

amg_time_docs

Описание

Решение разработано для

Запуск решения

Преднастройка окружения

Данное решение разработано с использованием фреймворка Django, что подразумевает наличие предварительно установленного Python ≥ 3.12

Кроме того для работы решения требуется развернутая СУБД PostgreSQL или иная, поддерживаемая фреймворком Django [документация](#)

Развертывание решения

Развертывание решения может быть выполнено 2-мя способами:

- через "стандартный" путь с написанием unit'ов Linux
- через Docker

"Стандартный" путь

Для запуска решения необходимо выполнить последовательность действий:

1. Установить зависимости backend

```
pip install -r requirements.txt
```

1. Установить зависимости frontend

```
npm i --legacy-peer-deps
```

1. Запустить backend

```
python manage.py runserver
```

1. Запустить frontend

```
npm start
```

По-умочанию сервисы доступны на следующих портах:

- backend :8000
- frontend :3000

Docker-compose

В комплекте поставки решения подготовлены несколько Dockerfile и docker-compose.yml

Предварительно для развертывания системы через Docker требуется установить необходимые пакеты:

- docker
- docker-compose

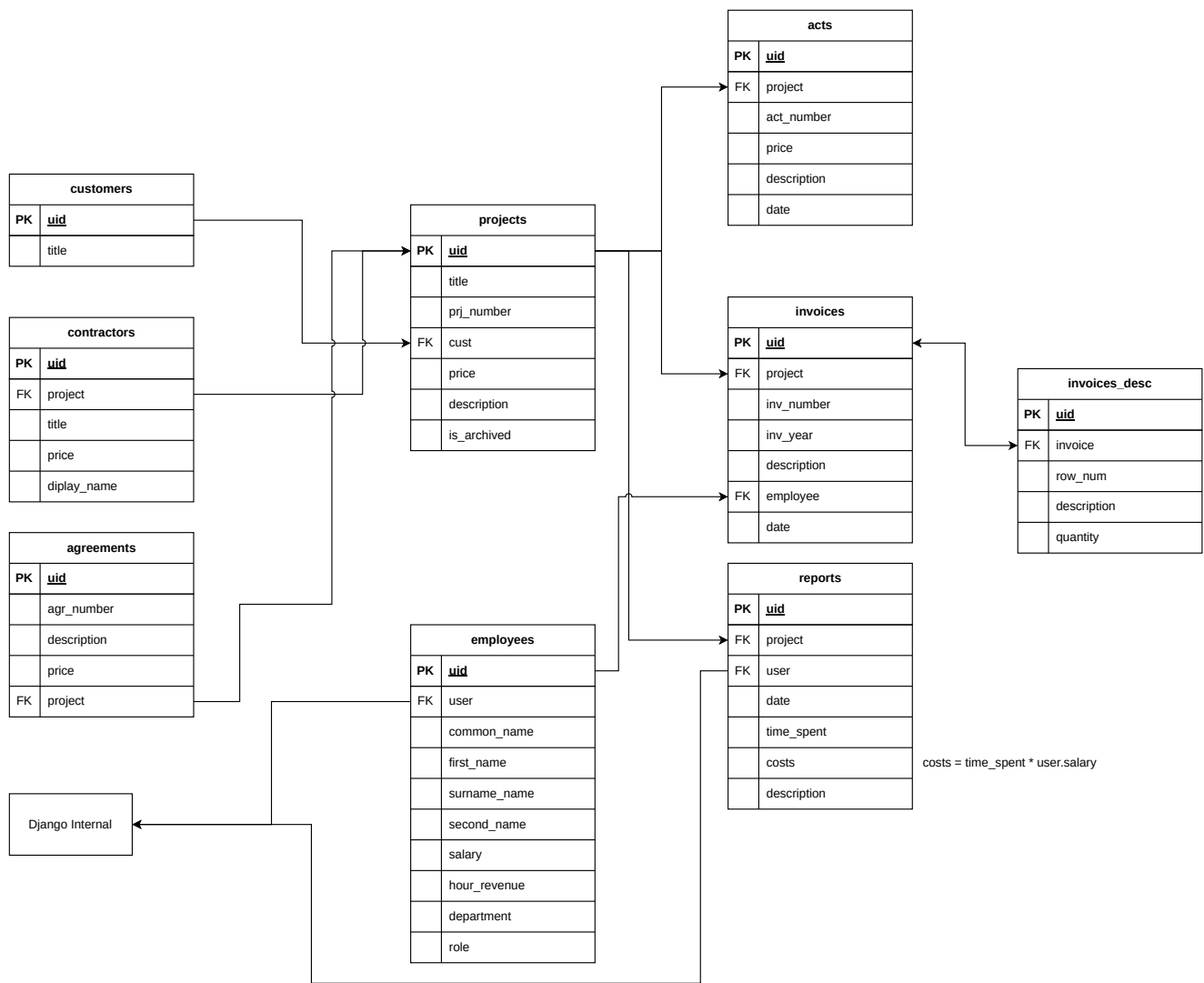
Затем достаточно выполнить:

```
docker compose -f docker-compose.yml up -d --build
```

