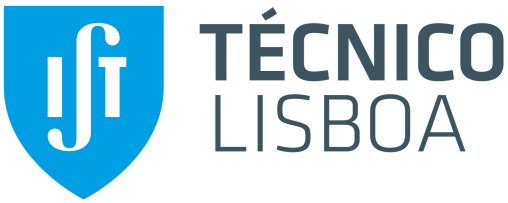

Arquitetura de Sistemas de Internet

Projeto

Diogo Ferreira 76090
Vera Pedras 75724

January, 2017

1 Introdução

Neste projeto é desenvolvido uma aplicação web para controlar a ocupação das salas numa organização como o Instituto Superior Técnico (IST). Nesta universidade, algumas salas encontram-se disponíveis para o uso dos estudantes. De maneira a perceber a ocupação das salas disponíveis é necessário que os alunos procedam a um registo nas salas.

O projeto desenvolvido oferece duas maneiras diferentes de operação. Onde o utilizador pode ser o administrador das salas ou o estudante que procuram utilizar as mesmas.

O administrador pode colocar disponiveis salas que se encontrem no sistema *FENIX* e ver a ocupação das mesmas. Os estudantes podem ver as salas disponíveis e entrar ou sair das mesmas, para além disso, podem também ver que estudantes se encontram em determinada sala e procurar a localização de um colega numa sala.

2 Arquitetura

O servidor da aplicação é implementado no Google Cloud Platform através do Google App Engine. No Google Platform é realizado o armazenamento de dados a partir do Google DataStorage. Este servidor deve receber pedidos HTTP vindos de um Browser através de código JavaScript. O cliente, que não é implementado neste projeto, realiza pedidos ao servidor. Sendo assim, este deve estar preparado para receber pedidos de um cliente, independetemente do código da API por este utilizado (Java, Python, etc...) Ao responder a pedidos HTTP o servidor pode chamar sobre si mesmo REST. O servidor realiza também pedidos a API do FenixEdu. O desenho desta arquitetura pode ser visto na Figura 1.

3 REST API

Para que seja REST (REpresentational State Transfer) necessita de ser:

- Uniform Interface;
  - Resource-Based, recursos são identificados por pedidos usando URIs como identificadores;

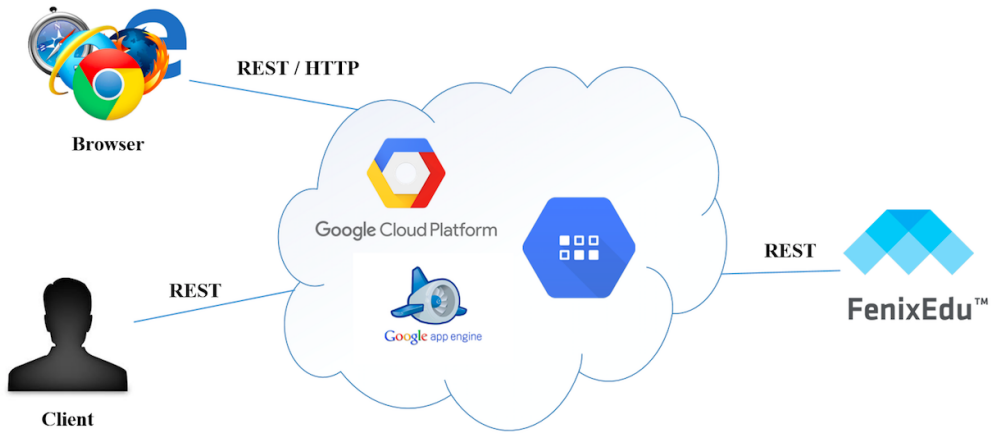

Figure 1: Arquitetura

  - Manipulation of Resources Through Representations, o cliente quando tem uma representação de um recurso, tem informação suficiente para poder alterar o mesmo, modificando-o ou apagando-o, caso tenha permissoes para isso;
  - Self-descriptive Messages, cada mensagem inclui informção suficiente que descreve a mesma;
  - Hypermedia as the Engine of Application State (HATEOAS).

