

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и кибербезопасности  
Высшая школа программной инженерии

Курсовая работа  
По дисциплине:  
«Конструирование программного обеспечения»

Выполнили  
Студенты:

Скалисусов К.Г. 5130904/20105  
Кончев М.Д. 5130904/20105  
Крючкова К.Ю. 5130904/20105  
Молотов К.Д. 5130904/20103

Руководитель

Юркин В.А.

# GarageFlow – приложение, позволяющее оптимизировать работу автосервиса и взаимодействие клиента с мастером.

---

## Ссылка на репозиторий:

<https://github.com/mkonchev/CRM-system>

## Этапы

- Определение проблемы
- Выработка требований
- Разработка архитектуры и детальное проектирование
- Кодирование и отладка
- Unit тестирование
- Интеграционное тестирование
- Сборка

## Определение проблемы

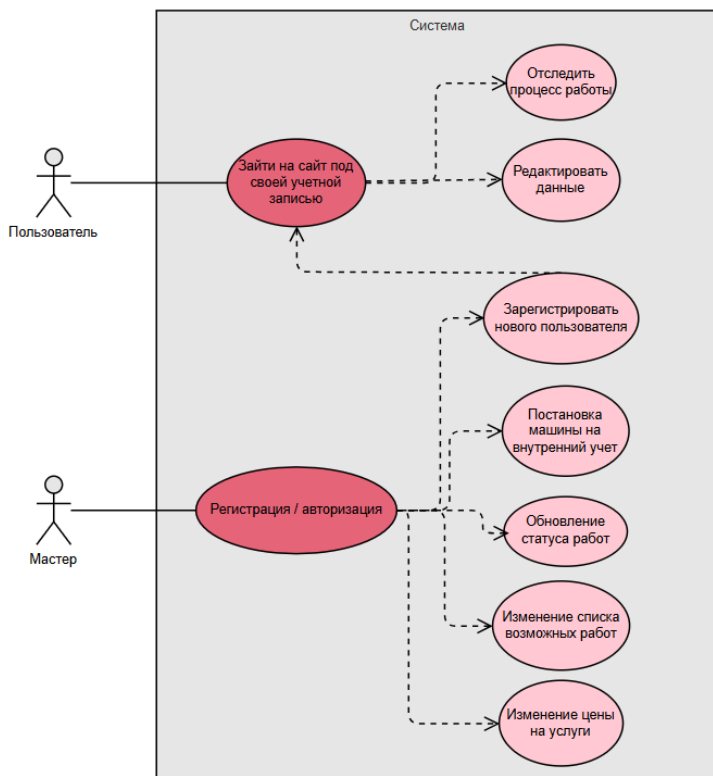
На данный момент многие автосервисы испытывают ряд проблем, связанных с отслеживанием наличия деталей и ведением учета потребителей, также автовладельцы сталкиваются с неудобствами, вызванными отсутствием единой системы, позволяющей отслеживать процесс работы в реальном времени. Во избежание утери информации и возможных последующих конфликтов на почве недопониманий, было принято решение по созданию единой системы для автосервисов и автовладельцев для их упрощенной коммуникации, оптимизации процессов и надежности хранения данных.

## Выработка требований

### Jobs to be done

Сегмент	Автовладелец	Мастер
Цели	Следить за процессом ремонта машины	Оптимизировать процесс работы
Задачи	Отслеживать процесс работы, вовремя забрать машину	Обновлять статус работ по машине
Действия	Зарегистрировался в приложении, отслеживает процесс работы, забирает машину	Зарегистрировался в приложении, поставил машину на внутренний учет, вовремя обновил статус всех работ
Описание	Приложение поможет отслеживать работу мастера в реальном времени	Приложение поможет оптимизировать работу и коммуникацию с клиентом

## UML-диаграмма



## Пользовательские истории

- *От имени пользователя:*

1. Когда моя машина будет на новом этапе ремонта, я зайду в личный кабинет и нажму на кнопку «Подробнее...» в разделе «Текущий ремонт», чтобы узнать об этом и быть уверенным, что меня не обманут о проведенных работах и конечной стоимости ремонта.
2. Когда моя машина будет на последнем этапе ремонта, я зайду в личный кабинет и нажму на кнопку «Подробнее...» в разделе «Текущий ремонт», чтобы спланировать свое время и забрать автомобиль вовремя.

