

SMART ENERGY



Integração de Sistemas de Informação
Licenciatura em Engenharia de Sistemas Informáticos
Regime Pós-Laboral
2022/2023

Alunos

Tiago Azevedo - N° 21153
Francisco Pereira – N° 21156

Orientação

Profº Óscar Ribeiro

Índice

1.	Introdução.....	1
2.	Levantamento de Requisitos	2
2.1.	Diagrama de Casos de uso.....	2
2.2.	Diagrama de Modelo de Dados (ER).....	3
2.3.	Diagramas de Atividade.....	4
2.4.	Diagrama de Estados	5
2.5.	Diagramas de Sequência	6
3.	Arquitetura da Solução	7
3.1.	Serviços FrontEnd	8
3.1.1.	FE#01 – Criar Utilizador.....	8
3.1.2.	FE#02 – Ver Utilizadores	8
3.1.3.	FE#03 – Autenticação Normal.....	9
3.1.4.	FE#04 – Autenticação Google-Auth	9
3.1.5.	FE#05 – Ver Equipamentos	9
3.1.6.	FE#06 – Ver Eventos de Equipamento	10
3.2.	Serviços BackEnd	11
3.2.1.	BE#01 – Autenticação de Poste	11
3.2.2.	BE#02 – Adicionar Evento	11
3.3.	Estratégia de Implementação.....	12
4.	Implementação	13
4.1.	Modelo.....	13
4.2.	Vistas / WebServices	15
4.2.1.	Users.....	16

4.2.2.	Groups	16
4.2.3.	Logs.....	17
4.2.4.	Devices	17
5.	Google Cloud.....	18
6.	Vista de Administrador	18
7.	Conclusão.....	19
8.	Bibliografia	20

Índice de ilustrações

Figura 1 - Diagrama de Casos de Uso.....	2
Figura 2 - Diagrama Entidade-Relação.....	3
Figura 3 - Diagrama de atividade de Inicialização.....	4
Figura 4 - Diagrama de atividade de Detecção de Movimento.....	4
Figura 5 - Diagrama de estados	5
Figura 6 - Diagrama de sequência do Movimento Não Ativa a Luz.....	6
Figura 7 - Diagrama de sequência do Movimento Ativa a Luz	6
Figura 8 - Camadas Lógicas do nosso sistema	7
Figura 9 - Funcionalidades da Camada de integração	7
Figura 10 - Esquema de Implementação	12
Figura 11 - Modelos de Base de Dados.....	13
Figura 12 - Diagrama ER - Django	14
Figura 13 - Serviços desenvolvidos	15
Figura 14 - Serviço de gestão de Utilizadores.....	16
Figura 15 - Serviços de Grupos	16
Figura 16 - Serviço de Logs.....	17
Figura 17 - Serviço de Dispositivos	17
Figura 18 - Django Admin.....	18

1. Introdução

No âmbito da UC de Integração de Sistemas de Informação, pretende-se desenvolver um projeto que visa usar os meios tecnológicos da atualidade para tornar um dos setores do nosso campus mais eficiente, tanto em termos funcionais como económicos. Este tema provém da proposta apresentada na unidade curricular de Projeto Aplicado, que pretende interligar várias UC's com o mesmo projeto.

A nossa equipa optou por escolher o setor energético considerando que o mesmo tem uma boa base para melhorias em termos de eficiência e, por conseguinte, em termos económicos. Um caso prático do nosso projeto será, por exemplo, otimizar a utilização das lâmpadas dos postes de iluminação das estradas com o objetivo de rentabilizar e prolongar o tempo de vida útil das lâmpadas, evitando assim, manutenções desnecessárias e, por consequência, reduzir a mão de obra de manutenção.

Neste momento não existe qualquer tipo de automatização no que toca ao controlo das luzes nos parques de estacionamento e vias públicas. Uma grande preocupação nos dias de hoje é o gasto excessivo de eletricidade, até porque grande parte dessa energia provém de combustíveis fósseis, levando a uma pegada de carbono significativa.

Este projeto poderá também contribuir para as metas da Comissão Europeia para as Smart Cities em 2030, que atualmente se encontram em risco de não serem alcançadas.

2. Levantamento de Requisitos

2.1. Diagrama de Casos de uso

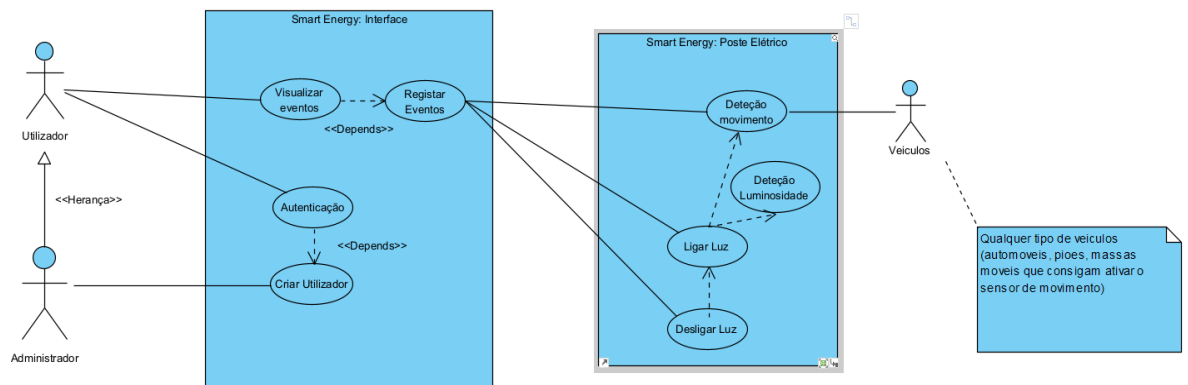


Figura 1 - Diagrama de Casos de Uso

Este diagrama foi concebido para demonstrar os casos de uso presentes no nosso projeto. Este, encontra-se dividido em dois sistemas, que se encontram interligados entre si. O primeiro sistema, “Smart Energy: Interface” é responsável pela visualização de eventos, pela autenticação e pela criação de utilizadores. O segundo sistema, “Smart Energy: Poste Elétrico”, é responsável pela deteção do movimento e pela deteção da luminosidade. Até ao momento, existem três atores: **Administrador, Utilizador e Veiculos**.

