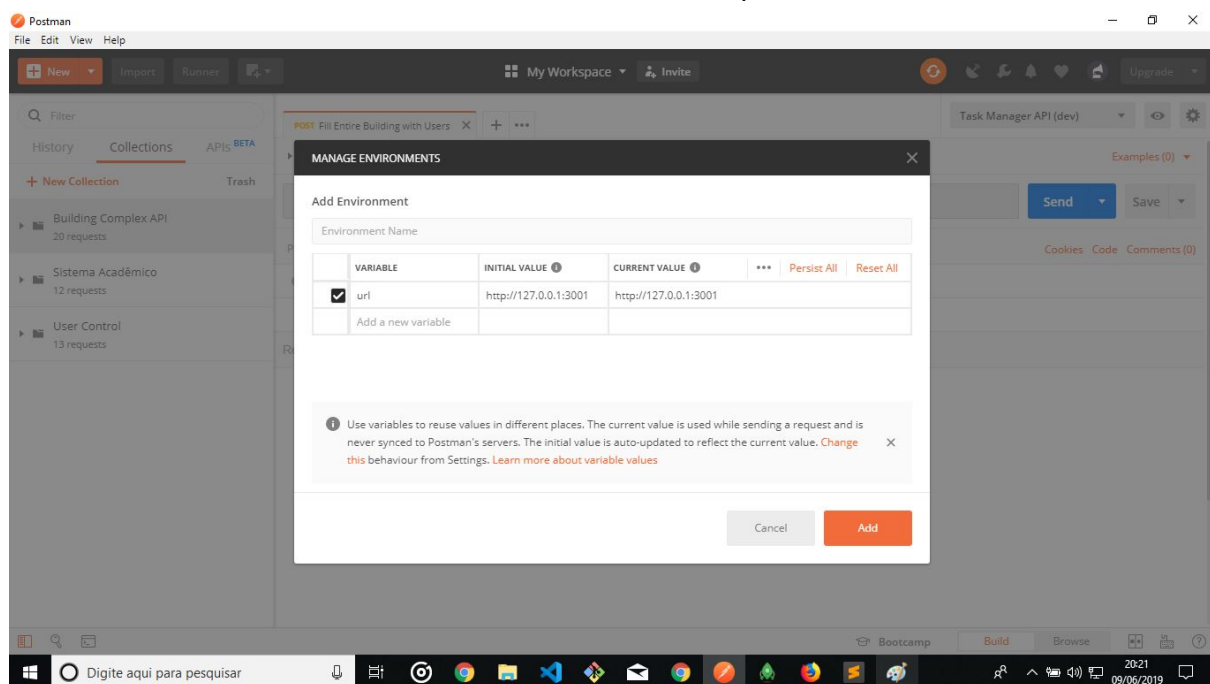


Depois crie uma variável chamada url. Para isso, clique na engrenagem.