- *От имени мастера:*

1. Когда ко мне приходит новый клиент, я завожу ему его собственный аккаунт с помощью кнопки «Добавить нового пользователя», чтобы во время следующего обращения знать, какие ремонтные работы проводились до этого.
2. Когда цены на детали повышаются, я зайду во вкладку «Прайс-лист» и нажму на кнопку «Редактировать», чтобы изменить цены на свои услуги систематизированно, чтобы не терять время на подсчет итоговой суммы при каждом обращении.
3. Когда я приступаю к осмотру машины, я зайду во вкладку «Прайс-лист» и выберу нужную мне модель машины, нажав на кнопку «Выбрать модель авто», чтобы иметь список возможных работ по конкретной модели, чтобы сразу оценить свои возможности и огласить цену клиенту.

## Оценка числа пользователей

Исходя из анализа посещаемости сайтов потенциальных конкурентов, была составлена высокоуровневая оценка числа пользователей разрабатываемого сервиса

1. Оценка числа пользователей
  - Среднесуточная активность: 10 000 пользователей
  - Месячная аудитория: 300 000 уникальных пользователей
  - Количество обращений: 25 000 обращений в сутки
2. Период хранения информации
  - Оптимальный срок хранения: 7-10 лет
  - Для логинов: бессрочно, с архивацией

## Разработка архитектуры и детальное проектирование

### Технологический стек

- Бэкенд: Python (Django), DRF
- Фронтенд: Python (Django)
- СУБД: PostgreSQL
- Инфраструктура: Docker
- Инструменты разработки: Git
- Внешняя зависимость: <https://vpic.nhtsa.dot.gov/api/>

### Характер нагрузки на сервис

- Соотношение R/W нагрузки
  - Нагрузка записей: 30%; максимальная загрузка системы 50 WPS (writes per seconds)
  - Нагрузка чтения: 70%; максимальная загрузка системы 60 RPS (reads per seconds)
- Объемы трафика
  - Средний объем передаваемых данных — 256 Мб/час, пиковый — 512 Мб/час
  - Входящий трафик — 200 Мбит/с, исходящий — 100 Мбит/с
- Объемы дисковой системы
  - Начальный объем данных 10 Гб
  - Рекомендуемый объем хранилища с расчетом на 3 года: 100 Гб

### Диаграммы

Диаграмма по стандарту c4 (Абстракции)