O Administrador será responsável por criar utilizadores, bem como tudo o que o utilizador é capaz de fazer, existindo aqui uma relação de herança para com o Utilizador.

Já o Utilizador poderá fazer o uso supervisionado do sistema, nomeadamente para recolha de dados e manutenção da aplicação, visualizando os eventos existentes até à data.

O ator Veiculos, representa qualquer tipo de veículos, sejam eles automóveis, sejam peões, entre outros, desde que consigam ativar o sensor de movimento.

Este diagrama poderá ser alterado mais tarde mediante as necessidades inerentes ao projeto.

2.2. Diagrama de Modelo de Dados (ER)

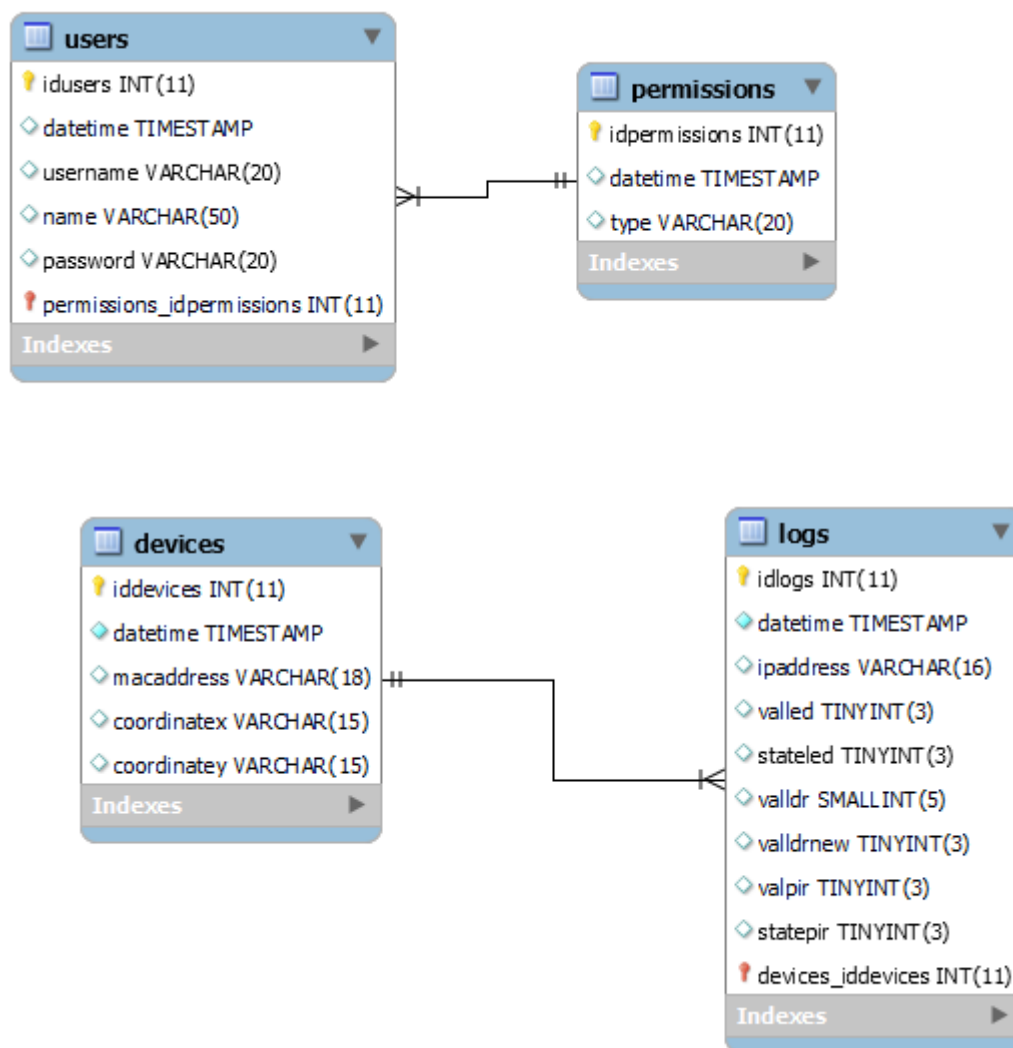


Figura 2 - Diagrama Entidade-Relação

O diagrama Entidade-Relação foi construído já com a concessão da base de dados em mente para que todo o projeto seja alvo de análise posterior. De modo a garantir isso, foram criadas quatro tabelas:

- *Users & Permissions*: Permitem guardar permanentemente os Utilizadores do sistema, assim como as suas respetivas permissões (ex: Administrador);
- *Devices & Logs*: Nestas duas tabelas interligadas, são guardados todos os dados relativamente aos dispositivos que fazem parte do sistema e, de

forma a obter dados concretos da análise dos mesmos, são guardados dados de diagnóstico ou históricos para poderem ser consultados na eventualidade de ser necessário, mediante uma data ou até mesmo um código de dispositivo.

Em suma, o desenvolvimento deste diagrama e a respetiva base de dados são um paço crucial para garantir um sistema seguro e viável de ser expandido sem a necessidade de guardar dados noutro tipo de formato.

2.3. Diagramas de Atividade

Os diagramas de atividade são um tipo de diagrama que mostra o fluxo de atividades ou ações dentro do nosso sistema. É uma representação visual da sequência de atividades que devem ser realizadas para concluir os processos ou fluxos de trabalho.