- Stateless, o estado necessário para realizar o pedidos é contido no próprio pedido como URI, query-string parameters, body, ou headers;
- Cacheable;
- Client-Server, a uniform interface separa os clients dos servidores. Clients não se preocupam com a data storage, que continua interna a cada servidor, assim a portabilidade do código do cliente é melhorada;
- Layered System, um client não pode conectar-se diretamente ao Data Store;
- Code on Demand (opcional).

Para a *API REST* utiliza-se o Bottle que é uma solução rápida, simples e leve de WSGI micro web-framework para Python. Esta realiza o routing para os

diferentes URLs. Os recursos, neste projecto correspondem aos utilizadores, às salas e aos check in/out. Para estes recursos são usados os métodos GET, PUT e POST, sendo que o último apenas é utilizado para os utilizadores. O servidor envia a informação da base de dados em JSON ou HTML, conforme o pedido do cliente.

4 Tecnologias e Bibliotecas

Neste projecto utiliza-se no código do utilizador tecnologias como o Javascript, HTML e o AJAX (Asynchronous JavaScript And XML) que consiste numa combinação entre XMLHttpRequest object, JavaScript e HTML DOM.

Utiliza-se Javascript para escrever funções incluídas no HTML, devido a este correr no Browser do utilizador e não no servidor remoto, o navegador pode responde mais rapidamente. Além disso, o código JavaScript pode detectar ações do utilizador que o HTML sozinho não pode. O AJAX usa-se para realizar os pedidos HTTP dentro do código de Javascript.

Utiliza-se também Google Datastore NDB Client Library que permite App Engine Python apps ligar-se ao Cloud Datastore. Neste foram criadas as classes apresentadas em baixo.

Listing 1: Classes

```
1 class Space(ndb.Model):
2     room_id = ndb.StringProperty()
3     room_name = ndb.StringProperty()
4
5 class User(ndb.Model):
6     user_username = ndb.StringProperty()
7     user_id = ndb.StringProperty()
8     user_room = ndb.StringProperty()
9
10 class Check(ndb.Model):
11     user_id = ndb.StringProperty()
12     type = ndb.StringProperty()
13     room = ndb.StringProperty()
14     time = ndb.DateTimeProperty(auto_now_add=True)
```

Para além dos nomes de utilizadores e das salas, considera-se os id destas como Strings devido ao seu tamanho.

5 Interface do Sistema

O deploy do projecto é feito para o seguinte url: `https://asint-151622.appspot.com/`. Para um utilizador se registar utiliza a extensão `/html/signup`, onde aparece a página apresentada na Figura 2.

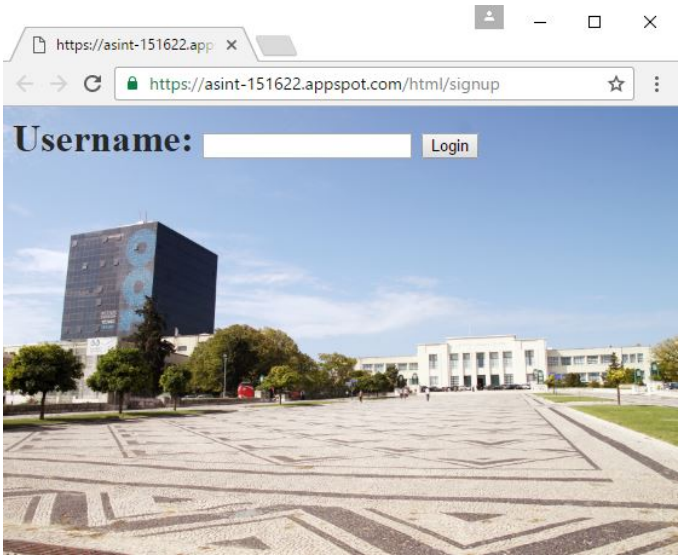

Figure 2: Página de login do utilizador

De seguida o utitizador insere o seu nome e faz o login aparecendo a página da Figura 3 presente na extensão `/html/login` , onde lhe é atribuido um ID.

De seguida o utilizador acede ao menu, como se demonstra na Figura 4, na extensão `/html/<id>`, em que <id> representa o seu id.

