“Внутренний Портал Компании QWEECO”

Содержание:

Установка и запуск проекта “Внутренний Портал Компании QWEECO”

1. Проект располагается в закрытом репозитории компании Qweeco по адресу: *https://bitbucket.org/qweeco/qweecoco*
   - Для установки проекта требуется:
     - Скопировать проект с *bitbucket.org* выполнив команду *git clone https://bitbucket.org/qweeco/qweecoco* **(Требуется git)**
     - Настроить виртуальное окружение (желательно Virtualenvwrapper)
     - Выполнить в консоли инициализацию файла requariment.txt командой *pip install -r requariment.txt.* Данные манипуляции выполнят установку нужных компонентов для корректной работы проекта. Будет установлено:
       - Django==1.9
       - djangorestframework==3.6.2
       - psycopg2==2.7.1
       - django-suit==0.2.25
     - Создать базу данных (в данном проекте используется PostgreSQL)
     - Создать пользователя (по умолчанию используется postgres)
     - Получить привилегии для работы с БД
     - Сделать миграцию для проекта используя команду *python manage.py migrate*
     - Создать суперпользователя для корректной работы с админкой и проектом в целом
     - Загрузить существующие фикстуры (**см. Работа с *Fixtures* п.2**)

Основные модули портала:

1. *Авторизация:*
   - Вход по e-mail и паролю
2. *Сотрудники:*
   - Список сотрудников. Доступно для всех
   - Удаление сотрудника(ов). Доступно только пользователям с определенными ролями

- Редактирование данных сотрудника. Доступно только пользователям с определенными ролями и самому себе
- Блокирование сотрудника(ов). Доступно только пользователям с определенными ролями

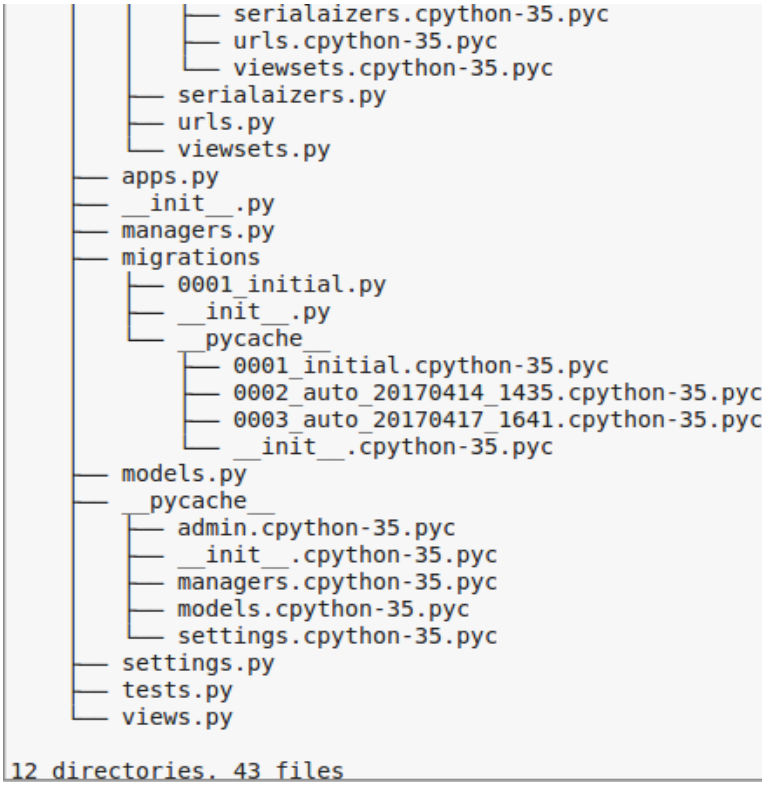
Доступные запросы портала

Доступные запросы	Тип запроса	Возвращаемое значение
Создание ресурса	POST	Вносит изменения в указанный ресурс
Получение ресурса	GET	Отображение запрошенного ресурса
Удаление ресурса	DELETE	Удаляет указанный ресурс
Обновление ресурса	PUT	Загрузка содержимого запроса на указанный в запросе URI
Создание и замена ресурса	PATCH	Загрузка содержимого запроса на указанный фрагмент ресурса

