Анастасия Трухина

+7-925-404-97-46 | trukhina12345@mail.ru | github.com/NT-005-TN

Образование

Московский технический университет связи и информатики Москва

Бакалавриат. Программная инженерия: Разработка и сопровождение программного обеспечения. *2024 – 2028*

Опыт работы

Android-разработчик (фриланс) Июнь 2025 – Август 2025(3 месяца)

Удаленный формат

- Реализация архитектуры экранов приложения с использованием паттерна Model-View-Presenter (MVP) для обеспечения четкого разделения логики и пользовательского интерфейса.
- Разработка асинхронного взаимодействия с сетью и базами данных с применением Kotlin Coroutines, предотвращая блокировку UI-потока и обеспечивая отзывчивость приложения.
- Интеграция локального хранилища данных через библиотеку Room Persistence Library (SQLite) с автоматической генерацией Data Access Objects (DAO) для типобезопасных операций с сущностями.
- Интеграция облачных сервисов Firebase (Authentication, Cloud Firestore) для реализации аутентификации пользователей и синхронизации данных в реальном времени.
- Внедрение системы автоматического управления зависимостями и жизненным циклом компонентов с использованием Dagger Hilt, что повысило модульность и тестируемость кода.
- Автоматизация процессов парсинга JSON-данных от внешнего REST API (NHTSA) с помощью библиотеки Moshi, обеспечивая эффективное преобразование сетевых ответов в объекты Kotlin.
- Реализация пользовательского интерфейса с применением ViewBinding для автоматической генерации типобезопасных ссылок на элементы макета, ускоряя разработку и снижая вероятность ошибок.

Проекты

Автомобильный менеджер |

Kotlin, Android SDK, Firebase, Room, Moshi, Coroutines, Hilt

- Разработано нативное Android-приложение с архитектурой MVP для управления личным автопарком пользователя.
- Реализована аутентификация и регистрация пользователей с помощью Firebase Authentication и синхронизация профилей через Cloud Firestore.
- Обеспечено надежное локальное хранение данных об автомобилях с использованием библиотеки Room (SQLite) с автоматической синхронизацией с облаком Firestore.
- Интегрировано внешнее REST API (NHTSA) для автоматического декодирования VIN-кодов, используя HttpUrlConnection и Moshi для парсинга JSON-ответов.
- Применены Kotlin Coroutines для асинхронного выполнения сетевых запросов, операций с БД и управления состоянием UI без блокировки основного потока.
- Внедрена система управления зависимостями и жизненным циклом компонентов с помощью Dagger Hilt, повысив модульность и тестируемость кода.
- Создан интуитивный пользовательский интерфейс с возможностью добавления автомобилей как по VIN-коду, так и через ручной ввод данных.
- Ссылка на репозиторий с кодом проекта: gitfront.io/r/NTr/X7ZAxurAcbvR/ctproject/

Веб-сервис поиска Wikipedia |

Python, FastAPI, Uvicorn, Pydantic, Wikipedia-API

- Разработан асинхронный веб-сервис для поиска и получения информации из Wikipedia с использованием фреймворка FastAPI.
- Реализован RESTful API с эндпоинтами для получения кратких сводок по темам, поиска статей и получения полной информации о статье.
- Интегрирована внешняя библиотека Wikipedia-API для взаимодействия с REST API Википедии и выполнения поисковых запросов.
- Применен Pydantic для определения строгих схем данных запросов и ответов, обеспечив автоматическую валидацию и документирование API.

- Использован Uvicorn в качестве ASGI-сервера для запуска и развертывания приложения с поддержкой автоматической перезагрузки при разработке.
- Автоматизирована генерация интерактивной документации API (Swagger UI, ReDoc) средствами FastAPI на основе определенных маршрутов и моделей.
- Обработаны различные типы параметров HTTP-запросов: path-параметры (`/summary/{topic}`), query-параметры (`?q=search_term`) и тело запроса (POST).
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/FastAPI-using-wikipedia-for-ex

Шуточный банковский веб-сайт "ФПЕ Банк" |
Python, Flask, Flask-Login, Pillow, JSON, Хэширование, HTML/CSS/JS

- Реализована веб-платформа-пародия на банковский сайт в рамках "первоапрельского хакатона"с сознательно ужасной архитектурой.
- Использован микрофреймворк `Flask` как основа для веб-приложения. Управление сессиями пользователей (регистрация, вход, выход) реализовано с помощью библиотеки `Flask-Login`.
- Вся пользовательская информация (имя, телефон, хэш пароля, баланс, номер карты, страна) хранится в едином JSON-файле (`data.json`) без использования полноценной СУБД.
- Интегрирована сатирическая "админ-панель"с уникальной формой аутентификации: доступ осуществлялся путем загрузки изображения и побайтного сравнения `MD5`-хэшей изображений с использованием библиотеки `Pillow` (PIL).
- Применена библиотека `Werkzeug` для безопасного хэширования паролей пользователей (`generate_password_hash`, `check_password_hash`).
- Реализованы REST-like API endpoint'ы для управления пользовательскими данными: получение случайных средств, "казино"с заведомо проигрышной механикой, экспорт/импорт всей пользовательской базы данных администратором.
- Создан пользовательский интерфейс с использованием `HTML/CSS/JS` и шаблонизатора `Jinja2`, включая элементы сатирического UX/UI (например, выбор страны через "колесо фортуны").
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/The$_w$ors$e_b$ank$_w$ebsite