De seguida o utilizador pode fazer a procura de um amigo de modo a ver em que sala está, indo para Search Friend, onde lhe aparece na extensão `/html/<id>/searchfriend` a página da Figura 5.

Ao procurar um amigo aparece a sala onde esse está como se apresenta na Figura 6.

No menu o utilizador pode também ver as salas disponiveis que estarão na extensão `/html/<id>/rooms` como se apresenta na Figura 7.

Ao aceder a uma sala, o utilizador pode fazer check in se não estiver na mesma ou check caso esteja.Na Figura 8 apresenta-se um exemplo de Check

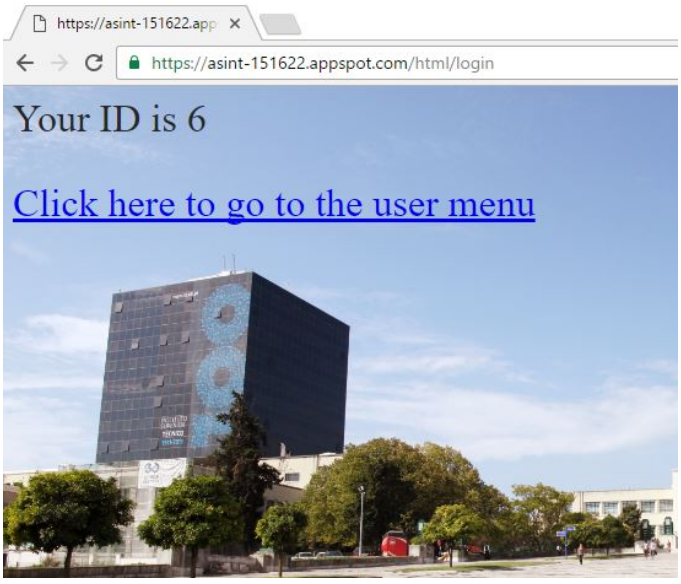

