

SMART ENERGY



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

Alunos

Francisco Pereira – N° 21156

Tiago Azevedo - N° 21153

Orientação

Profº Óscar Ribeiro

Conteúdo

Arquitetura Solução	3
Serviços Frontend.....	3
Serviço: Criar Utilizador.....	3
Serviço: Ver Utilizadores	4
Serviço: Autenticação Normal Frontend	4
Serviço: Autenticação Google-Auth Frontend	4
Serviço: Ver equipamentos	4
Serviço: Ver Eventos equipamento	4
Serviços Backend.....	5
Serviço: Autenticação Poste	5
Serviço: Adicionar Evento	5
Estratégia de Implementação	5
Modelo de Dados	6
Implementação	7
Modelo	7
Base de dados	8
Vistas / WebServices	8
Users.....	8
Groups.....	9
Logs.....	9
Devices	10
Google Cloud	11
Vista de Administrador.....	11
Conclusão	12
Bibliografia	13
Anexos.....	14
1. Diagrama Entidade Relação – Django	14

Arquitetura Solução

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



Figura 1: Camadas Lógicas do sistema SmartEnergy

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 Figura 2.



Figura 2: Funcionalidades Camada de Integração

Serviços Frontend

Serviço: 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

Serviço: 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

Serviço: Autenticação Normal Frontend

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

Serviço: Autenticação Google-Auth Frontend

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

Serviço: 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

Serviço: Ver Eventos 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

Serviços Backend

Serviço: Autenticação Poste

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

Serviço: 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 que estar autenticado

Estratégia de Implementação

Para implementar a arquitetura descrita, foi decidido em grupo, efetuar o hospedamento 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 Figura 3 apresenta as ferramentas a ser utilizadas para a implementação da solução Smart Energy.

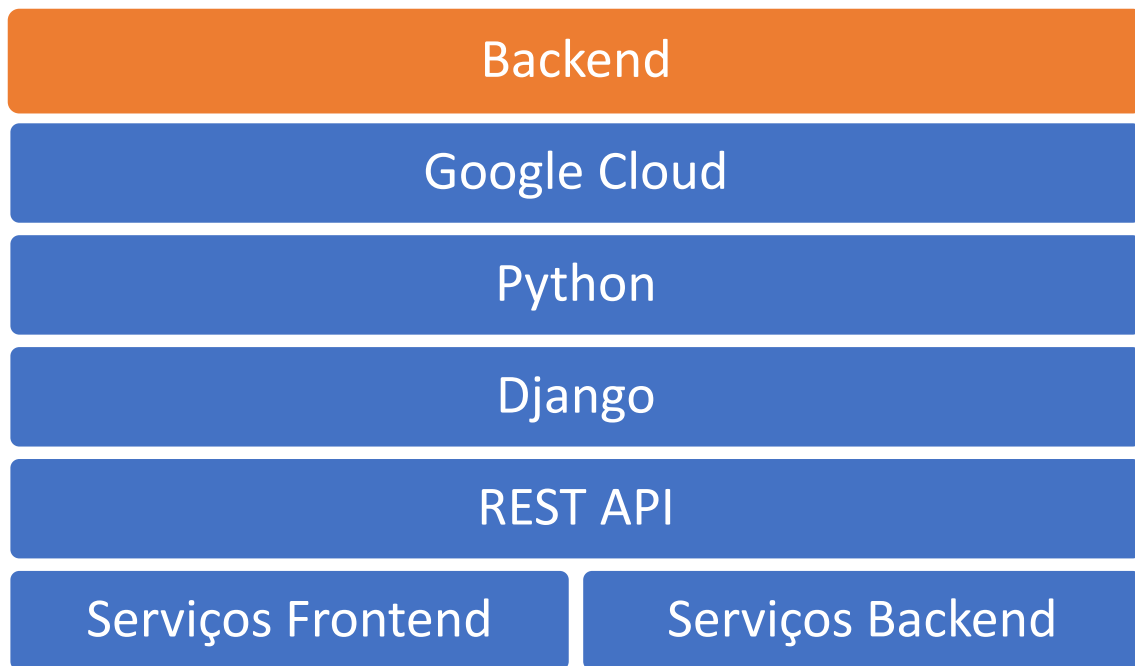


Figura 3: Estratégia de Implementação

Modelo de Dados

Detalhes acerca do modelo de dados estão documentados em detalhe em [1]. A Figura 4, exportada de [1], apresenta o modelo de dados em formato de Entidade-Relação