Preencha **VARIABLE** com url e **INITIAL VALUE** com `http://127.0.0.1:3001`

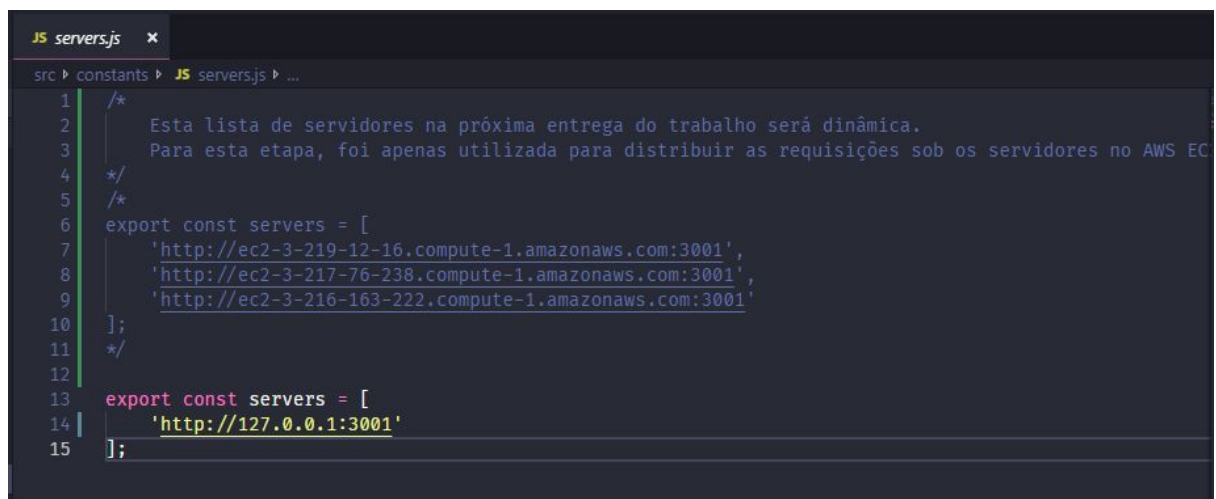




**Escolha uma rota e faça a requisição**

## Funcionamento

Obs: O sistema como um todo está configurado para execução em máquina local, para que a reprodução do mesmo seja realizada nas máquinas do AWS, basta informar o grupo para ligarmos as máquinas virtuais e utilizar a lista de servidores que está em “src/constants/servers.js”.

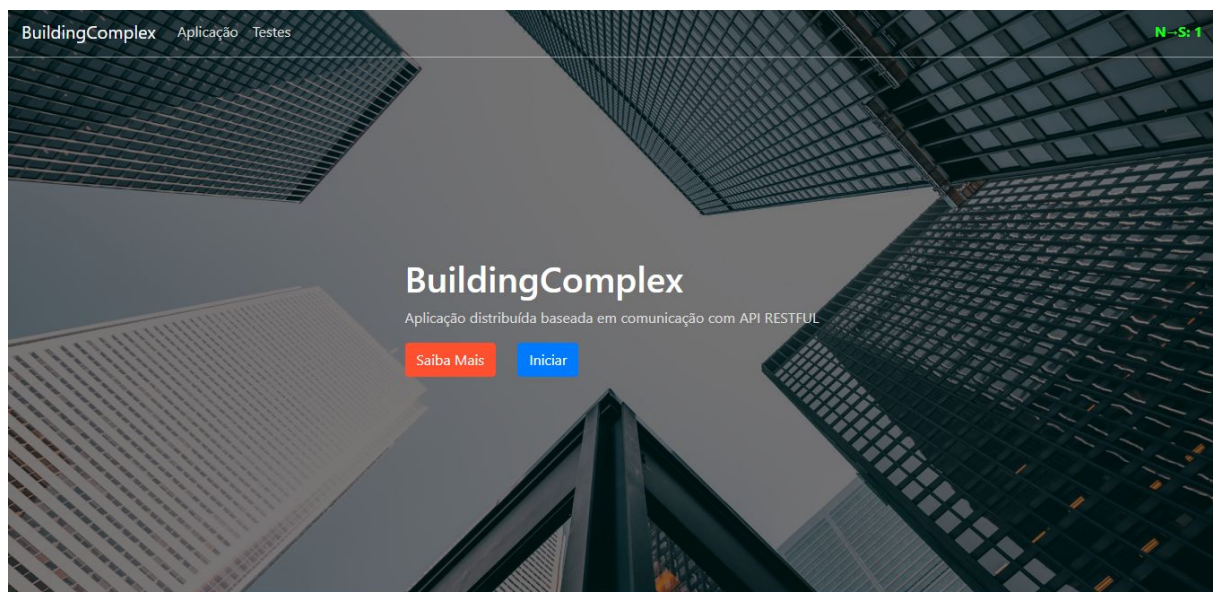


```
JS servers.js x
src ▾ constants ▾ JS servers.js ▾ ...
1  /*
2     Esta lista de servidores na próxima entrega do trabalho será dinâmica.
3     Para esta etapa, foi apenas utilizada para distribuir as requisições sob os servidores no AWS EC
4  */
5  /*
6  export const servers = [
7      'http://ec2-3-219-12-16.compute-1.amazonaws.com:3001',
8      'http://ec2-3-217-76-238.compute-1.amazonaws.com:3001',
9      'http://ec2-3-216-163-222.compute-1.amazonaws.com:3001'
10 ];
11 */
12
13 export const servers = [
14     'http://127.0.0.1:3001'
15 ];
```