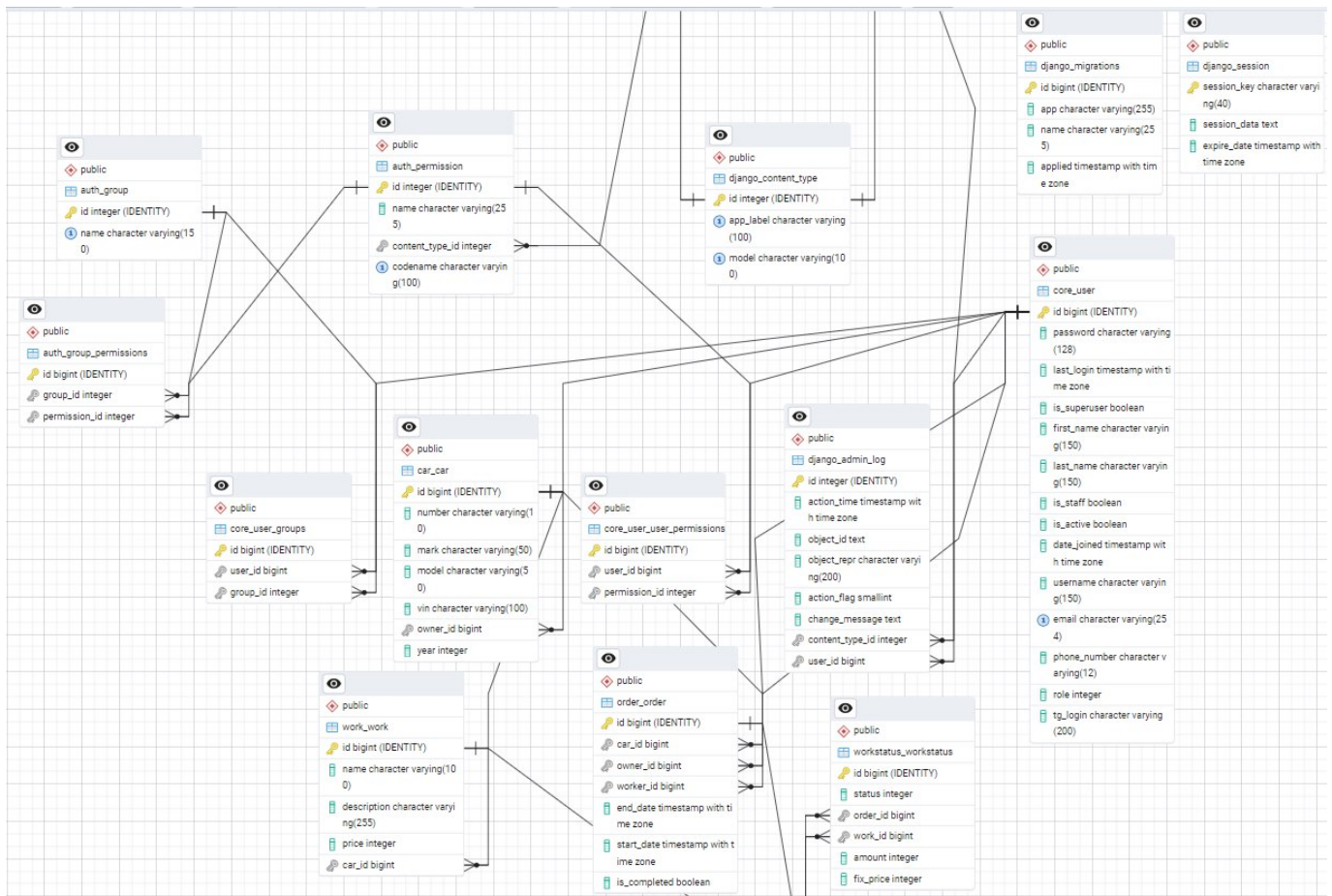
## Контракты API

- Эндпоинты (URL, методы: GET/POST)
- Параметры запроса и ответа (обязательные поля)
- Коды ответов (200 OK, 400 Bad Request, 500 Server Error)
- Авторизация (токены, API-ключи)

### Нефункциональные требования на время отклика

- Среднее время ответа API:  $\leq 200$  мс (для 95% запросов)
- Максимальное время ответа в пиковой нагрузке:  $\leq 500$  мс
- Допустимая задержка при 99-м перцентиле (p99):  $\leq 1$  сек

## Схема базы данных



### Обоснование выдержки нефункциональных требований

1. Индексы (PRIMARY KEY, INDEX): быстрый поиск.
2. Нормализация (нет дублирования данных): экономия места и целостность.
3. Партиционирование (если таблица большая, можно разбить по дате).
4. Репликация (читающие реплики для балансировки нагрузки).

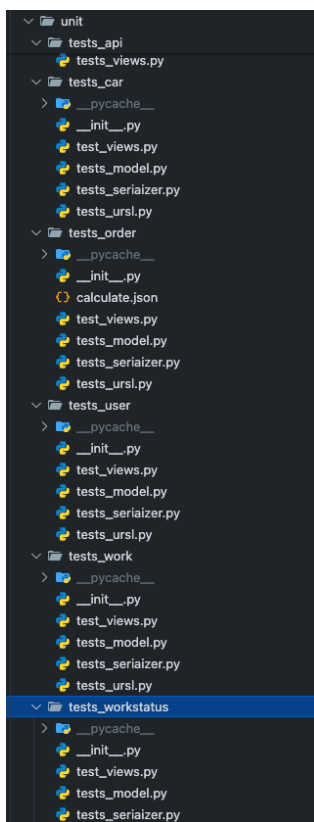
## Схема масштабирования сервиса

Для масштабирования сервиса будет использовано горизонтальное масштабирование

- Для базы данных используем шардирование и кэш для разгрузки
- Для Python приложения разделяем на микросервисы, каждый из которых масштабируется независимо
- Для асинхронной обработки запросов используем Kafka
- Для отслеживания нагрузки на систему может быть использована locust

Тестирование:

Unit тестирование:



Запуск unit тестов происходит при помощи команды

Make build-unit-test:

Или

docker exec backend python manage.py test apps.api.tests.unit

```

docker exec backend python manage.py test apps.api.tests.unit
Creating test database for alias 'default'...
Found 95 test(s).
System check identified some issues:

WARNINGS:
?: (staticfiles.W004) The directory '/app/apps/static' in the STATICFILES_DIRS setting does not exist.

System check identified 1 issue (0 silenced).
.....

Ran 95 tests in 0.919s

OK
Destroying test database for alias 'default'...