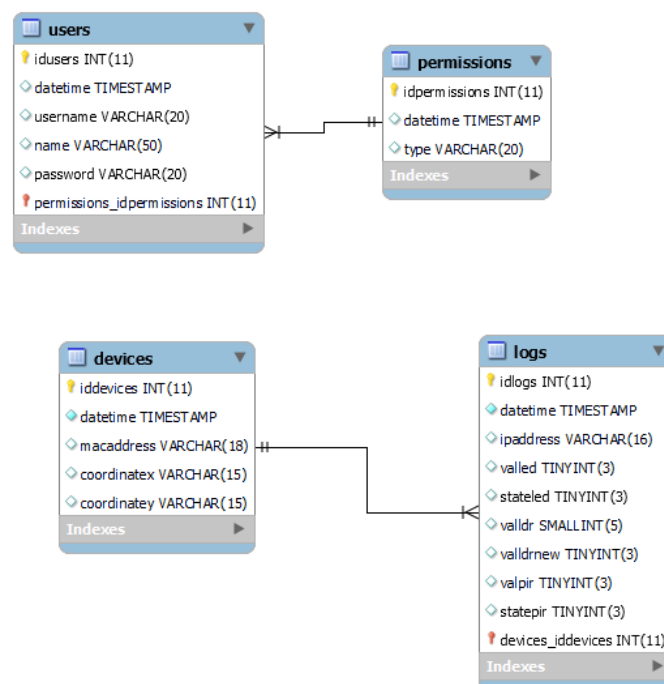


Figura 4: Modelo de dados

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].

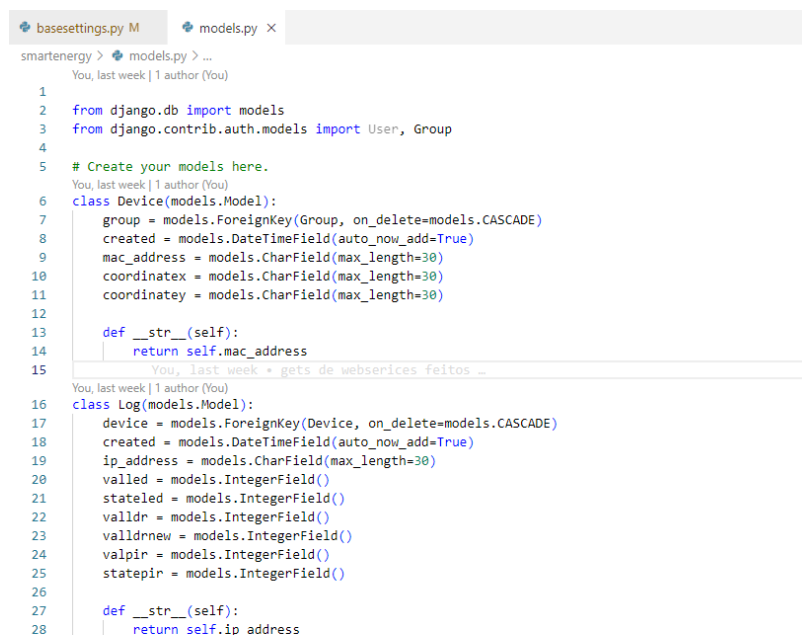
O repositório do código está armazenado em três sítios:

- 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>

De referir, que todos os serviços precisam de ser autenticados.

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 Figura 5 apresenta os modelos resultantes.



```
smartenergy > models.py x ...
You, last week | 1 author (You)

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

Figura 5 - Modelos da Base de dados

Após a execução do comando:

`python manage.py migrate`

O Django irá transformar os modelos acima descritos e irá fazer a interface com a tecnologia de base de dados escolhida, sendo que para o projeto, utiliza SQLite se for lançado localmente ou POSTGRES se lançado no google cloud.

De reter, que o Utilizador e as permissões será o default do Django, User e Group, respetivamente.

Base de dados

A estrutura da base de dados encontra-se apresentada em anexo na Figura 12, onde estão identificadas a vermelho as tabelas apresentadas na página 6, na secção Modelo de Dados. Através da utilização dos mecanismos do Django, foi possível criar várias tabelas de gestão interna do backend. Este facto, justifica o porquê de haver tantas tabelas, no entanto, esta característica permite reutilizar e desenvolver de forma segura e rápida novas aplicações.

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 dados foi criada pelo Django através do processamento dos modelos [4], que são apresentados na secção anterior.

Vistas / WebServices

No Django / Django RestFramework uma vista é um serviço. Os serviços desenvolvidos tomaram como especificação os definidos na página 5.

Após execução do servidor (por exemplo localmente) através da execução do comando:

```
python manage.py runserver
```

Poderão ser consultados os serviços desenvolvidos.

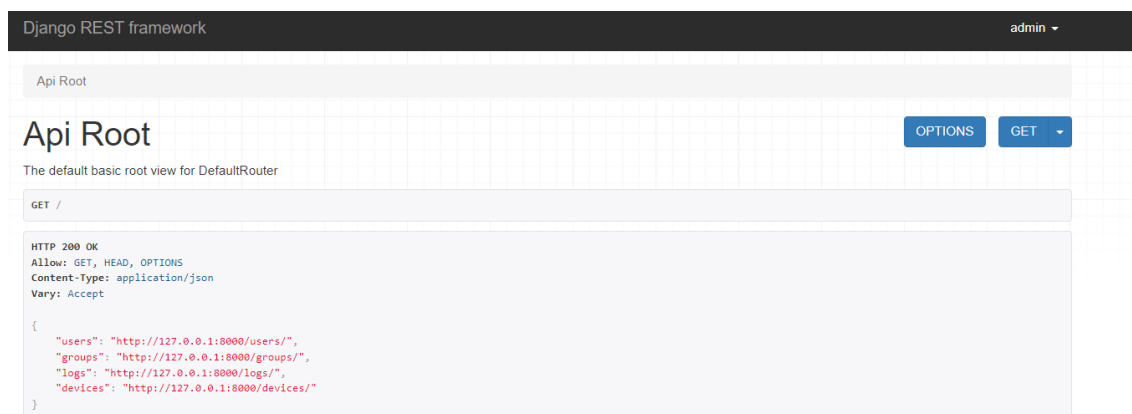


Figura 6 - Serviços desenvolvidos

Users

Interfaces:

- GET -> Listar

- POST -> Adicionar

Api Root / User List

User List

OPTIONS GET

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 7 - Serviço de gestão utilizadores

Groups

Interfaces:

- Get -> Listar

A gestão dos grupos será feita no Django Admin.

Api Root / Group List

Group List

OPTIONS GET

GET /groups/

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

```
[]
```

Raw data HTML form

Name

POST

Figura 8 - Serviço de Grupos

Logs

- Interfaces:
 - Get -> Listar / Consultar
- Post -> Adicionar

Api Root / Log List

Log List

OPTIONS GET

GET /logs/

```

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

```

Raw data HTML form

Device

Ip address

Valled

Stateded

Valldr

Valldrnew

Valpir

Statepir

POST

Figura 9 - Serviço de Logs

Devices

Interfaces:

- Get: Listar, Consultar
- Post: Adicionar

Api Root / Device List

Device List

OPTIONS GET

GET /devices/

