Все взаимодействия с базой данных, происходят путем использования встроенной ORM Django, пример модели:

```
class User(models.Model):
    id = models.BigAutoField(primary_key=True)
    user_id = models.BigIntegerField(unique=True)
    first_name = models.TextField(null=True, blank=True)
    username = models.TextField(null=True, blank=True)
    # newsletter_flag = models.BooleanField(default=True)
    chatgpt_flag = models.BooleanField(default=False)
    created_at = models.DateTimeField()
    updated_at = models.DateTimeField()
    balance = models.FloatField(default=0.0)
    media_flag = models.BooleanField(default=False)
    channel_up = models.BooleanField(default=False)
    # expired_at = models.DateTimeField(null=True, blank=True)
    language = models.TextField(default='ru')
    digest_time = models.TextField(null=True, blank=True)
    similar_news_filter = models.BooleanField(default=False)
    recommendation_flag = models.BooleanField(default=False)
    utm_source = models.CharField(null=True, blank=True)
    ban_date = models.DateTimeField()

    class Meta:
        db_table = '"public"."users"'
        managed = False
```

Для новых моделей обязательно добавляем класс Meta! (чтобы новая таблица не создавалась, т.к. создаём на другом уровне)

[[Требования к бэкэнду]] [[Как запустить проект в первый раз]] [[Документация/Сервисы/Админка/Информация для бэкэнда Админки/Взаимодействие с базой данных]]

Установка проекта

Клонируйте репозиторий и установите его в удобную вам папку. Далее запросите у коллег файл виртуального окружения(.env) для запуска проекта Для запуска установите docker dekstop ## Запуск проекта Проект можно запустить локально на компьютере таким образом: #### 1. Запустить файл compose.yml

В терминале docker desktop прописываем команду `docker-compose up --build`

```
services:
  web_admin:
    env_file:
      - .env
    ports:
      - "8080:8000"
    build:
      context: .
      dockerfile: Dockerfile
    command: "python3 digest_web/manage.py runserver 0.0.0.0:8000"
```

Переходим в терминал запущенного контейнера

1. В терминале переходим в директорию digest_web и применяем миграции

`cd digest_web` `python manage.py migrate`

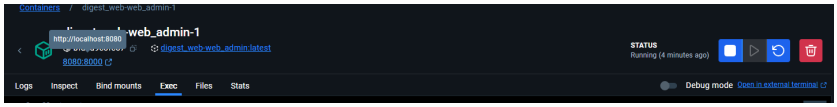
2. Создаём суперпользователя (через него осуществляется вход в админку)

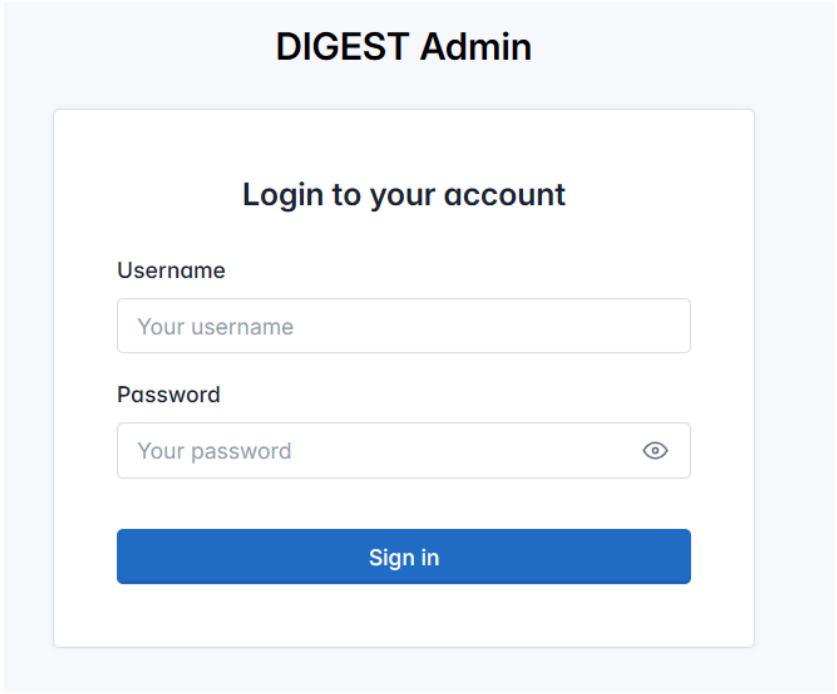
`python manage.py createsuperuser`

```
python: can't open file '/app/manage.py': [Errno 2] No such file or directory
# ls
Dockerfile  compose.yml  digest_web  requirements.txt
# cd digest_web
# ls
digest_web  manage.py  static  webadmin
# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, webadmin
Running migrations:
  No migrations to apply.
# python manage.py createsuperuser
Username (leave blank to use 'root'):
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

2. Переход в панель администратора

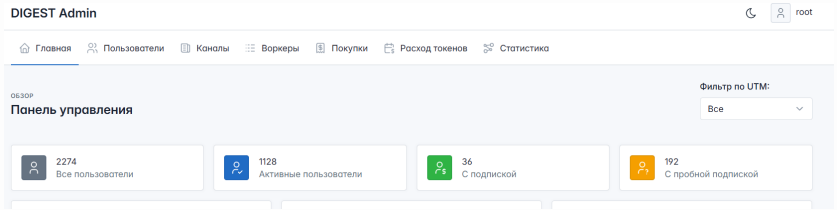
1. Переходим по адресу http://localhost:8080/login/





2. Вводим данные созданного суперпользователя и логинимся

Открывается главная страница админки:



Данные требование необходимо соблюдать в будущем!

Код

1. Названия функций и переменных должно отражать их предназначение.
2. Названия всех функций и методов классов, которые должны быть приватными в пакете/классе должны начинаться либо с `_`, либо с `__`
3. Каждая функция, класс должны иметь документацию в определенном формате. Данный формат должен быть обсужден
4. Наименования функций, классов должно иметь единый стиль в пакете.
5. Не стоит зависеть от общих классов. Исключением может являться объект, являющийся репрезентацией сущности с которой мы работаем.
6. Все функции и методы должны быть покрыты UNIT-тестами. Желательно иметь покрытие 60-80%.
7. Для каждого представления желательно иметь подробное описание взаимодействия с бд и шаблоном.

   ## GIT
8. Все ветки должны иметь в своем названии информацию по типу ветки, краткому описанию и идентификатору задачи в таск менеджеру

- `{prefix}/{name}-{identifier}` Префикс может быть один из следующих:
- `feature`
- `fix`
- `update`

1. Перед выпуском обновления в prod необходимо провести старшему специалисту code review

   # Особенности кода
2. При взаимодействии со временем необходимо переводить UTC время в МСК. По умолчанию в проекте у нас время по МСК
3. При запуске проекта необходимо добавить воркера в базу данных, в ином случае проект запустится, но не сможет обрабатывать каналы