```

## Покрытие unit тестов (97%)

Name	Stmts	Miss	Cover
apps/__init__.py	0	0	100%
apps/api/__init__.py	0	0	100%
apps/api/serializers/CarSerializer.py	6	0	100%
apps/api/serializers/OrderSerializer.py	6	0	100%
apps/api/serializers/UserSerializer.py	6	0	100%
apps/api/serializers/WorkSerializer.py	6	0	100%
apps/api/serializers/WorkstatusSerializer.py	6	0	100%
apps/api/tests/__init__.py	0	0	100%
apps/api/tests/factories.py	43	0	100%
apps/api/tests/integration/__init__.py	0	0	100%
apps/api/tests/integration/test_cars_integration.py	53	0	100%
apps/api/tests/integration/test_orders_integration.py	51	0	100%
apps/api/tests/integration/test_users_integration.py	61	0	100%
apps/api/tests/integration/test_works_integration.py	30	0	100%
apps/api/tests/integration/test_workstatus_integration.py	27	0	100%
apps/api/tests/unit/__init__.py	0	0	100%
apps/api/tests/unit/tests_api/__init__.py	0	0	100%
apps/api/tests/unit/tests_api/tests_views.py	15	0	100%
apps/api/tests/unit/tests_car/__init__.py	0	0	100%
apps/api/tests/unit/tests_car/test_views.py	47	0	100%
apps/api/tests/unit/tests_car/tests_model.py	27	0	100%
apps/api/tests/unit/tests_car/tests_serializer.py	30	0	100%
apps/api/tests/unit/tests_car/tests_ursl.py	19	0	100%
apps/api/tests/unit/tests_order/__init__.py	0	0	100%
apps/api/tests/unit/tests_order/test_views.py	51	0	100%
apps/api/tests/unit/tests_order/tests_model.py	36	0	100%
apps/api/tests/unit/tests_order/tests_serializer.py	45	0	100%
apps/api/tests/unit/tests_order/tests_ursl.py	19	0	100%
apps/api/tests/unit/tests_user/__init__.py	0	0	100%
apps/api/tests/unit/tests_user/test_views.py	46	0	100%
apps/api/tests/unit/tests_user/tests_model.py	12	0	100%
apps/api/tests/unit/tests_user/tests_serializer.py	32	0	100%
apps/api/tests/unit/tests_user/tests_ursl.py	19	0	100%
apps/api/tests/unit/tests_work/__init__.py	0	0	100%
apps/api/tests/unit/tests_work/test_views.py	49	0	100%
apps/api/tests/unit/tests_work/tests_model.py	25	0	100%
apps/api/tests/unit/tests_work/tests_serializer.py	35	0	100%
apps/api/tests/unit/tests_work/tests_ursl.py	19	0	100%
apps/api/tests/unit/tests_workstatus/__init__.py	0	0	100%
apps/api/tests/unit/tests_workstatus/test_views.py	49	0	100%
apps/api/tests/unit/tests_workstatus/tests_model.py	22	0	100%
apps/api/tests/unit/tests_workstatus/tests_serializer.py	38	0	100%
apps/api/tests/unit/tests_workstatus/tests_ursl.py	19	0	100%
apps/api/urls/__init__.py	0	0	100%
apps/api/urls/car_urls.py	3	0	100%
apps/api/urls/common.py	4	0	100%
apps/api/urls/order_urls.py	3	0	100%
apps/api/urls/user_urls.py	3	0	100%
apps/api/urls/work_urls.py	3	0	100%
apps/api/urls/workstatus_urls.py	3	0	100%
apps/api/views/__init__.py	0	0	100%
apps/api/views/api_view.py	7	0	100%
apps/api/views/car_views.py	44	2	95%
apps/api/views/order_views.py	37	2	95%
apps/api/views/user_views.py	43	1	98%
apps/api/views/work_views.py	43	1	98%
apps/api/views/workstatus_views.py	43	4	91%
apps/car/__init__.py	0	0	100%
apps/car/admin.py	4	0	100%
apps/car/apps.py	5	0	100%
apps/car/migrations/0001_initial.py	7	0	100%
apps/car/migrations/0002_alter_car_owner.py	6	0	100%
apps/car/migrations/0003_alter_car_owner.py	6	0	100%
apps/car/migrations/0004_car_year_alter_car_number_alter_car_owner.py	6	0	100%
apps/car/migrations/0005_alter_car_year.py	4	0	100%
apps/car/migrations/0006_alter_car_year.py	4	0	100%
apps/car/migrations/0007_alter_car_year.py	4	0	100%



