



**Частное учреждение профессионального образования
«Высшая школа предпринимательства»
(ЧУПО «ВШП»)**

ДНЕВНИК ПРАКТИКИ

Вид практики (учебная, производственная, преддипломная): _____

Установленный по КУГ срок прохождения практики: с _____.20__г. по _____.20__г.

Место прохождения практики (наименование организации): _____

Выполнил студент
____-го курса

(подпись)

(фамилия, имя, отчество)

Руководитель от
образовательной
организации

(подпись)

(ученая степень, фамилия, имя, отчество)

(должность)

Руководитель от
предприятия

(подпись)

(ученая степень, фамилия, имя, отчество)

(должность)

Тверь, 2025 г.

Содержание

Введение	3
1. Проектирование пользовательского интерфейса (UI/UX) в Figma	4
2. Тестирование функционала и нагрузочное тестирование.....	6
3. Деплой приложения на хостинг ISPManager	8
4. Настройка безопасности веб-приложения	10
Заключение	11
Список источников	12
Приложение 1. Макет в Figma	13
Приложение 2. Антиплагиат.....	13
Приложение 3. QR-код репозитория	13

Введение

Производственная практика является неотъемлемой частью учебного процесса, позволяющей студентам применить теоретические знания в реальных условиях.

В ходе практики я занимался разработкой веб-приложения для онлайн-школы водителей, которое предназначено для организации дистанционного обучения, управления курсами и взаимодействия с пользователями.

Актуальность проекта

В условиях роста числа дорожно-транспортных происшествий, связанных с человеческим фактором, качественная подготовка водителей становится критически важной. Традиционные методы обучения часто устаревают и не успевают адаптироваться к быстро меняющимся требованиям ПДД и технологиям. Веб-приложение для онлайн-обучения водителей решает эти проблемы

Сроки прохождения практики:

Практика проходила с 24 марта по 20 апреля 2025 года.

Цель практики:

Создание функционального веб-приложения для обучения водителей, включающего систему регистрации, тестирование и личный кабинет.

Задачи практики:

1. Проектирование пользовательского интерфейса (UI/UX) в Figma.
2. Тестирование функционала и нагрузочное тестирование.
3. Деплой приложения на хостинг ISPManager.
4. Настройка безопасности и резервного копирования.

Место прохождения практики: Практика проводилась в ООО «Автопросвещение-Тверь». Моя роль заключалась в позиции веб-разработчика.

1. Проектирование пользовательского интерфейса (UI/UX) в Figma

Проектирование пользовательского интерфейса (UI/UX) — это ключевой этап в создании любого современного веб-приложения. Именно на этом этапе формируется внешний вид продукта, его структура, логика взаимодействия и общее впечатление пользователя. Для выполнения этих задач сегодня одним из самых мощных инструментов является Figma.

Что такое UI и UX?

- **UI (User Interface)** — это визуальная часть продукта: кнопки, поля ввода, меню, цвета, шрифты и другие элементы, с которыми взаимодействует пользователь.
- **UX (User Experience)** — это опыт пользователя при работе с продуктом, включая удобство, простоту навигации, скорость выполнения задач и общее удовлетворение от использования. [1]

Эффективное проектирование UI/UX позволяет создавать продукты, которыми приятно и просто пользоваться, что напрямую влияет на их успех и востребованность.

Почему Figma?

Figma — это облачная платформа для проектирования интерфейсов, которая объединила в себе инструменты для создания wireframes, прототипов, финальных макетов и даже совместной работы в реальном времени. До появления Figma процесс проектирования часто требовал использования нескольких программ (например, Balsamiq для wireframes, Sketch для макетов, inVision для прототипирования), что усложняло и замедляло работу. Figma позволила выполнять все этапы в одном месте, обеспечив прозрачность, скорость и удобство командной работы. [1]

Преимущества Figma:

- Совместная работа в реальном времени (несколько человек могут редактировать один макет одновременно)

- Простота обмена файлами и получения обратной связи
- Возможность создавать и использовать дизайн-системы и компоненты для единообразия интерфейса
- Интеграция с инструментами для разработчиков (Dev Mode)
- Мощные инструменты для прототипирования и анимации