Структура проекта

Схематичное отображение:

```
.
├── manage.py
├── QWEECOCO
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-35.pyc
│   │   ├── settings.cpython-35.pyc
│   │   ├── urls.cpython-35.pyc
│   │   └── wsgi.cpython-35.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── requirements.txt
├── static
│   └── js
│       └── script.js
├── templates
│   ├── base.html
│   ├── index.html
│   └── registration
│       └── login.html
└── worker
    ├── admin.py
    ├── api
    │   ├── __init__.py
    │   ├── permissions.py
    │   ├── __pycache__
    │   │   ├── __init__.cpython-35.pyc
    │   │   ├── permissions.cpython-35.pyc
```



Пояснение структуры:

1. Символ « **.** » - корневая папка проекта
2. ***QWEECOCO*** – папка с проектом
3. ***requirements.txt*** – файл для установки утилит для корректной работы приложений
4. ***static*** – папка статических файлов проекта
   - **js** – папка для хранения JavaScript компонентов
5. ***templates*** – папка для хранения шаблонов проекта
6. ***worker*** – папка приложения сотрудников
   - **api** – папка REST API модуля
   - **migrations** – папка с миграциями(изменениями) проекта

Работа со стандартными компонентами Django

Работа с проектом:

1. *Settings.py – основные настройки над проектом*
   Основные модификации:
   - *NSTALLED_APPS:*
     - *Добавлены приложения: rest_framework, worker, suit*
   - *DATABASES:*
     - *Подключение базы данных PostgreSQL (предварительно установлена lib psycopg2)*
   - *AUTH_USER_MODEL:*

- *Переопределение стандартной модели User*
- *REST_FRAMEWORK:*
  - *Покдлючение* ***DEFAULT_AUTHENTICATION_CLASSES***

2. Urls.py – составление и подключение Urls(маршрутов) проекта
   - Создан url маршрут к админке
   - Создан url маршрут на подключение api информации
   - Создан url маршрут авторизации (login)
   - Создан url маршрут выхода (logout)
   - Создан url маршрут регистрации
3. manage.py – основной компонент управления проектом

Работа с приложением:

1. *admin.py – отвечает за работу админки*
   Основные модификации:
   - UserCreationForm
     - Переопределена форма для создания новых пользователей
   - UserChangeForm
     - Переопределена форма для обновления пользователей
   - WorkerAdmin
     - Переопределена форма для добавления и изменения экземпляров пользователей

2. *models.py – модель связей с БД*
   Основные модификации:
   - Worker
     - Модель работника (переопределённая стандартная модель пользователя)
     - Созданные поля:
       - email — тип поля EmailField (уникальное)
       - first_name - тип поля CharField (макс длина = 30 символов, поле может быть пустым)
       - last_name - тип поля CharField (макс длина = 30 символов, поле может быть пустым)
       - date_joined - тип поля DateTimeField
       - is_active - тип поля BooleanField
       - age - тип поля PositiveIntegerField (по умолчанию = 21 год)
       - start_date_of_work - тип поля DateField
       - role - тип поля CharField (макс длина = 100)
       - middle_name - тип поля CharField (макс длина = 50 символов, поле может быть пустым)
       - is_admin - тип поля BooleanField
   - get_full_name
     - Переопределён стандартный метод получения полного имени, унаследованного класса AbstractBaseUser
   - is_staff
     - Переопределён стандартный метод получения доступа к интерфейсу администратора
   - get_short_name

    - Переопределён стандартный метод получения короткого имени, унаследованного класса AbstractBaseUser
- email_user
    - Метод отправки электронной почты пользователю
- has_perm
    - Переопределён стандартный метод по проверке прав, унаследованного класса PermissionsMixin
- has_module_perms
    - Переопределён стандартный метод по проверке прав, унаследованного класса PermissionsMixin
- USERNAME_FIELD
    - Переопределено имя поля модели, которое используется для уникального идентификатора
- Meta
    - verbose_name, verbose_name_plural
        - Читабельное название модели в единственном и множественном числе