apps/car/migrations/0009_alter_car_year.py	4	0	100%
apps/car/migrations/0010_remove_car_year.py	4	0	100%
apps/car/migrations/0011_car_year.py	4	0	100%
apps/car/migrations/0012_alter_car_owner.py	6	0	100%
apps/car/migrations/0013_alter_car_mark_alter_car_model_alter_car_vin.py	4	0	100%
apps/car/migrations/__init__.py	0	0	100%
apps/car/models/CarModel.py	25	5	80%
apps/car/models/__init__.py	4	0	100%
apps/car/models/admins/CarModelAdmin.py	4	0	100%
apps/car/models/admins/__init__.py	0	0	100%
apps/car/services/VINDecoder.py	22	14	36%
apps/car/services/__init__.py	0	0	100%
apps/core/__init__.py	0	0	100%
apps/core/admin.py	11	1	91%
apps/core/apps.py	4	0	100%
apps/core/migrations/0001_initial.py	7	0	100%
apps/core/migrations/0002_user_phone_number_user_role_user_tg_login.py	4	0	100%
apps/core/migrations/0003_alter_user_phone_number.py	4	0	100%
apps/core/migrations/0004_alter_user_phone_number.py	4	0	100%
apps/core/migrations/__init__.py	0	0	100%
apps/core/models/UserModel.py	15	0	100%
apps/core/models/__init__.py	4	0	100%
apps/core/models/admins/UserModelAdmin.py	8	0	100%
apps/core/models/admins/__init__.py	0	0	100%
apps/core/models/consts.py	5	0	100%
apps/core/models/managers/UserManager.py	20	14	30%
apps/core/models/managers/__init__.py	0	0	100%
apps/order/__init__.py	0	0	100%
apps/order/admin.py	4	0	100%
apps/order/apps.py	7	0	100%
apps/order/migrations/0001_initial.py	7	0	100%
apps/order/migrations/0002_alter_order_owner.py	6	0	100%
apps/order/migrations/0003_alter_order_owner.py	6	0	100%
apps/order/migrations/0004_order_end_date_order_start_date.py	4	0	100%
apps/order/migrations/0005_alter_order_worker.py	6	0	100%
apps/order/migrations/0006_alter_order_owner_alter_order_worker.py	6	0	100%
apps/order/migrations/0007_alter_order_car_alter_order_owner_alter_order_worker.py	6	0	100%
apps/order/migrations/0008_order_works.py	4	0	100%
apps/order/migrations/0009_alter_order_car_alter_order_owner_alter_order_worker.py	6	0	100%
apps/order/migrations/0010_alter_order_works.py	4	0	100%
apps/order/migrations/0011_order_workstatus_alter_order_works.py	4	0	100%
apps/order/migrations/0012_alter_order_workstatus.py	4	0	100%
apps/order/migrations/0013_remove_order_workstatus.py	4	0	100%
apps/order/migrations/0014_order_workstatuses.py	4	0	100%
apps/order/migrations/0015_rename_workstatuses_order_workstatus.py	4	0	100%
apps/order/migrations/0016_remove_order_works_remove_order_workstatus.py	4	0	100%
apps/order/migrations/0017_order_is_completed.py	4	0	100%
apps/order/migrations/0018_alter_order_start_date.py	4	0	100%
apps/order/migrations/__init__.py	0	0	100%
apps/order/models/OrderModel.py	24	6	75%
apps/order/models/__init__.py	4	0	100%
apps/order/models/admins/OrderModelAdmin.py	12	0	100%
apps/order/models/admins/__init__.py	0	0	100%
apps/order/signals.py	6	0	100%
apps/work/__init__.py	0	0	100%
apps/work/admin.py	4	0	100%
apps/work/apps.py	5	0	100%
apps/work/migrations/0001_initial.py	5	0	100%
apps/work/migrations/0002_work_car.py	5	0	100%
apps/work/migrations/0003_remove_work_status.py	4	0	100%
apps/work/migrations/__init__.py	0	0	100%
apps/work/models/WorkModel.py	14	0	100%
apps/work/models/__init__.py	4	0	100%
apps/work/models/admins/WorkModelAdmin.py	4	0	100%
apps/work/models/admins/__init__.py	3	0	100%
apps/workstatus/__init__.py	0	0	100%
apps/workstatus/admin.py	4	0	100%
apps/workstatus/apps.py	7	0	100%
apps/workstatus/migrations/0001_initial.py	6	0	100%
apps/workstatus/migrations/0002_alter_workstatus_order.py	5	0	100%