Figure 3: Página inicial do utilizador

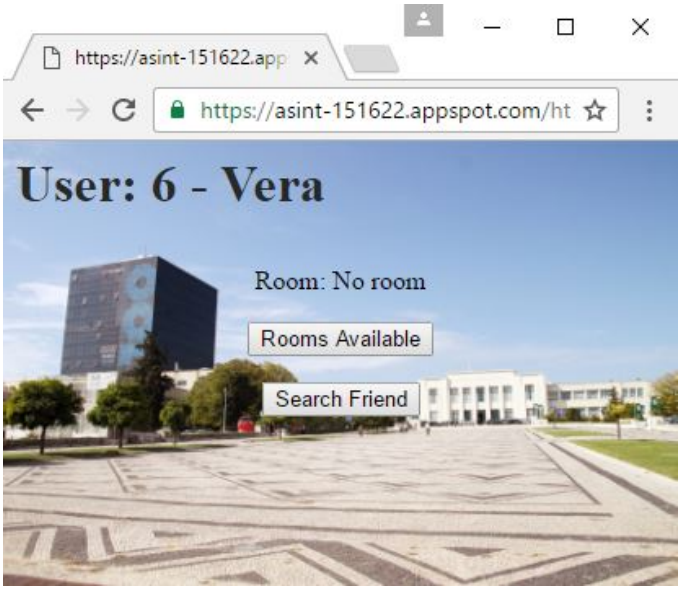

Figure 4: Página do menu do utilizador

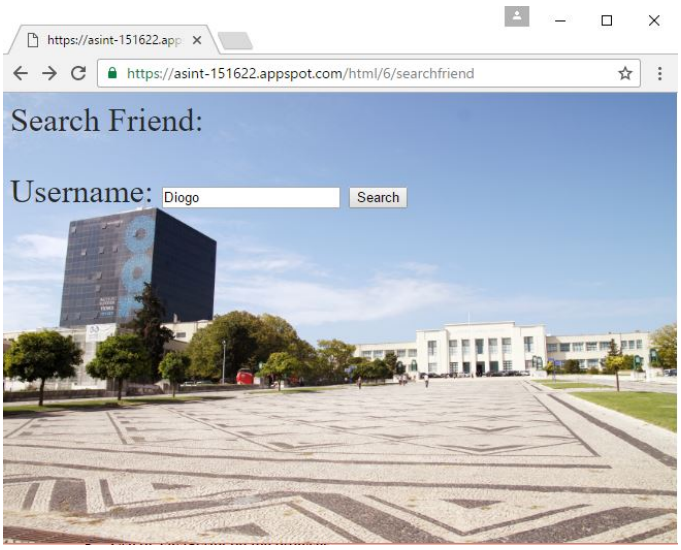

Figure 5: Página de Search Friend

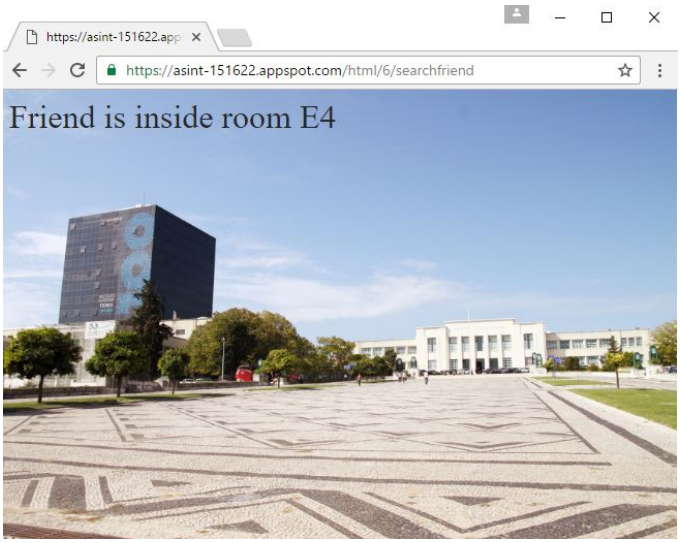

Figure 6: Página de indicação de onde está o amigo

in e na Figura 9 apresenta-se um exemplo de Check out. De notar que o utilizador também pode ver os outros utilizadores que estão nessa sala.

Na extensão `/html/admin` é possível aceder ao menu do administrador onde pode ver escolher entre adicionar salas ou ver a ocupação das salas já