3. settings.py – *содержит всю информацию choices для поля role (models.py – Worker)*
    - *ROLE_DIRECTOR = ROLE_ALL_PRIVILEGES = 'director'*
        - *Роль директора. Имеет все привелегии для модификаций*
    - *ROLE_QA = 'qa'*
        - *Роль инженера-тестировщика*
    - *ROLE_DEVELOPER = 'developer'*
        - *Роль разработчика*
    - *ROLE_OFFICE_MANAGER = 'manager'*
        - *Роль офис-менеджера. Имеет все привелегии для модификаций*
    - *ROLE_DESIGNER = 'designer'*
        - *Роль дизайнера*
    - *ROLES_WITH_ALL_PRIVILEGES = [ROLE_DIRECTOR, ROLE_OFFICE_MANAGER]*
        - *Список дозволенных ролей для модификаций*

4. managers.py – *содержит всю информацию стандартной модели-менеджера пользователя*
    - WorkerManager
        - Модель-менеджер работника (переопределённая стандартная модель-менеджер пользователя)
            - create_user
                - Переопределённый стандартный метод создание пользователя
            - create_superuser
                - Переопределённый стандартный метод создание супер-пользователя

Местонахождение(PATH) данных модулей: Project/worker/

4. scripts – *содержит все JavaScript модули.*
    - *Функция getAllWorkers*

    - *Получение таблицы всех пользователей*
- *Функция deleteWorker*
    - *Удаление пользователя из таблицы*
- *Функция getWorker*
    - *Редактирование пользователя*
- *Функция saveWorker*
    - *Сохранение изменений пользователя*
- *Функция patchWorker*
    - *Блокировка пользователя*

Местонахождение(PATH) данных модулей: Project/static/js/

5. templates – *содержит все шаблоны используемые приложением*
    - login.html
        - *Шаблон авторизации пользователей*
    - base.html
        - *Основной шаблон*
    - index.html
        - *Шаблон стартовой страницы с выводом данных*

Местонахождение(PATH) данных модулей: Project/templates/

Работа с REST API

Для создания портала было задействовано REST API дополнение, которое содержит в себе следующие модули:

1. *Serialaizers*:
    - Сериализатор позволят сложным типам данных типа querysets и экземпляру Django модели быть конвертированым в типы данных Python и затем быть просто преобразованы в JSON, XML или другие типы данных.
    - Сериализация Qweeco портала была выполнена по следующим полям:
        - ***id*** - идентификатор пользователя
        - ***email*** – email-адрес пользователя
        - ***first_name*** – имя пользователя
        - ***last_name*** – фамилия пользователя
        - ***middle_name*** – отчество пользователя
        - ***role*** – роль пользователя
        - ***is_active*** – идентификатор активности пользователя

2. *Permissions*:
    - Permissions(разрешения) вместе с authentication, определяют следует ли получить или оказать в доступе для запроса. Базируется на моделях Django.
    - Разрешения Qweeco портала были выполнены по следующим критериям:
        - Если пользователь соответствует самому себе.
        - Если роль пользователя позволяет произвести запрашиваемое действие.

3. *Viewsets*:
   - Данный модуль содержит основную логику, которая выполняется на проекте. Базируется на (моделях Django, Permissions, Serialaizers, Authentication) Содержит в себе обязательные атрибуты:
     - queryset — принимает модель Django
     - serializer_class — принимает Serialaizers
     - permission_classes — принимает виды Permissions
     - authentication_classes — принимает виды Authentication

Местонахождение(PATH) данных модулей: Project/worker/api/

Работа с Fixtures

Подключение:

1. В файле основных настроек проекта *settings.py* подключена FIXTURE_DIRS, где указан путь до папки в которой будут храниться файлы дампов БД.

Работа с *Fixture:*

1. Перейти в папку с проектом используя команду: *cd /PATH/TO/MY_PROJECT*
2. Выполнить команду на загрузку существующего дампа в БД: *python manage.py loaddata myapp/fixtures/initial_data.json*

Местонахождение(PATH) данных модулей: Project/worker/fixtures/

Этот проект использует следующие технологии:

1. **Python**
2. **Django**
3. **jquery (ajax)**
4. **Bootstrap**
5. **PostgreSQL**
6. **Django-Rest-Framework**