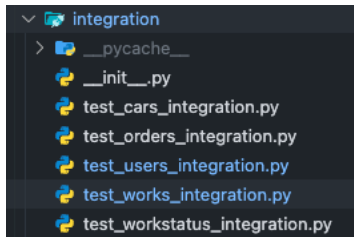
apps/workstatus/migrations/0002_alter_workstatus_order.py	5	0	100%
apps/workstatus/migrations/0003_alter_workstatus_options.py	4	0	100%
apps/workstatus/migrations/0004_alter_workstatus_work.py	5	0	100%
apps/workstatus/migrations/0005_alter_workstatus_work.py	5	0	100%
apps/workstatus/migrations/0006_workstatus_amount_workstatus_fix_price_and_more.py	5	0	100%
apps/workstatus/migrations/0007_alter_workstatus_fix_price.py	4	0	100%
apps/workstatus/migrations/__init__.py	0	0	100%
apps/workstatus/models/WorkstatusModel.py	22	2	91%
apps/workstatus/models/__init__.py	4	0	100%
apps/workstatus/models/admins/WorkstatusModelAdmin.py	5	0	100%
apps/workstatus/models/admins/__init__.py	3	0	100%
apps/workstatus/models/consts.py	5	0	100%
apps/workstatus/signals.py	9	0	100%
config/__init__.py	2	0	100%
config/celery.py	6	0	100%
config/constants.py	1	0	100%
config/settings/__init__.py	0	0	100%
config/settings/common.py	26	0	100%
config/settings/dev.py	14	2	86%
config/urls/__init__.py	0	0	100%
config/urls/common.py	6	0	100%
config/urls/dev.py	3	0	100%
manage.py	9	2	78%

TOTAL	1750	56	97%
-------	------	----	-----



## Интеграционное тестирование:

Для каждой из моделей был реализован полный жизненный цикл работы



Запуск интеграционных тестов выполняется при помощи команд:

Make build-integration-test

или

docker exec backend python manage.py test apps.api.tests.integration

```
docker exec backend python manage.py test apps.api.tests.integration
Creating test database for alias 'default'...
Found 19 test(s).
System check identified some issues:

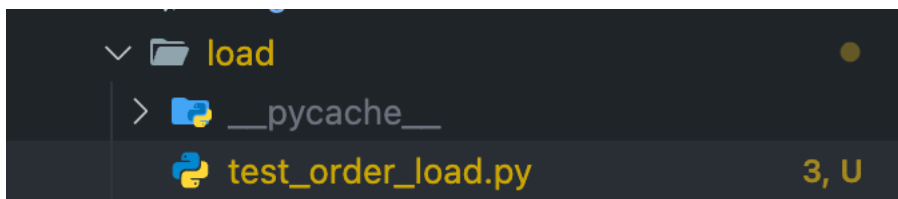
WARNINGS:
?: (staticfiles.W004) The directory '/app/apps/static' in the STATICFILES_DIRS setting does not exist.

System check identified 1 issue (0 silenced).
.....
-----
Ran 19 tests in 1.781s

OK
Destroying test database for alias 'default'...
```

## Нагрузочное тестирование:

Для реализации нагрузочного тестирования используется инструмент locust. Для нагрузочного тестирования был написан скрипт, который создает пользователей, получает список всех пользователей, изменяет данные у пользователя и удаляет пользователя.