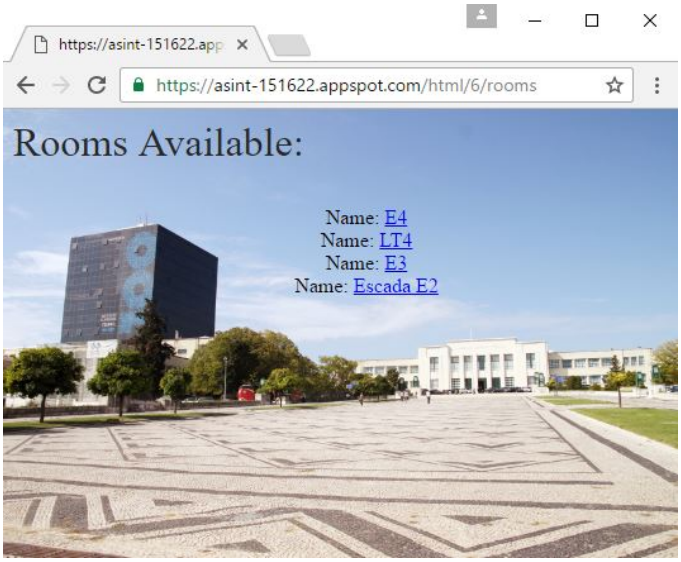

Figure 7: Página de salas disponíveis

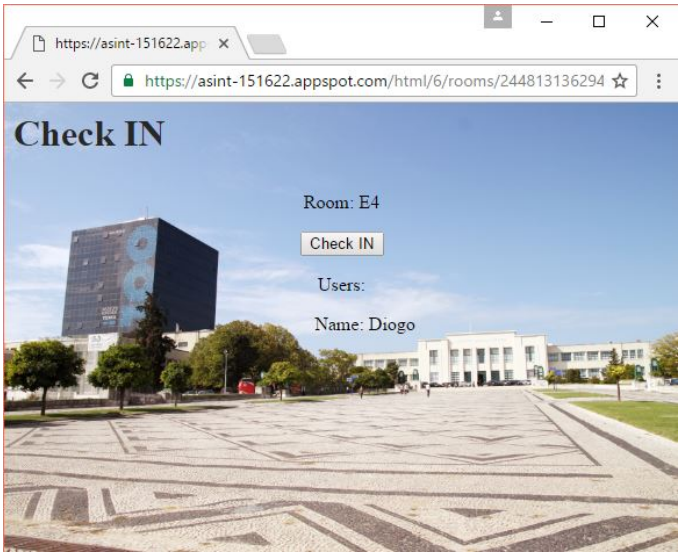

Figure 8: Página de Check In

adicionadas, ver Figura 10.

No URI `/html/admin/searchroom` é iniciado a procura das salas tal como no FENIX API quando não indicado o <id> é enviado os três CAMPUS

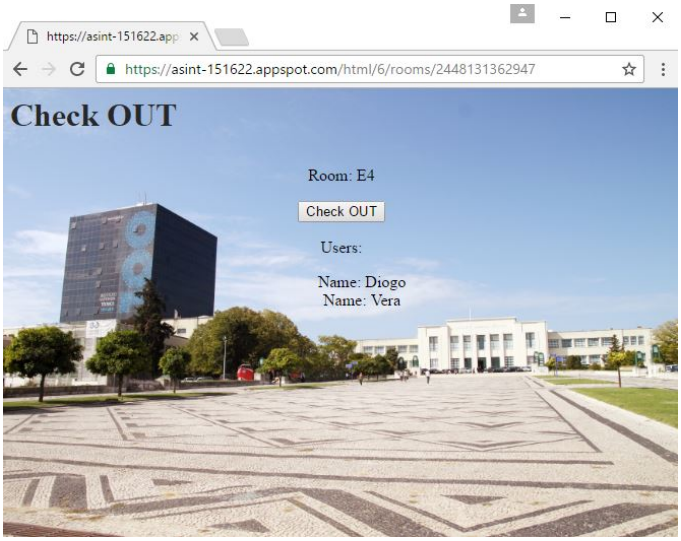

Figure 9: Página de Check Out

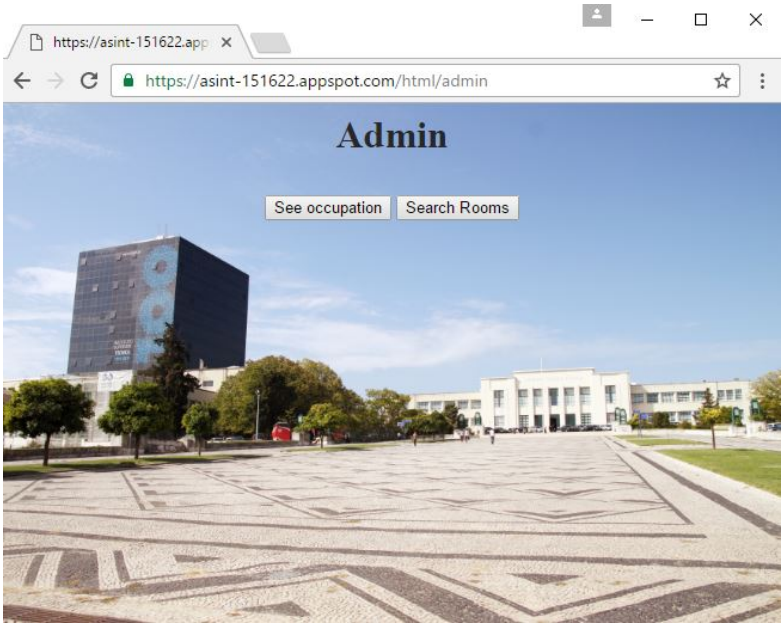

Figure 10: Página de login do utilizador

distintos, ver Figura 11.