Игра Крестики-Нолики (Tic Tac Toe) |
Java, Swing, AWT, Event Handling

- Разработано настольное графическое приложение "Крестики-Нолики"с использованием стандартной библиотеки Java Swing для создания пользовательского интерфейса.
- Реализована игровая логика с чередованием ходов игроков (X и O) и проверкой условий победы (ряды, диагонали) после каждого хода.
- Использована объектно-ориентированная модель событий AWT/Swing: зарегистрированы слушатели событий (`ActionListener`) для кнопок игрового поля и кнопки перезапуска.
- Применены компоненты Swing: `JFrame` (главное окно), `JPanel` (контейнер для элементов), `JLabel` (отображение текущего игрока/результата), `JButton` (клетки поля и кнопка перезапуска).
- Реализован алгоритм определения победителя путем анализа состояния всех кнопок игрового поля после каждого хода и сравнения комбинаций с выигрышными условиями.
- Создан интуитивный пользовательский интерфейс с визуальной обратной связью: отображение текущего игрока, блокировка поля после победы или заполнения, кнопка перезапуска игры.
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/Portfolio/blob/main/Java$_T$ic$_T$ac$_T$oe

Телеграм-бот прогноза погоды |
Python, pyTelegramBotAPI, Requests, OpenWeatherMap API

- Разработан Telegram-бот для предоставления пользователю актуального прогноза погоды для любого города мира по запросу.
- Интегрирован внешний API сервиса OpenWeatherMap для получения данных о погоде (температура, ощущается как, скорость ветра, состояние неба) по названию города.
- Реализована обработка HTTP-запросов к REST API OpenWeatherMap с использованием библиотеки `requests` и парсинг JSON-ответа для извлечения необходимых meteorological данных.
- Использована библиотека `pyTelegramBotAPI` (telebot) для создания обработчиков сообщений (`@bot.message_handler`), управления состоянием бота и отправки ответов пользователю через Telegram API.
- Обеспечена надежная работа бота в непрерывном режиме с использованием `bot.polling` и реализации базовой обработки исключений для предотвращения краха приложения.

- Создан интуитивный пользовательский интерфейс бота: реализована команда `/start` для приветствия и инструкции пользователю, автоматическая обработка текстовых сообщений как названий городов.
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/Portfolio/tree/main/Python_Telegram_bot_weathersearcher

Игра "Убегающая кнопка"(Running Button) |
Java, Swing, AWT, Event Handling, Random

- Разработано настольное графическое приложение-игра "Убегающая кнопка"с использованием стандартной библиотеки Java Swing для создания динамического пользовательского интерфейса.
- Реализована игровая механика, при которой кнопка случайным образом перемещается по игровому полю при каждом клике пользователя, изменяя свое положение и цвет.
- Использована объектно-ориентированная модель событий AWT/Swing: зарегистрированы слушатели событий (`ActionListener`) для обработки кликов по кнопке.
- Применены компоненты Swing: `JFrame` (главное окно), `JPanel` (контейнер для элементов с `null` layout для абсолютного позиционирования), `JButton` (основной игровой элемент).
- Реализована генерация псевдослучайных чисел (`Math.random()`, `Random`) для определения новых координат кнопки, цвета и условий появления новых элементов.
- Создана динамическая логика интерфейса: при достижении определенного счетчика кликов кнопка исчезает, а при четных значениях счетчика появляются новые кнопки, увеличивая сложность.
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/Portfolio/blob/main/RunningButton.java

Графический калькулятор |
Java, Swing, AWT, Event Handling

- Разработано настольное графическое приложение калькулятор с использованием стандартной библиотеки Java Swing для создания пользовательского интерфейса.
- Реализована логика арифметических операций (сложение, вычитание, умножение, деление) с вещественными числами, включая обработку последовательного ввода операндов и операторов.
- Использована объектно-ориентированная модель событий AWT/Swing: зарегистрированы слушатели событий (`ActionListener`) для кнопок (цифры, операции, равно, очистка) и текстовых полей.
- Применены компоненты Swing: `JFrame` (главное окно), `JPanel` (контейнер для элементов), `JTextField` (поля для ввода операндов и отображения результата), `JButton` (кнопки цифр, операций, управления), `JLabel` (статусное поле).
- Реализована базовая валидация пользовательского ввода для предотвращения ошибок вычисления (пустые поля, некорректные символы).
- Создан интуитивный пользовательский интерфейс с визуальной обратной связью: отдельные поля для операндов и оператора, кнопка очистки (`C`), отображение результата и сообщений об ошибках.
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/Portfolio/blob/main/Calc2.java