Запуск тестирования:

locust -f dct/apps/api/tests/load/test\_order\_load.py --host=http://localhost:8000

Запуск нагрузочного тестирования:

Окно с указанием количества пользователей и запросы в секунду:




Host  
http://localhost:8000

Status  
READY

RPS  
0

Failures  
0%



Start new load test

Number of users (peak concurrency) \*

300

Ramp up (users started/second) \*

500

Host


http://localhost:8000

Advanced options

▼

START

Нагрузочный тест:



Host  
http://localhost:8000

Status  
RUNNING

Users  
300


RPS  
21.83

Failures  
9%

EDIT

STOP

RESET



STATISTICS

CHARTS

FAILURES

EXCEPTIONS

CURRENT RATIO

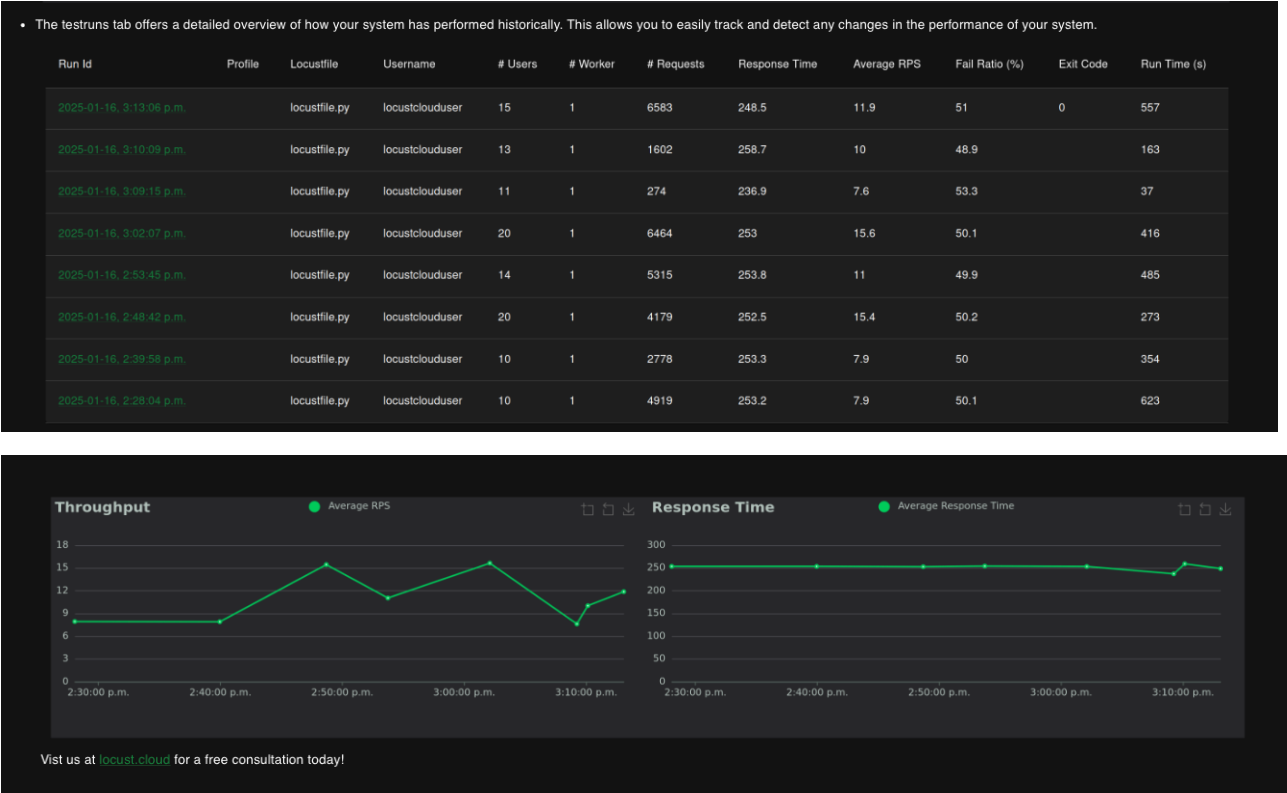
DOWNLOAD DATA

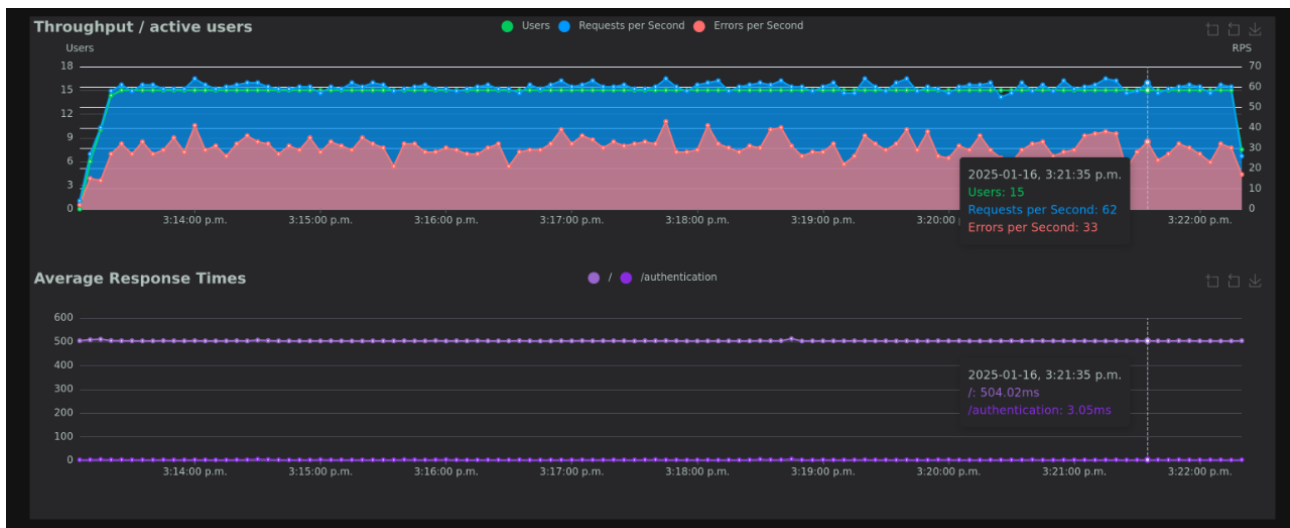
LOGS

LOCUST CLOUD

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/user/5616	1	0	146.84	150	150	146.84	147	147	351	0.17	0
DELETE	/api/user/5616/delete	1	0	464.49	460	460	464.49	464	464	0	0.17	0
DELETE	/api/user/5618/delete	1	0	1025.37	1000	1000	1025.37	1025	1025	0	0.17	0
PATCH	/api/user/5622/update	1	1	15.17	15	15	15.17	15	15	56	0.17	0.17
GET	/api/user/5623	1	0	499	500	500	499	499	499	341	0.17	0

Полученные результаты:





По полученным результатам можно сделать следующий вывод, что система обрабатывает в среднем 62 запроса в секунду.

Что можно сделать для улучшения результатов:

#### 1. Оптимизация серверной части

- Кэширование данных (Redis, Memcached) для снижения нагрузки на базу данных.
- Асинхронная обработка задач (Celery + RabbitMQ/Redis) для выноса тяжелых операций в фоновые процессы.
- Оптимизация SQL-запросов (индексы, исключение N+1 проблем через `select_related` и `prefetch_related` в Django).
- Использование более эффективных сериализаторов (например, `orjson` вместо стандартного `json`).
- Сжатие ответов API (Gzip, Brotli).

#### 2. Масштабирование инфраструктуры

- Балансировка нагрузки (Nginx, Traefik, HAProxy) для распределения запросов между несколькими серверами.
- Использование Kubernetes (K8s) для автоматического масштабирования под нагрузкой.

#### 3. Оптимизация базы данных

- Репликация БД (Master-Slave) для распределения read-запросов.
- Шардинг (разделение данных по серверам).
- Настройка пула соединений (PgBouncer для PostgreSQL).

#### 5. Мониторинг и профилирование

- Логирование медленных запросов (Django Debug Toolbar, Sentry).
- Профилирование кода (cProfile, Py-Spy).
- Нагрузочное тестирование (Locust, JMeter) для выявления узких мест.

Ожидаемый результат после оптимизации:

При грамотной настройке можно добиться увеличения RPS в 2-5 раз (до 120-300 запросов/сек), в зависимости от конкретных узких мест в системе.