Основные этапы проектирования UI/UX в Figma 1. Исследование и анализ
Перед началом проектирования важно изучить целевую аудиторию, её потребности, привычки и проблемы. На этом этапе проводится анализ конкурентов и сбор требований к продукту. [1]

2. Вайрфрейминг (Wireframing)

Создание низкоуровневых схем (wireframes), которые отражают структуру будущего интерфейса без детализации дизайна. Это помогает быстро согласовать основные сценарии и расположение элементов. [1]

3. Проектирование макетов (Visual Design)

На этом этапе разрабатываются финальные визуальные решения: цветовая палитра, типографика, иконки, изображения и другие элементы фирменного стиля. В Figma удобно использовать готовые библиотеки компонентов и дизайн-системы, что обеспечивает единообразие интерфейса на всех страницах. [1]

4. Прототипирование и добавление интерактивности

Figma позволяет создавать интерактивные прототипы, имитирующие поведение настоящего приложения. Можно задавать переходы между экранами, анимировать элементы, реализовывать различные сценарии взаимодействия (например, наведение курсора, нажатие кнопок). [1]

После всех этапов такой разработки, был разработан такой дизайн в Figma, который к сожалению не поместился в эту страницу. см. Приложение 1

2. Тестирование функционала и нагрузочное тестирование

- **Функциональное тестирование:**

Обеспечивает корректную работу всех функций веб-приложения: регистрация, тестирование, личный кабинет. Проверяется работоспособность ссылок, форм, сценариев использования, выявляются и устраняются ошибки на ранних этапах. [2]

- **Нагрузочное тестирование:**

Метод оценки производительности приложения под реальной или ожидаемой нагрузкой. Позволяет выявить узкие места, проверить стабильность и устойчивость при большом числе пользователей, что критично для веб-приложения с возможным массовым использованием. [3]

- **Этапы нагрузочного тестирования:**

- Определение целей и метрик (время отклика, ошибки, нагрузка на процессор и память).
- Создание сценариев нагрузки, имитирующих поведение пользователей.
- Настройка тестовой среды, максимально приближенной к рабочей.
- Проведение тестов, сбор и анализ результатов.
- Оптимизация и повторное тестирование. [3]

В ходе разработки было написано нагрузочное тестирование и тестирование функционала. Ниже приведен пример кода тестирования функциональности:

```
# tests/test_api.py
from django.test import TestCase
from rest_framework.test import APIClient
from rest_framework import status
from django.urls import reverse
from .models import Task

class TaskAPITestCase(TestCase):
    def setUp(self):
        self.client = APIClient()
        self.task_data = {'title': 'Тестовая задача', 'completed':
False}
        self.task = Task.objects.create(**self.task_data)

    def test_get_tasks_list(self):
```

```

url = reverse('task-list')
response = self.client.get(url)
self.assertEqual(response.status_code, status.HTTP_200_OK)
self.assertContains(response, self.task.title) # Проверка
содержимого

def test_create_task(self):
    url = reverse('task-list')
    response = self.client.post(url, self.task_data,
format='json')
    self.assertEqual(response.status_code,
status.HTTP_201_CREATED)
    self.assertEqual(Task.objects.count(), 2)

```

Ключевые аспекты:

1. `APIClient` имитирует HTTP-запросы к API
2. `reverse()` получает URL по имени роута
3. Проверка статусов (200, 201) и содержимого ответа

Мой код нагрузочного тестирования:

```

# locustfile.py
from locust import HttpUser, task, between

class DjangoLoadUser(HttpUser):
    wait_time = between(1, 3)

    @task
    def load_tasks(self):
        self.client.get("/api/tasks")
        self.client.post("/api/tasks", json={
            "title": "Load Test Task",
            "completed": False
        })

```

Для комплексного подхода рекомендуется сочетать:

1. Модульные тесты (проверка отдельных компонентов)
2. Интеграционные тесты (взаимодействие систем)
3. E2E-тесты (полные пользовательские сценарии)
4. Нагрузочные тесты (имитация реальной нагрузки)

3. Деплой приложения на хостинг ISPManager

Деплой Django-приложения на хостинг с панелью ISPManager включает несколько важных этапов, позволяющих корректно запустить и настроить веб-приложение в готовой серверной среде.