## Desta forma

```
JS servers.js
src ▸ constants ▸ JS servers.js ▸ ...
1  /*
2     Esta lista de servidores na próxima entrega do trabalho será dinâmica.
3     Para esta etapa, foi apenas utilizada para distribuir as requisições sob os servidores no AWS EC
4  */
5
6  export const servers = [
7    'http://ec2-3-219-12-16.compute-1.amazonaws.com:3001',
8    'http://ec2-3-217-76-238.compute-1.amazonaws.com:3001',
9    'http://ec2-3-216-163-222.compute-1.amazonaws.com:3001'
10 ];
11
12
13 // export const servers = [
14 //   'http://127.0.0.1:3001'
15 // ];
```

Após a instalação do cliente, uma tela igual a esta aparecerá no navegador padrão.



Para iniciar, acesse a sessão de testes clicando em Testes na barra de navegação.

## Sessão de Testes

### Adicionar Prédios

**Descrição:** Adicione uma lista de prédios, definindo a quantidade de prédios e a quantidade máxima de andares de cada um

[Opções](#)

### Remover tudo

**Descrição:** Remover todos os prédios do complexo cadastrados até agora, juntamente com os andares e todos os usuários do sistema

[Excluir tudo](#)

### Simular usuários

**Descrição:** Simula a entrada de vários usuários no prédio de uma só vez

[Opções](#)

### Expulsar o usuário atual do prédio

**Descrição:** Remover o usuário que seja **visitante** do prédio atual.

[Expulsar Usuário](#)

### Lotar um edifício

### Modo Administrador

## Adicione os prédios e defina as permissões desejadas

BuildingComplex Aplicação Testes N→S: 1

### Opções para adicionar prédios

Quantidade de prédios  
10

Quantidade de andares  
10

Número máximo de andares que serão escolhidos através de uma função randômica

Permissões de cada andar

- ☒ Visitante
- ☒ Funcionário
- ☒ Administrador

[Acessar](#)

[Fechar](#)

Uma lista de prédios com nomes aleatórios será gerada de acordo com as definições estabelecidas na página anterior.



No momento que decidir entrar em algum edifício, o sistema de autenticação é extremamente simples. O usuário define qual tipo de nível de acesso ele possui e digita os dados necessários para autenticá-lo. Caso seja visitante, foi acordado que o sistema armazenará seus dados apenas enquanto o mesmo estiver dentro de algum edifício. A partir do momento que o usuário decidir sair do prédio, seus dados serão permanentemente apagados. Porém, se o usuário for um funcionário, suas credenciais serão armazenadas enquanto a função de apagar todos os dados seja chamada.

Portanto, para entrar em algum prédio do complexo, basta digitar as credenciais. Foi definido também, por questão de testes, que os funcionários serão criados no momento que o usuário digitar suas credenciais. O ideal seria criá-los a partir de uma função definida pelo administrador, porém por questão de praticidade e de testes, foi definido este comportamento.

## Acessar Edifício

Selecione o seu nível de acesso

Visitante ▼

Endereço de Email

Informe seu endereço de email

Nós nunca divulgaremos o seu endereço de email para ninguém

Nome

Informe seu nome completo

Acessar

Ao acessar um determinado edifício, o usuário estará no hall do prédio, que possui uma capacidade pré-definida (a mesma dos andares). É possível que o usuário veja as permissões e a capacidade de cada andar antes que o mesmo consiga acessá-lo, além da quantidade de pessoas que estão no andar no momento e a capacidade total do edifício.