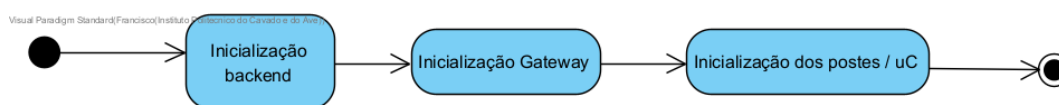


Figura 3 - Diagrama de atividade de Inicialização

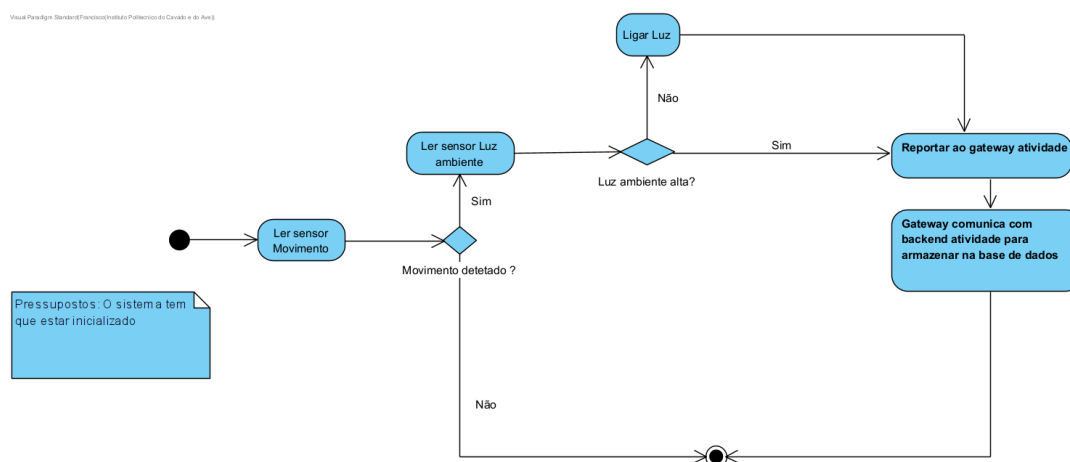


Figura 4 - Diagrama de atividade de Detecção de Movimento

2.4. Diagrama de Estados

Os diagramas de estados são um tipo de diagrama que mostra o comportamento do nosso sistema ao longo do tempo. Eles são usados para modelar a dinâmica do sistema e mostrar como ele muda de um estado para outro ao longo do tempo.

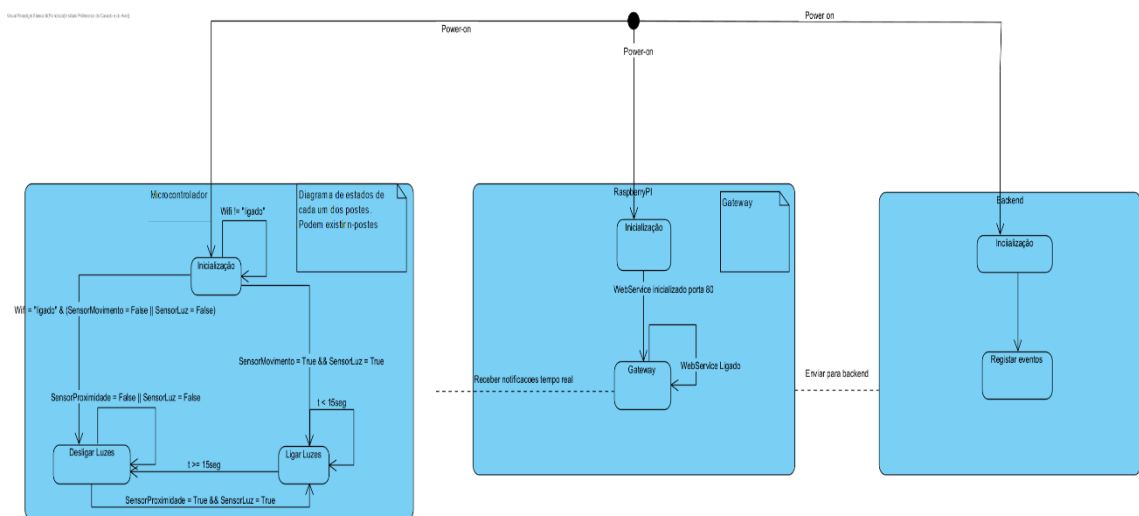


Figura 5 - Diagrama de estados

Este diagrama foi concebido para demonstrar os possíveis estados do poste. Dessa forma, faz sentido começar a explicar os estados do poste. O poste tem três estados: os:

- Inicialização : Objetivo de ligar ao gateway via wifi e inicializar os sensores.
- Desligar Luzes: Serve para desligar a luz do poste.
- Ligar Luzes: Serve para ligar a luz do poste.

Dentro destes três estados, as transições são feitas através dos dados dos sensores maioritariamente. Por exemplo, no caso de o sistema estar no estado "Desligar Luzes", apenas pode transitar para "Ligar Luzes" se detetar movimento e a intensidade de Luz ambiente for baixa.

Acerca do Gateway e backend, estes apenas dependem maioritariamente de notificações enviadas pelo poste, sendo que, não existem mais estados relevantes a identificar, além de, armazenamento da informação.

2.5. Diagramas de Sequência

Os diagramas de sequência são um tipo de diagrama que mostra a interação entre diferentes entidades no nosso sistema. Ele é usado para modelar a sequência de mensagens trocadas entre as entidades no nosso sistema, bem como a ordem em que essas mensagens são trocadas.