Основные этапы деплоя Django-приложения в ISPManager

1. Подготовка окружения и установка Python

В панели ISPManager необходимо включить поддержку Python и установить нужную версию (например, Python 3.11). Это делается в разделе «Настройки» → «Конфигурация ПО», где выбирается и устанавливается нужное ПО, включая Python и веб-сервер (обычно nginx). [4]

2. Создание сайта и настройка домена

В панели создаётся новый сайт, к которому привязывается доменное имя. Важно, чтобы домен был корректно настроен и указывал на IP сервера. В настройках сайта нужно включить поддержку CGI-скриптов или Python-приложений. [4]

После этого перейдем к следующему этапу

1. Настройка обработчика Python в ISPManager

В разделе «Обработчик (Python)» панели ISPManager укажите:

- Режим работы приложения — web-сервер Python
- Путь к серверу — полный путь к файлу `manage.py` нашего проекта, например: `/home/username/private/app/myapp/manage.py`
- Аргументы сервера — команда запуска, например: `runserver 8000` (порт нужно указать, он будет назначен при создании сайта)
- Выберите версию Python, соответствующую нашему окружению (например, 3.11). [4]

2. Запуск и тестирование приложения

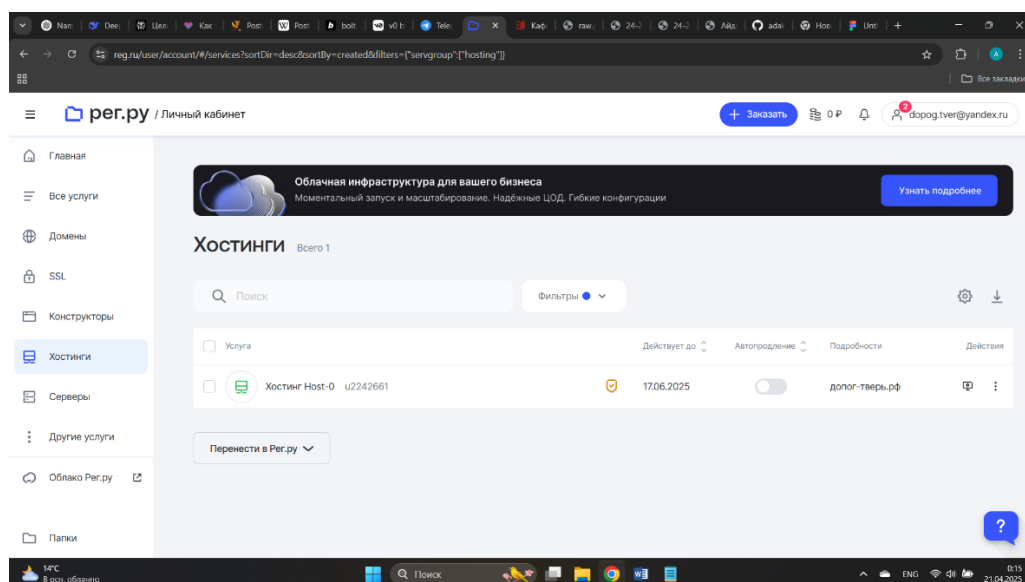
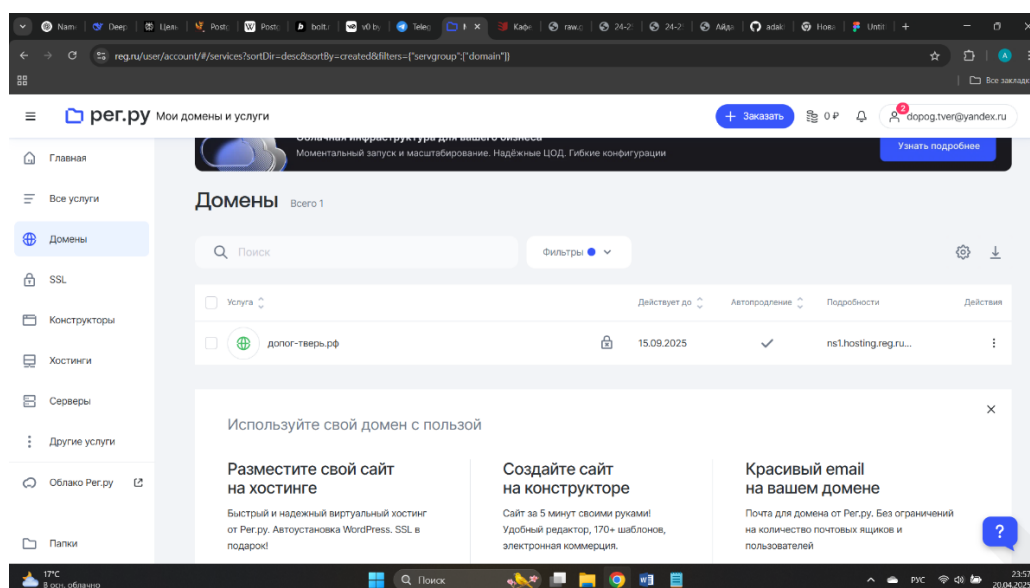
После настройки обработчика запустим приложение через панель или командой.