### Nome do Prédio: Predio-avq64pv4d

Existem atualmente 1 pessoas nesse prédio

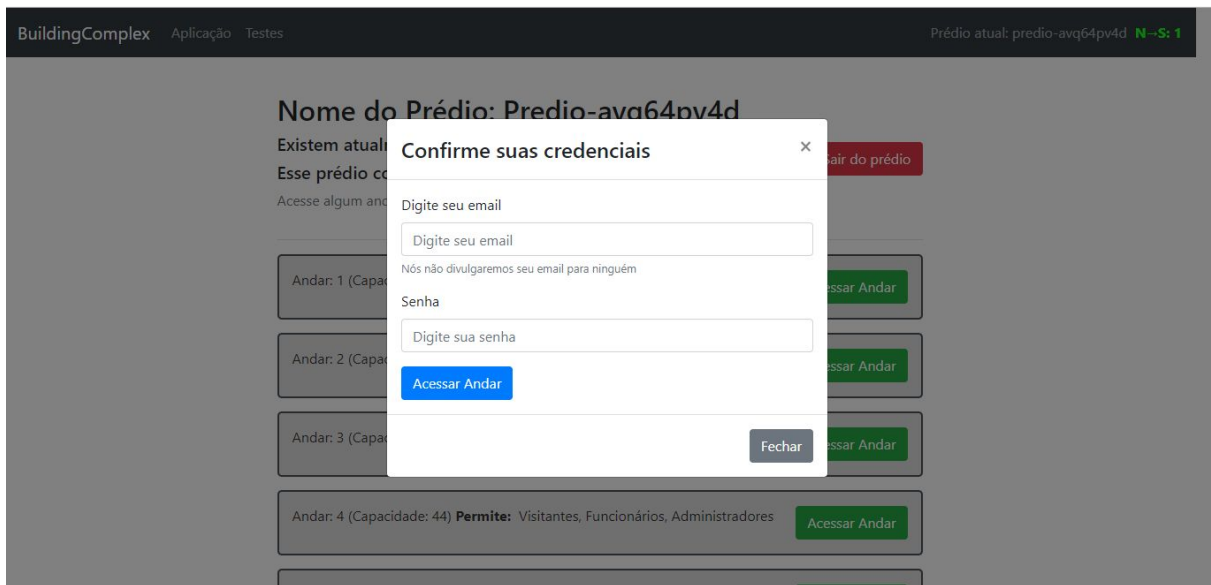
Esse prédio comporta 440 pessoas

Acesse algum andar do prédio

Sair do prédio

Andar: 1 (Capacidade: 44) <b>Permite:</b> Visitantes, Funcionários, Administradores	Acessar Andar
Andar: 2 (Capacidade: 44) <b>Permite:</b> Visitantes, Funcionários, Administradores	Acessar Andar
Andar: 3 (Capacidade: 44) <b>Permite:</b> Visitantes, Funcionários, Administradores	Acessar Andar
Andar: 4 (Capacidade: 44) <b>Permite:</b> Visitantes, Funcionários, Administradores	Acessar Andar

Ao tentar acessar algum andar, as credenciais de acordo com seu nível de acesso serão requisitadas, independentemente do seu nível de acesso e do andar que o mesmo deseje acessar.



Após acessar o andar requisitado, o usuário poderá ver o andar que o mesmo se encontra, a capacidade e permissões de cada andar e a quantidade de pessoas que estão ali no momento.



Caso o usuário esteja acessando como funcionário, uma opção no painel de testes está disponível para torná-lo administrador.