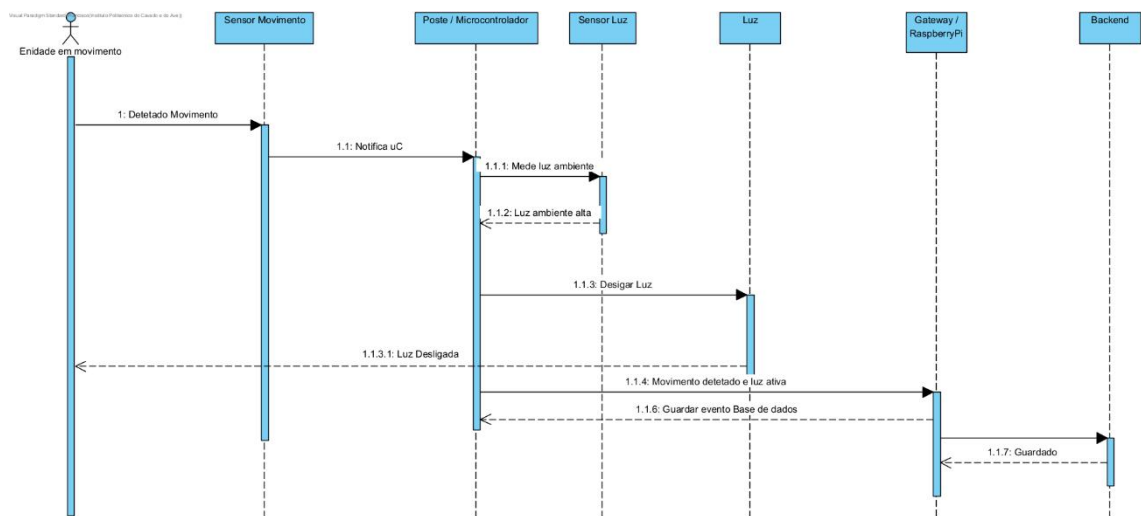


Figura 6 - Diagrama de sequência do Movimento Não Ativa a Luz

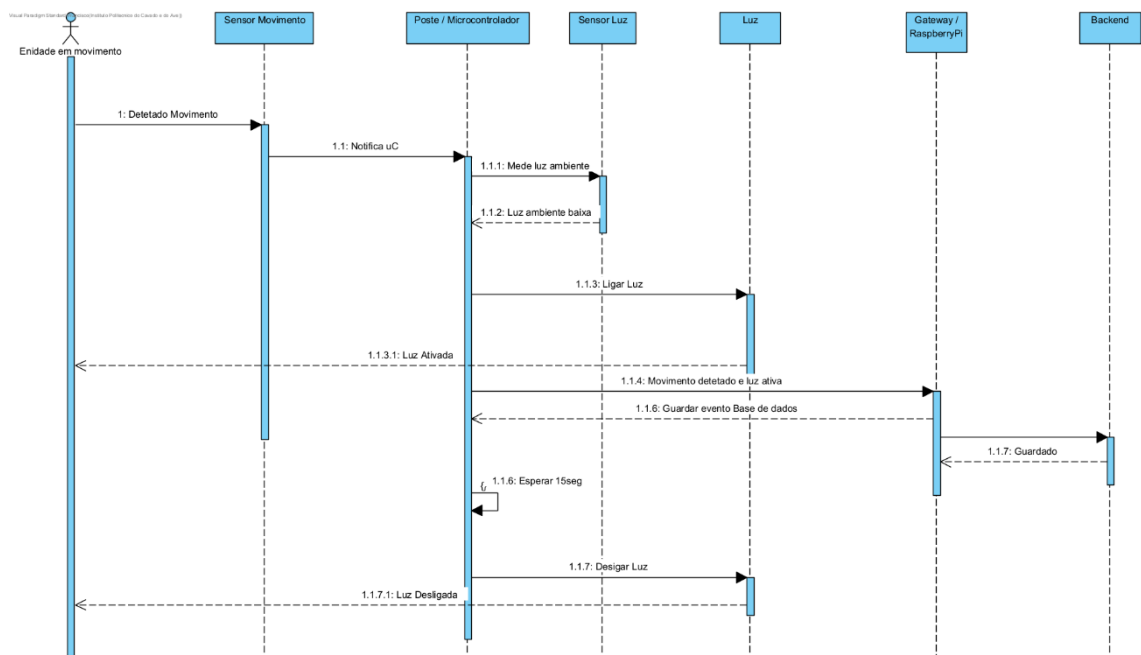


Figura 7 - Diagrama de sequência do Movimento Ativa a Luz

3. Arquitetura da Solução

Esta secção documenta todas as decisões arquiteturais do projeto SmartEnergy. Este, irá ser composto maioritariamente por três camadas distintas:



Figura 8 - Camadas Lógicas do nosso sistema

No contexto da disciplina de Integração de Sistemas de Informação, a camada que será descrita neste relatório é a camada de Backend. Para mais informação acerca do projeto aconselha-se a leitura do documento [1].

A camada de Integração tem como objetivo receber e guardar informação enviado pelos postes e providenciar esta informação organizada para a aplicação Mobile (frontend). A aplicação mobile terá a responsabilidade de providenciar uma interface simples e intuitiva aos utilizadores finais.