Docker построит и запустит контейнеры. Сервисы системы будут доступны на тех же портах, что и при "стандартном" развертывании.

Архитектура решения

Архитектура БД



Настройка Django

Настройка соединения с БД

В файле **settings.py** описаны настройки для подключения к СУБД PostgreSQL:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'amg',
        'USER': 'amg',
        'PASSWORD': 'postgres_password',
        'HOST': 'localhost',
        'PORT': '5432',
        # 'OPTIONS': {
        #     'service': 'pg_service',
        #     'passfile': '.pg_pgpass',
        # },
    },
}
```

```
}  
  
}
```

При развороте решения в продуктивной среде эти настройки необходимо изменить на актуальные.

- ENGINE - для СУБД PostgreSQL не изменять
- NAME - имя БД
- USER - имя пользователя, должен быть владельцем БД
- PASSWORD - пароль пользователя
- HOST - адрес сервера с установленной СУБД
- PORT - порт для подключения к БД

Настройка Django

Создание суперпользователя

Учетная запись суперпользователя создается в окружении, где запущен проект. Перед выполнением команды необходимо перейти в директорию, где расположен файл **manage.py** и выполнить:

```
python manage.py createsuperuser
```

Дополнительная настройка

Изменение формулы расчета ставки

Для корректировки формулы расчета ставки в файле **amg_time/models.py** необходимо изменить строку:

```
self.salary = self.salary * 1.68 / 21 / 8
```

на требуемое значение.

 **Важно!**

Данные изменения коснутся только вновь созданных пользователей

Руководство администратора

Добавление пользователей



Для добавления пользователей необходимы права суперпользователя

Добавление пользователей выполняется на странице администрирования

<имя_сервера>:8000 /admin/auth/user/

Пример страницы:

Start typing to filter...

AMG_TIME

Acts + Add

Agreements + Add

Contractors + Add

Customers + Add

Invoice desc's + Add

Invoices + Add

Projects + Add

Reports + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

Select user to change

Q

Search

Action:

 Go 0 of 3 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	admin	-	-	-	✓
<input type="checkbox"/>	ivanov.aa	-	-	-	✗
<input type="checkbox"/>	petrov.pp	-	-	-	✗

3 users

ADD USER +

FILTER

Show counts

By staff status

All

Yes

No

By superuser status

All

Yes

No

By active

All

Yes

No

С помощью кнопки "+ Add" осуществляется переход на страницу создания пользователя

Username:

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

Password-based authentication:

☒ Enabled

☐ Disabled

Whether the user will be able to authenticate using a password or not. If disabled, they may still be able to authenticate using another method.

Password:

Your password can't be too similar to your other personal information.

Your password must contain at least 8 characters.

Your password can't be a commonly used password.

Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

EMPLOYEE

Employee: #1

Фамилия:

Имя:

Отчество:

Отдел:

▼

Роль:

▼

Оклад:

0

▲▼

Ставка (заполняется автоматически):

0

▲▼

SAVE

Save and add another

Save and continue editing

Поля формы:

- Username - имя пользователя в системе, например ivanov.aa
 - Password - пароль пользователя (не менее 8 символов, содержать буквы и цифры)
 - Password Confirmation - подтверждение пароля
 - Фамилия - фамилия сотрудника
 - Имя - имя сотрудника
 - Отчество - отчество сотрудника
 - Отдел - отдел (ОВ/ВК и т.д.)
 - Роль - Доступны роли "Пользователь" и "Админ" (подробнее в [Ролевая модель](#))
 - Оклад - размер вознаграждения
 - Ставка - не требует заполнения, т.к. рассчитывается на основании оклада
- После заполнения формы нужно нажать "SAVE" или, если требуется добавить еще

сотрудника, то "Save and add another".

Ролевая модель

Ролевая модель приложения состоит из 2-х ролей:

- Пользователь
- Админ

Пользователь

Роль обладает следующими полномочиями

Объект	Чтение	Запись
Договора		
Заказчики		
Подрядчики		
Акты		
Накладные	X	X
Отчеты	X	X

Админ

Роль обладает следующими полномочиями

Объект	Чтение	Запись
Договора	X	X
Заказчики	X	X
Подрядчики	X	X
Акты	X	X
Накладные	X	X
Отчеты	X	X

Добавление ролей

При необходимости возможно добавление дополнительных ролей, однако это требует изучения архитектуры решения. Ниже представлена краткая инструкция по модификации решения в части ролевой модели.

Для реализации ролевой модели используется модуль **rest_access_policy** фреймворка django [документация](#)

Правила разграничения доступа размещены в файле **views.py**

Пример правила для роли "Админ"