<div>Opções</div>	<div>Excluir tudo</div>
<div><div>Simular usuários</div><div>Descrição: Simula a entrada de vários usuários no prédio de uma só vez</div><div>Opções</div></div>	<div><div>Expulsar o usuário atual do prédio</div><div>Descrição: Remover o usuário que seja <b>visitante</b> do prédio atual.</div><div>Expulsar Usuário</div></div>
<div><div>Lotar um edifício</div><div>Descrição: Adicionar pessoas aleatórias no edifício inteiro, tanto no hall quanto em todos os andares. Impossibilitando a entrada de novas pessoas</div><div>Opções</div></div>	<div><div>Modo Administrador</div><div>Descrição: Atualiza permissões para Administrador ao usuário atual do sistema que seja Funcionário</div><div>Atualizar</div></div>

## Sessão de Testes

Permissões atualizadas

Voltar para o Complexo

<div><div>Adicionar Prédios</div><div>Descrição: Adicione uma lista de prédios, definindo a quantidade de prédios e a quantidade máxima de andares de cada um</div><div>Opções</div></div>	<div><div>Remover tudo</div><div>Descrição: Remover todos os prédios do complexo cadastrados até agora, juntamente com os andares e todos os usuários do sistema</div><div>Excluir tudo</div></div>
<div><div>Simular usuários</div><div>Descrição: Simula a entrada de vários usuários no prédio de uma só vez</div><div>Opções</div></div>	<div><div>Expulsar o usuário atual do prédio</div><div>Descrição: Remover o usuário que seja <b>visitante</b> do prédio atual.</div><div>Expulsar Usuário</div></div>

Após atualizar as permissões do usuário atual da sessão, é possível alterar as permissões e a capacidade de qualquer andar que o mesmo entrar.

BuildingComplex

Aplicação

Testes

Prédio atual: predio-avq64pv4d **N-S: 1**

Andar Acessado

Você está atualmente no 1º andar

Capacidade: **44** pessoas

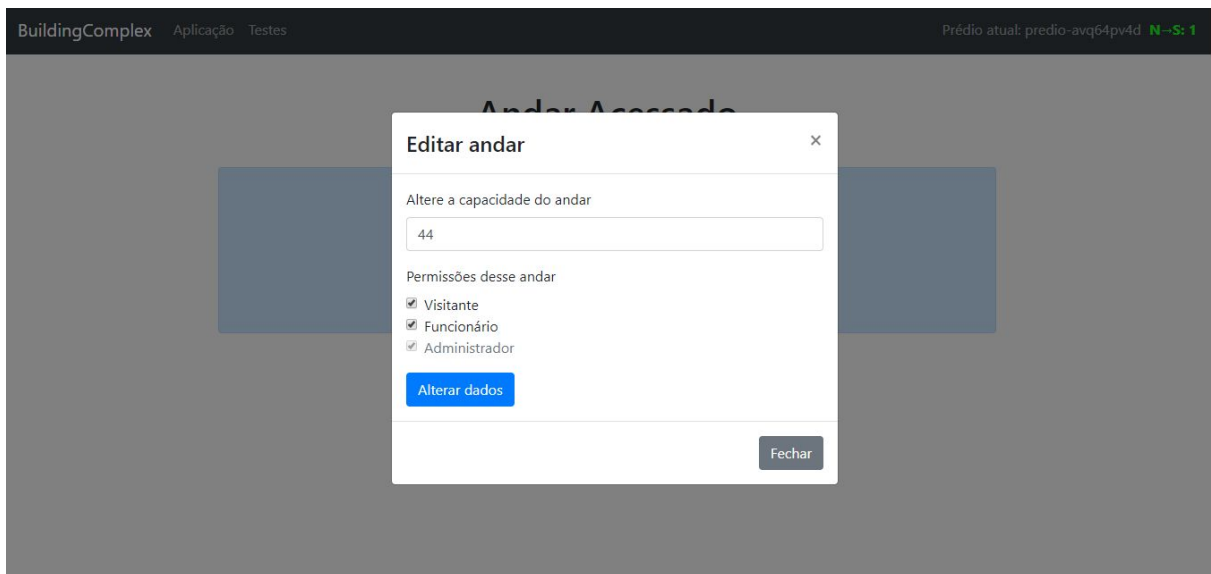
Existem **1** pessoas neste andar

Permite: Visitantes, Funcionários, Administradores

Voltar para o Hall

Sair do prédio

Editar Andar



**Após as alterações das capacidades, tanto a capacidade do edifício atual quanto do complexo serão atualizadas também.**