A camada de Backend tem a responsabilidade de providenciar os seguintes serviços/funcionalidades para o Frontend e Backend, como apresentado na seguinte figura:



Figura 9 - Funcionalidades da Camada de integração

3.1. Serviços FrontEnd

Para mais fácil visualização dos serviços implementados na camada do FrontEnd, a próxima secção será apresentada em formato de tabela com os seguintes dados:

- Nome do Serviço;
- Objetivo;
- Parâmetros;
- Retorno;
- Caminho;
- Pré-condições

3.1.1. FE#01 – Criar Utilizador

Nome do Serviço	Criar Utilizador
Objetivo	Permitir o Frontend criar um novo utilizador
Parâmetros	Username/Email: String Password: String Tipo: <Utilizador, Administrador>
Retorno	Resultado Operação: Boolean
Caminho	/utilizador/criar
Pré-condições	Utilizador com login feito e administrador

3.1.2. FE#02 – Ver Utilizadores

Nome do Serviço	Ver utilizadores
Objetivo	Permitir o Frontend Listar Utilizadores
Parâmetros	
Retorno	Resultado Operação: Boolean Lista Username/Email: String
Caminho	/utilizador/listar
Pré-condições	Utilizador com login feito e administrador

3.1.3. FE#03 – Autenticação Normal

Nome do Serviço	Autenticação
Objetivo	Permitir o Frontend autenticar utilizadores
Parâmetros	Utilizador: String Password: String
Retorno	Resultado Operação: Boolean
Caminho	/autenticacao/normal
Pré-condições	Utilizador tem de estar registado no Smart Energy

3.1.4. FE#04 – Autenticação Google-Auth

Nome do Serviço	Autenticação
Objetivo	Permitir o Frontend autenticar utilizadores utilizando conta google
Parâmetros	Email: String Password: String
Retorno	Resultado Operação: Boolean
Caminho	/autenticação/2google
Pré-condições	Utilizador tem de estar registado no Smart Energy

3.1.5. FE#05 – Ver Equipamentos

Nome do Serviço	Ver Equipamentos
Objetivo	Permitir o Frontend receber a lista de equipamentos
Parâmetros	ID equipamento: String Token login: String
Retorno	Resultado Operação Boolean ID equipamento: String Estado: Online/Offline

Caminho	/equipamentos/listar
Pré-condições	Utilizador com login feito

3.1.6. FE#06 – Ver Eventos de Equipamento

Nome do Serviço	Ver Eventos de Equipamento
Objetivo	Permitir o Frontend ver detalhes de equipamento
Parâmetros	Token login: String ID equipamento: String
Retorno	Resultado Operação: Boolean ID equipamento: String Estado: Online/Offline Mac Address: String IPV4 Address: String Localizacao: String Lista Eventos: Data: Date Iluminação: Float Estado Iluminação: Boolean Valor Luz: Float Movimento: Boolean
Caminho	/equipamentos/ver?id=<ID>
Pré-condições	Utilizador com login feito

3.2. Serviços BackEnd

Para mais fácil visualização dos serviços implementados na camada do BackEnd, tal como na secção anterior, a próxima secção será apresentada em formato de tabela com os mesmos dados da secção anterior.

3.2.1. BE#01 – Autenticação de Poste

Nome do Serviço	Autenticação
Objetivo	Permitir aos postes/equipamentos adicionar um novo evento
Parâmetros	ID Poste: String
Retorno	Resultado Operação: Boolean Token Autenticação: String
Caminho	/autenticação/poste
Pré-condições	Nenhuma

3.2.2. BE#02 – Adicionar Evento

Nome do Serviço	Autenticação
Objetivo	Permitir aos postes/equipamentos adicionar um novo evento
Parâmetros	ID Poste: String Token Autenticação: String Estado: Online/Offline Mac Address: String IPV4 Address: String Localizacao: String Data: Date Iluminação: Float Estado Iluminação: Boolean Valor Luz: Float Movimento: Boolean
Retorno	Resultado Operação: Boolean
Caminho	/poste/evento
Pré-condições	Poste tem de estar autenticado

3.3. Estratégia de Implementação

Para implementar a arquitetura descrita, foi decidido em grupo, efetuar a hospedagem na Google Cloud. A linguagem de programação utilizada será Python com a Framework Django. Será utilizado a biblioteca do Django de REST para os WebServices onde estarão hospedados os serviços de frontend e backend. A próxima figura apresenta as ferramentas a ser utilizadas para a implementação da solução Smart Energy:

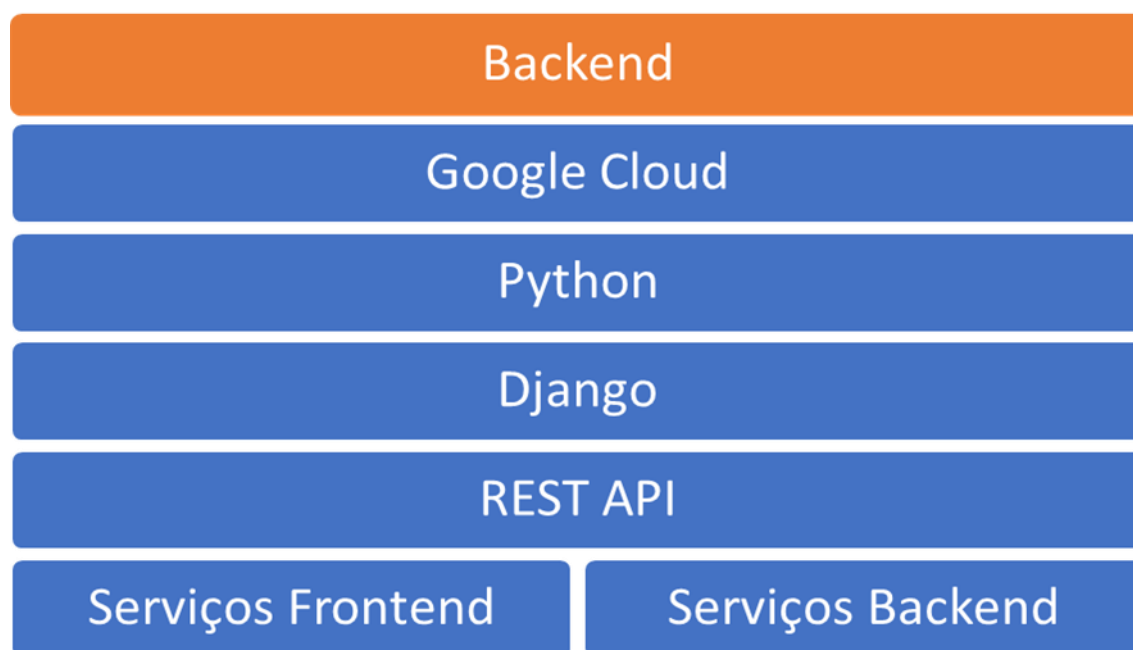


Figura 10 - Esquema de Implementação

4. Implementação

Este capítulo documenta todas as decisões técnicas consideradas para o desenvolvimento do backend Smart Energy. Como referido anteriormente, o projeto foi desenvolvido em Python com a Framework Django [2] e o hosting na Google Cloud [3].

O repositório do código está armazenado em três sítios:

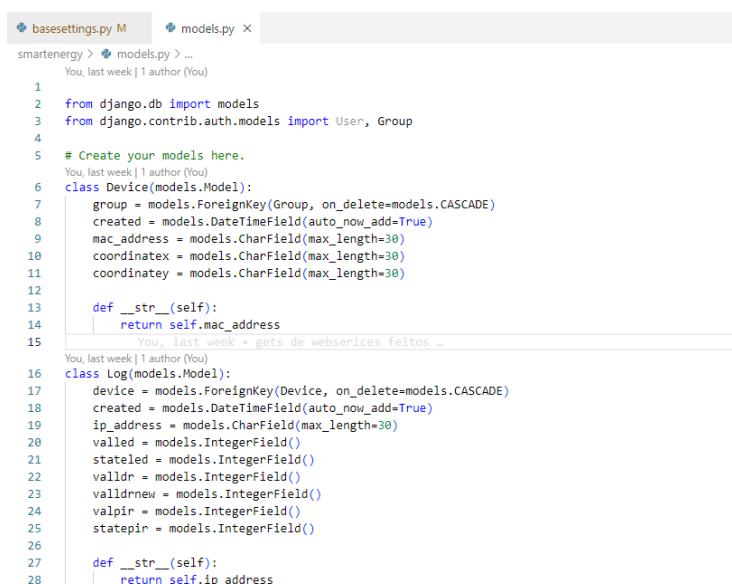
- Github: https://github.com/tiagoazevedo22/LESI3_SmartCampus.git (Pasta: ISI com um git submodule);
- Github: https://github.com/Franciscopereira2001/isi_tp2/tree/master ;
- Google Cloud:

<https://franciscorafaeldpereira@gmail.com@source.developers.google.com:2022/p/s/martenergy-backend/r/smartenergy-backend>.

É de referir que todos os serviços precisam de ser autenticados.

4.1. Modelo

Este capítulo documenta a estrutura da base de dados. A base de dados foi desenvolvida utilizando os modelos do Django [4]. Estes modelos recebidos pela Framework Django e gerado o SQL correspondente. A próxima figura apresenta os modelos resultantes:



```
1 from django.db import models
2 from django.contrib.auth.models import User, Group
3
4 # Create your models here.
5
6 class Device(models.Model):
7     group = models.ForeignKey(Group, on_delete=models.CASCADE)
8     created = models.DateTimeField(auto_now_add=True)
9     mac_address = models.CharField(max_length=30)
10    coordinate = models.CharField(max_length=30)
11    coordinatey = models.CharField(max_length=30)
12
13    def __str__(self):
14        return self.mac_address
15
16 class Log(models.Model):
17     device = models.ForeignKey(Device, on_delete=models.CASCADE)
18     created = models.DateTimeField(auto_now_add=True)
19     ip_address = models.CharField(max_length=30)
20     valled = models.IntegerField()
21     stateled = models.IntegerField()
22     valldr = models.IntegerField()
23     valldrnew = models.IntegerField()
24     valpir = models.IntegerField()
25     statepir = models.IntegerField()
26
27    def __str__(self):
28        return self.ip_address
```

Figura 11 - Modelos de Base de Dados

Algumas considerações:

1. O Utilizador criado é o disponibilizado por defeito no Django, onde, é possível gerir e associar grupos e assim, permissões. Esta característica será usada para atribuir dispositivos a grupos, e desta forma, cumprir o requisito de gestão de permissões.
2. Um Utilizador pode fazer parte de um ou mais grupos
3. Um dispositivo está associado a um único grupo
4. A base de dados foi criada pelo Django através do processamento dos modelos [4], que são apresentados na secção anterior.

4.2. Vistas / WebServices

No Django / Django RestFramework uma vista é um serviço. Os serviços desenvolvidos tomaram como especificação os definidos anteriormente.

Após execução do servidor (por exemplo localmente) através da execução do comando: ***python manage.py runserver***

Poderão ser consultados os serviços desenvolvidos.