Quando indicado o <id> em `/html/admin/searchroom/<id>` apresenta-

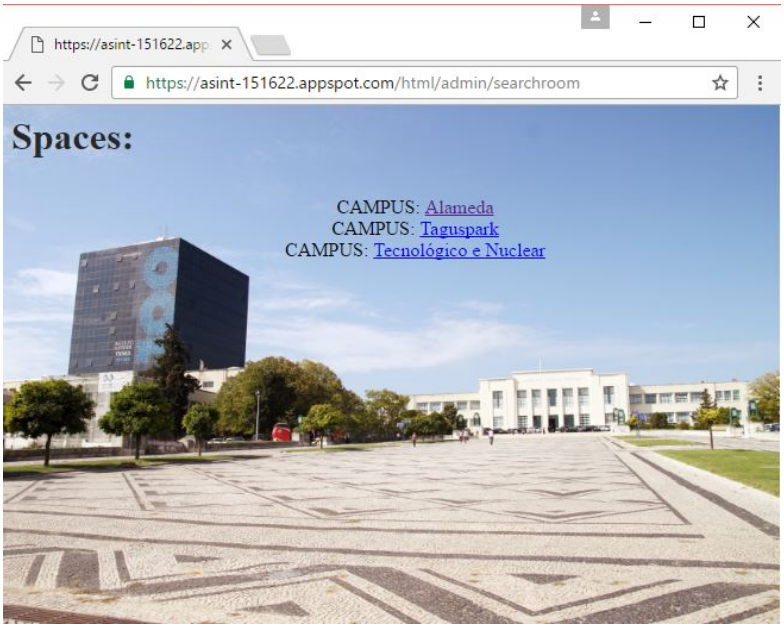

Figure 11: Procura de Salas - CAMPUS

se os espaços contidos no espaço indicado, ver Figura 12.

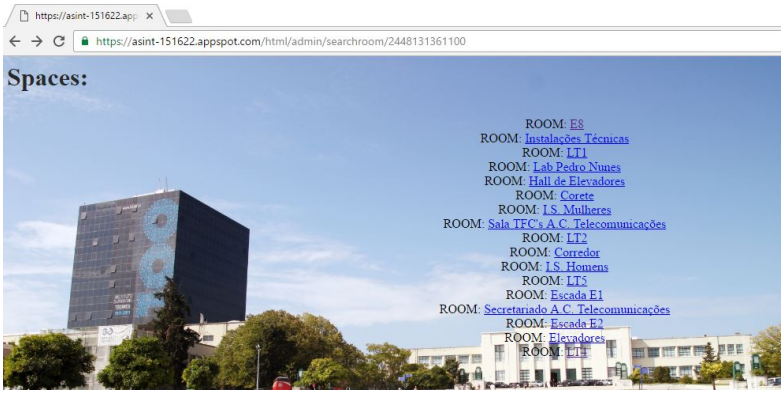

Figure 12: Procura de salas - ROOMS

Para a mesma extensão caso o identificador pertença a um ROOM surge a possibilidade de adicionar este às salas disponíveis, ver Figura 13.

Na extensão `/html/admin/occupation` aparece uma lista de todas as salas que o administrador colocou disponível sendo possível selecionar uma

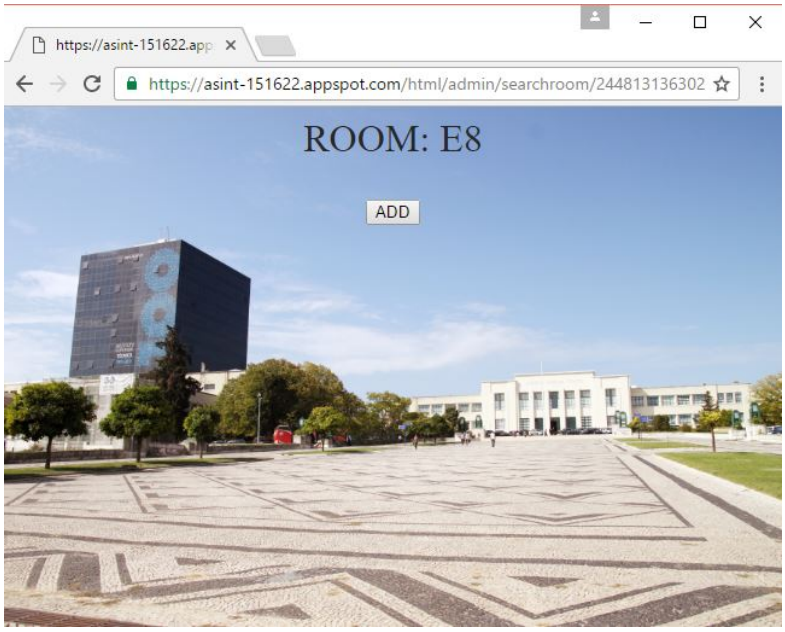

Figure 13: Procura de salas - ROOM

delas para ver a sua ocupação em particular, ver na Figura 14.