Убедимся, что сайт доступен по доменному имени и корректно обрабатывает запросы.

3. Настройка безопасности и дополнительных параметров

- Включение HTTPS с помощью SSL-сертификата (Let's Encrypt или другой).
- Настройка прав доступа к файлам и папкам.
- При необходимости настраивание автоматического запуска приложения через supervisor или systemd.
- Проверка логов сервера и приложения для выявления ошибок. [4]

В итоге у нас отдельный проект с доменом и окружением в ISPManager



4. Настройка безопасности веб-приложения

Безопасность веб-приложения — это комплекс мероприятий, направленных на защиту данных пользователей и самого приложения от различных видов атак и уязвимостей. Django предоставляет широкий набор встроенных механизмов и рекомендаций для обеспечения высокого уровня безопасности.

Основные меры безопасности в Django

1. Защита от межсайтовых скриптов (XSS)

Django автоматически экранирует опасные символы в шаблонах, что предотвращает внедрение вредоносных скриптов через пользовательский ввод. Однако важно избегать использования небезопасных конструкций, таких как `|safe` без необходимости, и внимательно работать с HTML, хранящимся в базе данных. [5]

2. Защита от подделки межсайтовых запросов (CSRF)

Django включает встроенную защиту от CSRF-атак с помощью middleware и шаблонного тега `{% csrf_token %}` в формах. Это предотвращает выполнение нежелательных действий от имени пользователя без его ведома. [5]

3. Защита от SQL-инъекций

Django ORM использует параметризацию запросов, что исключает возможность внедрения вредоносного SQL-кода через пользовательские данные.

Разработчикам рекомендуется избегать прямого написания SQL-запросов и использовать ORM или тщательно экранировать параметры. [5]

4. Защита от кликджекинга

Django содержит middleware `X-Frame-Options`, который запрещает отображение страниц приложения внутри `<iframe>` на сторонних сайтах, что предотвращает атаки кликджекинга. [5]

В фреймворке Django уже предусмотрены все меры безопасности, в таком случае дается возможность уделять время только разработке.

Заключение

В ходе практики я приобрёл ценный опыт по созданию функционального веб-приложения для обучения водителей, включающего систему регистрации, тестирование и личный кабинет. Практическая работа позволила мне освоить ключевые этапы разработки современного веб-продукта — от проектирования пользовательского интерфейса до его развертывания и обеспечения безопасности. Особое внимание уделялось проектированию UI/UX в Figma, где я научился создавать удобные и интуитивно понятные интерфейсы, работать с прототипами и дизайн-системами, а также эффективно взаимодействовать с командой. Это помогло понять, как важна визуальная и функциональная составляющая для успешного восприятия приложения пользователями.

Также я освоил методы функционального и нагрузочного тестирования, что позволило убедиться в корректной работе всех компонентов приложения и его устойчивости под высокой нагрузкой. Практическое применение инструментов, таких как Django TestCase и Locust, дало понимание, как выявлять и устранять ошибки ещё на ранних этапах, обеспечивая стабильность и надёжность продукта. Процесс деплоя приложения на хостинг ISPManager научил меня управлять серверным окружением, настраивать обработчики Python и обеспечивать доступность веб-приложения в интернете. Это важный навык для выпуска продукта в реальную эксплуатацию.