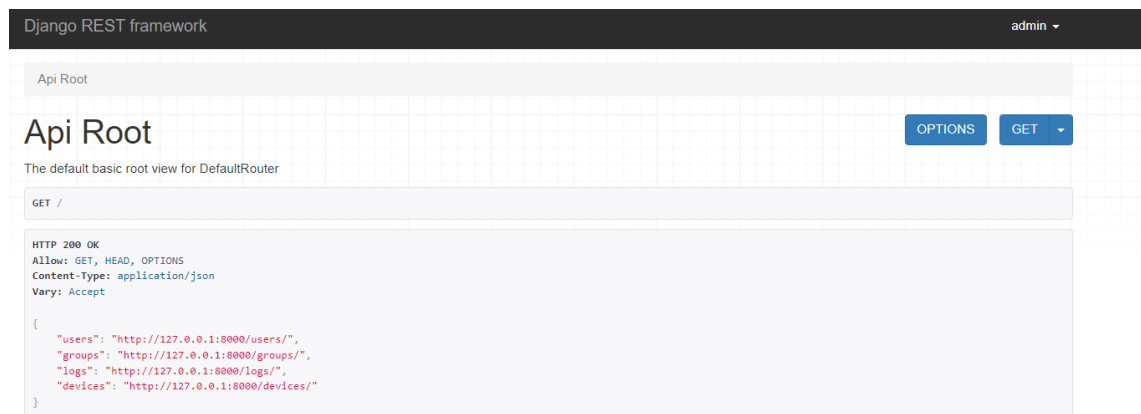
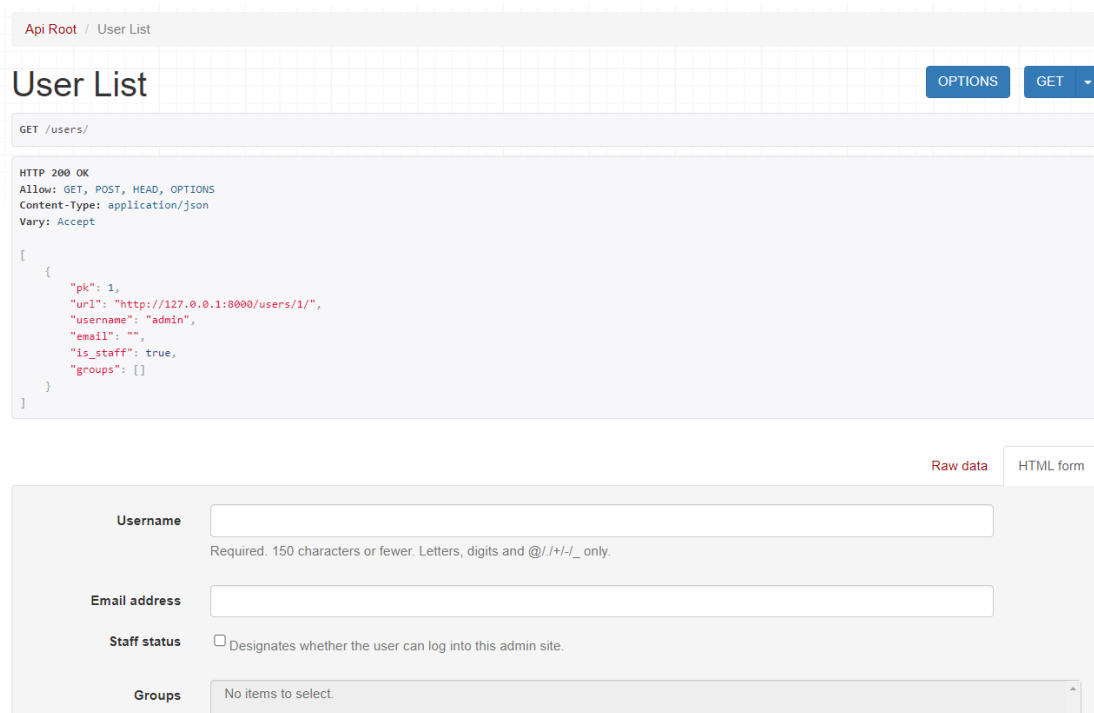


Figura 13 - Serviços desenvolvidos

4.2.1. Users

Interfaces:

- GET -> Listar
- POST -> Adicionar



Api Root / User List

User List

GET /users/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "pk": 1,
    "url": "http://127.0.0.1:8000/users/1/",
    "username": "admin",
    "email": "",
    "is_staff": true,
    "groups": []
  }
]
```

Raw data HTML form

Username
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address

Staff status ☐ Designates whether the user can log into this admin site.

Groups No items to select.

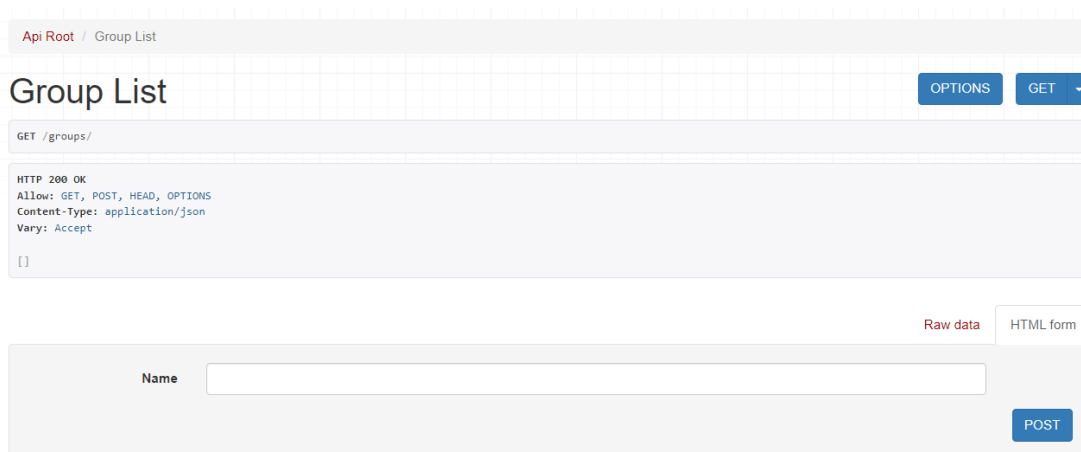
Figura 14 - Serviço de gestão de Utilizadores

4.2.2. Groups

Interfaces:

- GET -> Listar

A gestão dos grupos será feita no Django Admin.



Api Root / Group List

Group List

GET /groups/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[ ]
```

Raw data HTML form

Name

POST

Figura 15 - Serviços de Grupos

4.2.3. Logs

Interfaces:

- GET -> Listar / Consultar
- POST -> Adicionar

The screenshot shows the 'Log List' interface. At the top, there's a breadcrumb 'Api Root / Log List'. Below it, the title 'Log List' is displayed next to 'OPTIONS' and 'GET' buttons. The URL bar shows 'GET /logs/'. The response area displays an HTTP 200 OK status with headers: 'Allow: GET, POST, HEAD, OPTIONS', 'Content-Type: application/json', and 'Vary: Accept'. The response body is an empty array '[]'. Below this, there are tabs for 'Raw data' and 'HTML form'. The 'HTML form' tab is active, showing a form with the following fields: 'Device' (a dropdown menu), 'Ip address', 'Valled', 'Stateled', 'Valldr', 'Valldrnew', 'Valpir', and 'Statepir'. A 'POST' button is located at the bottom right of the form.