Para a extensão `/html/admin/occupation/<id>` é possível ver a ocupação da sala que tem esse id, para além do número de utilizadores é possível quem estes são, pode ser visto na Figura 15.

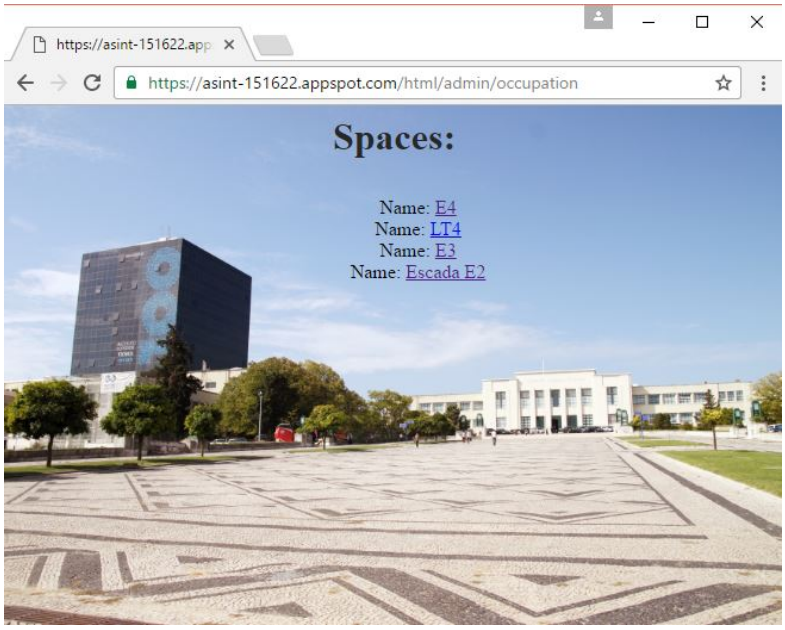

Figure 14: Salas disponíveis

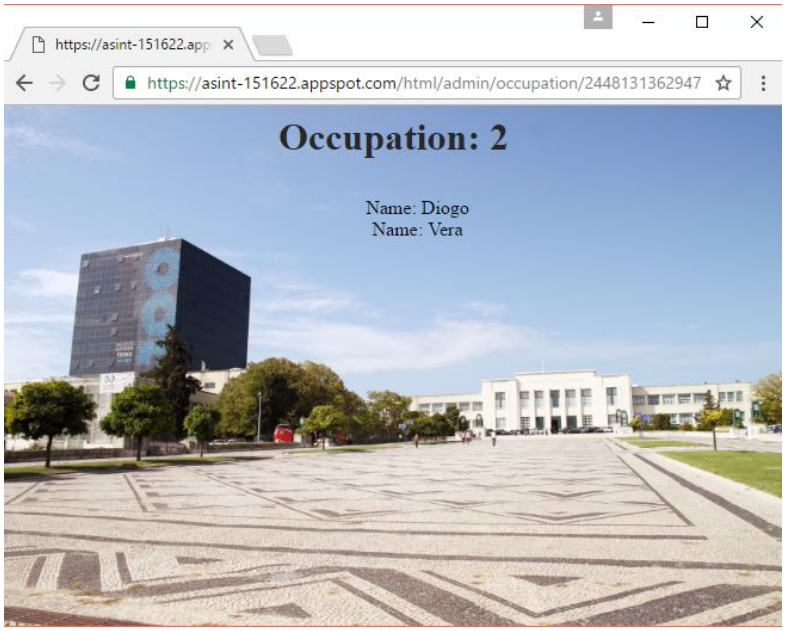

Figure 15: Ocupação da sala selecionada