```

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

```

Raw data HTML form

Mac address

Coordinatex

Coordinatey

Group

POST

Figura 10 - Serviço de dispositivos

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

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>

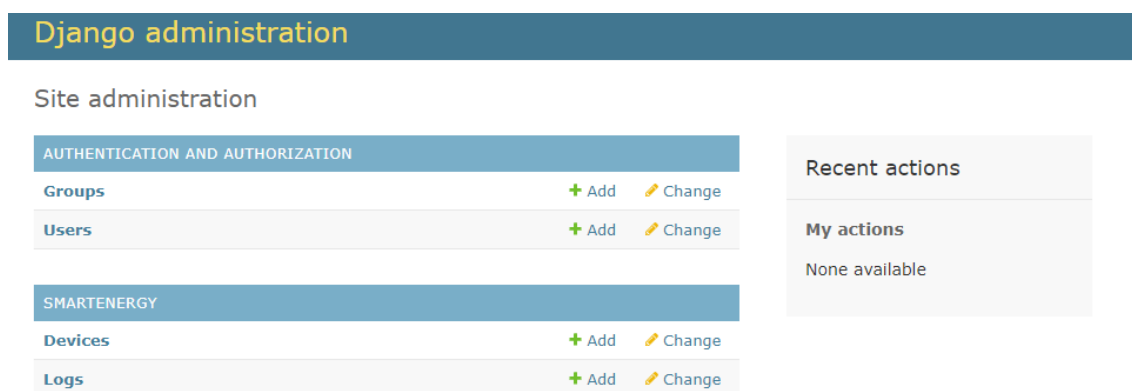


Figura 11 - Django Admin

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

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 definidos 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.

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].

Anexos

1. Diagrama Entidade Relação – Django

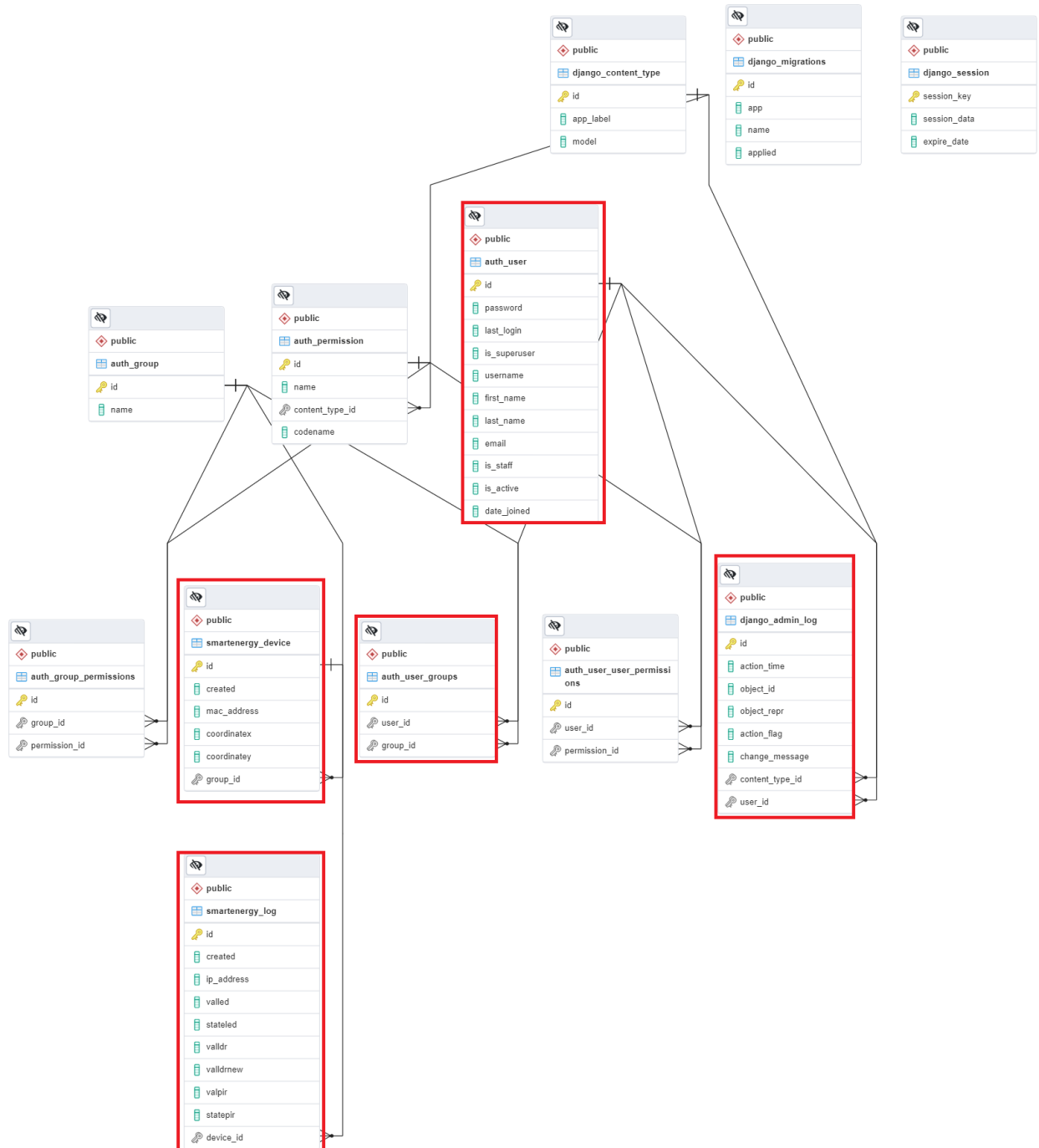


Figura 12: ERD Django