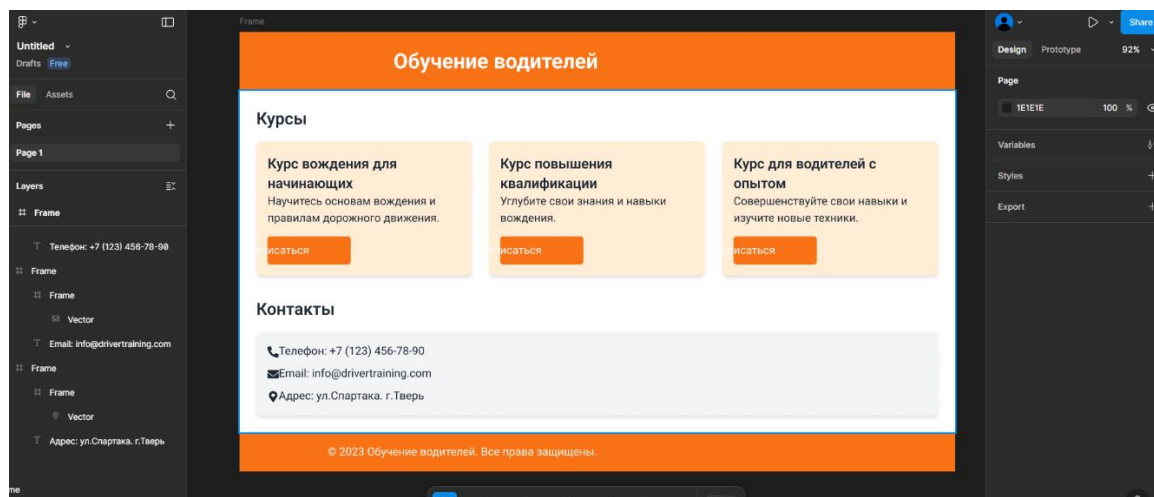
Наконец, настройка безопасности веб-приложения позволила глубже понять современные угрозы и способы их предотвращения с помощью встроенных механизмов Django и дополнительных мер. Я осознал, что безопасность — это неотъемлемая часть разработки, требующая постоянного внимания и совершенствования.

В целом, практика стала отличной возможностью применить теоретические знания на практике, освоить современные инструменты и технологии, а также подготовиться к решению реальных задач в сфере веб-разработки. Полученные навыки и опыт будут полезны для дальнейшего роста.


Список источников

1. Разработка в Figma и что такое UI/UX [Электронный ресурс] / Режим доступа: <https://codecoda.com/en/blog/entry/using-figma-for-user-experience-design>
2. Тестирование функционала [Электронный ресурс] / Режим доступа: <https://habr.com/ru/articles/715376/>
3. Нагрузочное тестирование [Электронный ресурс] / Режим доступа: <https://ifellow.ru/media-center/rukovodstvo-po-nagruzochnomu-testirovaniyu-veb-prilozheniy/>
4. Деплой приложения на хостинг ISPManager [Электронный ресурс] / Режим доступа: <https://mchost.ru/qa/q/kak-ustanovit-django-na-khosting-s-ispmanager>
5. Настройка безопасности веб-приложение [Электронный ресурс] / Режим доступа: <https://django.fun/docs/django/5.0/topics/security/>

Приложение 1. Макет в Figma



Приложение 2. Антиплагиат

**Antiplagius**
№1 в России

Антиплагиат 2.0, Проверка и повышение
уникальности текста за 2 минуты

antiplagius.ru

Уважаемый пользователь!
Обращаем ваше внимание, что система Антиплагиус отвечает на вопрос, является тот или иной фрагмент текста заимствованным или нет. Ответ на вопрос, является ли заимствованный фрагмент именно плагиатом, а не законной цитатой, система оставляет на ваше усмотрение.

Отчет о проверке № 9270383

Дата загрузки: 2025-04-21 00:45:25
Пользователь: aidarbekovadahan8@gmail.com, ID: 9270383

Отчет предоставлен сервисом «Антиплагиат»
на сайте antiplagius.ru/

Информация о документе

№ документа: 9270383
Имя исходного файла: Айдарбеков Адахан. Отчет по практике.docx
Размер файла: 1.16 МБ
Размер текста: 11913
Слов в тексте: 1526
Число предложений: 154

Информация об отчете

Дата: 2025-04-21 00:45:25 - Последний готовый отчет
Оценка оригинальности: 83%
Заемствования: 17%

83.25%

16.75%

Приложение 3. QR-код репозитория