Figura 16 - Serviço de Logs

4.2.4. Devices

Interfaces:

- GET -> Listar, Consultar
- POST -> Adicionar

The screenshot shows the 'Device List' interface. At the top, there's a breadcrumb 'Api Root / Device List'. Below it, the title 'Device List' is displayed next to 'OPTIONS' and 'GET' buttons. The URL bar shows 'GET /devices/'. The response area displays an HTTP 200 OK status with headers: 'Allow: GET, POST, HEAD, OPTIONS', 'Content-Type: application/json', and 'Vary: Accept'. The response body is an empty array '[]'. Below this, there are tabs for 'Raw data' and 'HTML form'. The 'HTML form' tab is active, showing a form with the following fields: 'Mac address', 'Coordinatex', 'Coordinatey', and 'Group' (a dropdown menu). A 'POST' button is located at the bottom right of the form.

Figura 17 - Serviço de Dispositivos

5. Google Cloud

Como referido anteriormente, anteriormente, o Hosting da plataforma será feito no Google Cloud e pode ser acedido através do seguinte link:

<https://django-cloudrun-gbt6bcmkta-nw.a.run.app>

O processo de integração do projeto foi feito conforme este tutorial da google [3].

O utilizador de Administrador por defeito detém as seguintes credenciais:

- **Username:** admin
- **Password:** admin

6. Vista de Administrador

A gestão da base de dados pode ser feita utilizando a funcionalidade do Django, Django Admin [5].

No site hospedado na Google Cloud, esta vista pode ser acedida através:

<https://django-cloudrun-gbt6bcmkta-nw.a.run.app/admin>

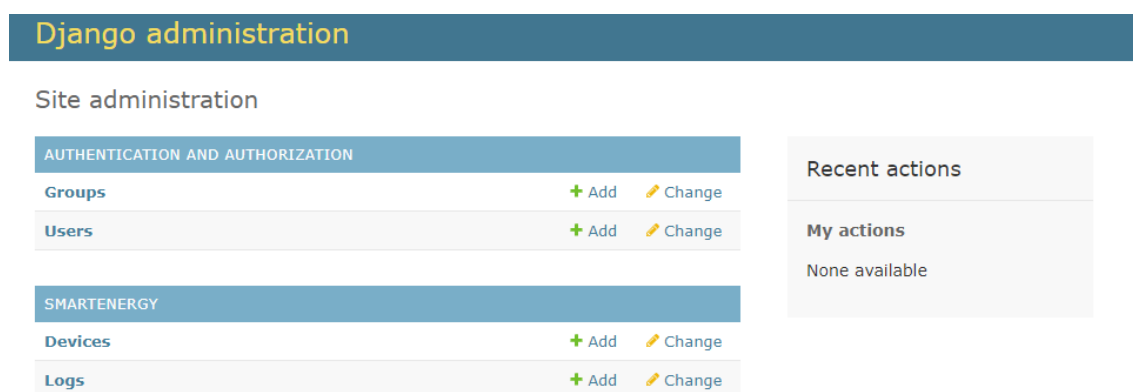


Figura 18 - Django Admin

Nesta página é possível remover e adicionar dados e cada uma das tabelas de uma forma simples e com ajudas pré-desenvolvidas no Django.

7. Conclusão

Este trabalho foi muito desafiante desenvolver devido ao facto de termos tido de aprender duas tecnologias, o Python/Django e Google Cloud. Foram atingidos todos os objetivos definidas na parte de Arquitetura e estamos em uma posição vantajosa para iniciar o trabalho de desenvolvimento movel que irá assentar sobre esta solução. O Django foi a framework adequada devido ao facto de já ter pré-implementados vários processos e de ser fácil a criação de uma base de dados.

Tivemos imensos problemas na integração da Google Cloud, maioritariamente devido a ligações à base de dados da GCloud e porque não configuramos corretamente as configurações de região dando origem a vários problemas. Esta integração consumiu bastante tempo.

Foi um trabalho muito interessante e que cumpriu todos os objetivos e expectativas.

8. Bibliografia

- [1] G. 6, “Relatório de Projeto Aplicado: Smart Energy,” IPCA, Barcelos, 2022.
- [2] Django, “Django Project,” [Online]. Available: <https://www.djangoproject.com/>.
- [3] google, “Django on Cloud Run,” [Online]. Available: <https://codelabs.developers.google.com/codelabs/cloud-run-django>. [Acedido em 3 1 2023].
- [4] Django Project, “Migrations,” [Online]. Available: <https://docs.djangoproject.com/en/4.1/topics/migrations/>. [Acedido em 3 1 2023].
- [5] Django, “Django Admin,” [Online]. Available: <https://docs.djangoproject.com/en/4.1/ref/contrib/admin/>. [Acedido em 6 1 2023].
- [6] Django Rest Framework, “Django Rest Framework,” [Online]. Available: <https://www.django-rest-framework.org/#quickstart>. [Acedido em 3 1 2023].