```
class AdminOnly(AccessPolicy):
    statements = [
        {
            "action": "*",
            "principal": "*",
            "effect": "allow",
            "condition": ["user_must_be:Admin"]
        },
        {
            "action": "*",
            "principal": "*",
            "effect": "deny",
            "condition": ["user_must_be:User"]
        },
    ]

    def user_must_be(self, request, view, action, field: str) -> bool:
        user = User.objects.get(username=request.user)
        return user.employee.role == field
```

Аналогичным образом настроено правило для роли "Пользователь".

Далее, данное правило применяется для отдельных View решения в декораторе `@permission_classes()`, например так:

```
@api_view(["GET", "POST"])
@authentication_classes([JWTAuthentication])
@permission_classes([AdminOnly])
def projects_list(request):
    if request.method == "GET":
        data = Project.objects.all()
        print(f"request {request}")
        serializer = ProjectListSerializer(data, context={"request": request}, many=True)
        return Response(serializer.data)
    elif request.method == "POST":
        print("post >>>", request)
        serializer = ProjectSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Frontend

Frontend построен на ReactJS и обращается по REST-API к django.

Frontend передается в открытом виде и доступен для модификации.

Резервирование

Резервирование решения происходит стандартными средствами БД или ФС.

Troubleshooting

Пользователь авторизован в системе, но не может выполнять операции

Иногда, когда пользователь длительное время не выполняет вход в систему его JWT токен может истечь. В таком случае, чтобы обновить токен пользователя нужно выполнить:

1. Открыть меню разработчика браузера, который использует пользователь
2. Перейти на вкладку "Storage" (или Данные)
3. Выбрать "Local storage" (или Локальное хранилище)
4. Выполнить удаление данных

После этого пользователь сможет снова авторизоваться в системе.

При заполнении формы пользователь получает ошибки django

Некоторые ошибки ввода приводят к ошибкам на backend. Для анализа нужно открыть меню разработчика используемого браузера и с помощью вкладки "Консоль" провести анализ проблемы

Возможные доработки

Ниже перечислены возможные доработки решения для улучшения производительности или удобства использования

- Добавление прокси-сервера
Возможно добавить прокси-сервер для управления доступом и балансировкой нагрузки на frontend (например, nginx)
- Построить frontend
Для исключения возможности несанкционированного доступа к исходному коду решения возможно выполнить `npm run build`. Данная команда выполнить построение решения, но необходимо учитывать, что в таком случае нужно дополнительно настраивать web-сервер
- Кластеризация решения
Для резервирования и увеличения стабильности решение возможно кластеризовать через `docker swarm` или другими способами
- Обработка ошибок
Добавить в код frontend обработку ошибок при заполнении форм