Телеграм-бот прогноза погоды |
Python, pyTelegramBotAPI, Requests, OpenWeatherMap API

- Разработан Telegram-бот для предоставления пользователю актуального прогноза погоды для любого города мира по запросу.
- Интегрирован внешний API сервиса OpenWeatherMap для получения данных о погоде (температура, ощущается как, скорость ветра, состояние неба) по названию города.
- Реализована обработка HTTP-запросов к REST API OpenWeatherMap с использованием библиотеки `requests` и парсинг JSON-ответа для извлечения необходимых meteorological данных.
- Использована библиотека `pyTelegramBotAPI` (telebot) для создания обработчиков сообщений (`@bot.message_handler`), управления состоянием бота и отправки ответов пользователю через Telegram API.
- Обеспечена надежная работа бота в непрерывном режиме с использованием `bot.polling` и реализации базовой обработки исключений для предотвращения краха приложения.
- Создан интуитивный пользовательский интерфейс бота: реализована команда `/start` для приветствия и инструкции пользователю, автоматическая обработка текстовых сообщений как названий городов.
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/Portfolio/blob/main/TelegramBot.py

Игра "Жизнь"(Conway's Game of Life) |
Python, Pygame, ctypes

- Разработана визуализация клеточного автомата "Игра жизнь"Конвея с использованием библиотеки `Pygame` для создания графического интерфейса и анимации.
- Реализована основная логика игры: автоматическое обновление состояния клеток на двумерной сетке в соответствии с правилами рождения, выживания и смерти, зависящими от количества соседей.
- Использована библиотека `ctypes` для получения реального разрешения экрана монитора с учетом DPI, что позволяет динамически подстраивать размер игрового поля под конкретное устройство.
- Применены компоненты `Pygame`: `pg.display.set_mode` (создание окна), `pg.time.Clock` (управление частотой кадров FPS), `pg.event.get` (обработка событий закрытия окна), `pg.draw.line/rect` (рисование сетки и клеток).
- Реализована оптимизация производительности: использование двойной буферизации (`field` и `new_field`) и глубокое копирование (`copy.deepcopy`) для корректного обновления состояния сетки на каждом шаге симуляции.
- Создан адаптивный пользовательский интерфейс: игровое поле автоматически масштабируется под разрешение экрана, отрисована сетка для визуального разделения клеток, черные прямоугольники представляют живые клетки.
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/LifeGamePython

Демонстрационный веб-сайт с формами авторизации/регистрации |
HTML, CSS, JavaScript, Google Fonts API

- Разработана статическая веб-страница с адаптивным пользовательским интерфейсом, демонстрирующая базовые формы входа и регистрации пользователя.
- Использована семантическая разметка HTML для структурирования контента: элементы `header`, `nav`, `div`, `form`, `input`, `button`, `label`, `a`.
- Применены современные возможности CSS3: Flexbox для выравнивания элементов, CSS Grid (косвенно, через Flex), псевдоклассы (`:hover`, `:focus`, `:valid`), анимации (`transition`, `transform`), медиазапросы, свойства фона (`background-size`, `background-position`), размытие (`backdrop-filter`) и тени (`box-shadow`).
- Интегрирован внешний шрифт `Poppins` от Google Fonts API для улучшения визуальной привлекательности и типографики сайта.
- Реализована интерактивность пользовательского интерфейса с помощью нативного JavaScript (ES6): переключение между формами входа/регистрации, открытие/закрытие модального окна форм, анимация переходов.
- Создан адаптивный дизайн с плавными переходами и эффектами при наведении, обеспечивающий современный и интуитивно понятный пользовательский опыт.
- Ссылка на репозиторий с кодом проекта: github.com/NT-005-TN/Site-Works_Successful

Технические навыки

Languages: Java, Python, Kotlin, C/C++, SQL (Postgres, SQLite, Oracle), JavaScript, HTML/CSS, VBA
Frameworks: React, Node.js, Flask, JUnit, WordPress, Material-UI, FastAPI, Android SDK
Developer Tools: Git, VS Code, Visual Studio, PyCharm, IntelliJ IDEA, Eclipse, Android Studio, Hilt, Room, Firebase, Moshi, Coroutines, Gradle, Postman
Libraries: Pygame, Telebot, Requests, Pydantic, OpenWeatherMap API, NHTSA API, Wikipedia-API, ctypes, fnmatch
Technologies & Concepts: Object-Oriented Programming (OOP), REST APIs, MVC/MVP Architecture, Database Design (SQLite, Room, JSON, Oracle), Asynchronous Programming (Coroutines), Dependency Injection (Hilt), Mobile Development (Android), Web Development (HTML/CSS/JS, Flask, FastAPI), Git Version Control, Excel
Additional: Technical English (B2)