

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа «Программная инженерия»

СОГЛАСОВАНО

Приглашенный преподаватель ФКН
НИУ ВШЭ

_____ Н. И. Веселко
«__» _____ 2026 г.

УТВЕРЖДЕНО

Академический руководитель
образовательной программы
«Программная инженерия», старший
преподаватель департамента
программной инженерии

_____ Н. А. Павлов
«__» _____ 2026 г.

**ГЕЙМИФИЦИРОВАННОЕ МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ
УПРАВЛЕНИЯ ЗАДАЧАМИ И ВИЗУАЛИЗАЦИИ ПРОГРЕССА - TASKORIA**

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.06.05-01 12 01-1-ЛУ

Исполнитель:

Студент группы БПИ247

_____/ В. А. Путинцева /

«__» _____ 2026 г.

Подп. и дата	
Инв.№ дубл.	
Взам. инв.№	
Подп. и дата	
Инв.№ подл	

УТВЕРЖДЕН

RU.17701729.06.05-01 12 01-1-ЛУ

**ГЕЙМИФИЦИРОВАННОЕ МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ
УПРАВЛЕНИЯ ЗАДАЧАМИ И ВИЗУАЛИЗАЦИИ ПРОГРЕССА - TASKORIA**

Текст программы

RU.17701729.06.05-01 12 01-1

Листов 9

Инв.№ подл	Подп. и дата	Взам. инв.№	Инв.№ дубл.	Подп. и дата

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	3
1. ТЕКСТ ПРОГРАММЫ.....	4
1.1. Структура репозитория.....	4
1.2. Серверная часть	4
1.3. Уровень API серверной части.....	5
1.4. Модели, схемы и репозитории.....	5
1.5. Сервисный слой серверной части	6
1.6. Дополнительные архитектурные модули backend	6
1.7. Клиентская часть.....	6
1.8. Слой app клиентской части.....	7
1.9. Слой features клиентской части	7
1.10. Графические ресурсы.....	7
1.11. Тесты	8
1.12. Конфигурационные файлы	8
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ.....	9

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.05-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1. ТЕКСТ ПРОГРАММЫ

Программа Taskoria представляет собой программный продукт, состоящий из серверной части и мобильного Android-приложения. Полный текст программы размещен в репозитории GitHub по адресу: <https://github.com/vputt/Taskoria>. В настоящем документе приведены состав исходного кода, структура репозитория и назначение основных программных модулей.

1.1. Структура репозитория

Исходный код проекта расположен в корневом каталоге репозитория и разделен на несколько основных частей:

- backend - серверная часть приложения;
- productivity_city - клиентское мобильное приложение на Flutter;
- README.md - общее описание проекта и команды запуска;
- docker-compose.yml - конфигурация локального запуска backend и PostgreSQL в контейнерах;
- .gitignore - перечень файлов и каталогов, не включаемых в репозиторий.

Разделение проекта на backend и frontend позволяет независимо развивать серверную бизнес-логику и пользовательский интерфейс. Серверная часть отвечает за хранение и обработку данных, а клиентская часть - за отображение экранов, обработку действий пользователя и взаимодействие с API.

1.2. Серверная часть

Код серверной части находится в каталоге backend. Серверная часть реализована на Python с использованием FastAPI. Она отвечает за регистрацию и авторизацию пользователя, работу с задачами и подзадачами, начисление наград, обновление города, достижения, магазин и взаимодействие с PostgreSQL.

Основные файлы и каталоги серверной части:

- backend/app/main.py - точка входа FastAPI-приложения, подключение маршрутов и настройка приложения;
- backend/app/database.py - настройка подключения к базе данных и создание сессий;
- backend/app/api - HTTP-маршруты серверного API;
- backend/app/models - SQLAlchemy-модели базы данных;
- backend/app/schemas - Pydantic-схемы запросов и ответов API;
- backend/app/repositories - слой доступа к данным;
- backend/app/services - бизнес-логика приложения;

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.05-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

- backend/app/strategies - стратегии работы с ИИ-провайдерами;
- backend/app/builders - сборка состояния города;
- backend/app/factories - фабрики создания доменных объектов;
- backend/app/states - логика состояний, связанных со streak пользователя;
- backend/app/templates - шаблонные алгоритмы расчета наград;
- backend/app/utils - вспомогательные модули, включая клиент GigaChat и seed-данные;
- backend/tests - автоматические тесты серверной части;
- backend/Dockerfile - описание контейнера backend.

1.3. Уровень API серверной части

В каталоге backend/app/api реализован слой HTTP-маршрутов. Он принимает запросы от клиентского приложения, выполняет проверку авторизации и передает управление сервисам. Файл auth.py содержит маршруты регистрации и входа. Файл users.py отвечает за получение данных текущего пользователя. Файл tasks.py реализует операции с задачами, включая создание, редактирование, удаление, завершение и разбиение задачи на подзадачи. Файл subtasks.py содержит операции с подзадачами. Файлы city.py и shop.py отвечают за состояние города и магазина.

В файле deps.py описаны зависимости FastAPI: получение текущего пользователя, подключение репозитория и сервисов. Такой подход позволяет отделить обработку HTTP-запросов от бизнес-логики и упростить тестирование.

1.4. Модели, схемы и репозитории

В каталоге backend/app/models описаны основные сущности базы данных: пользователь, задача, подзадача, здание, достижение, статистика и товар магазина. Модели определяют структуру таблиц, поля, связи и ограничения на уровне базы данных.

В каталоге backend/app/schemas находятся Pydantic-схемы. Они описывают формат входных и выходных данных API. Схемы используются для валидации запросов клиента и для формирования корректных JSON-ответов.

В каталоге backend/app/repositories реализован слой доступа к данным. Репозитории инкапсулируют операции чтения, создания, обновления и удаления записей. Например, task_repository.py работает с задачами, subtask_repository.py - с подзадачами, user_repository.py - с пользователями, shop_repository.py - с покупками магазина. Наличие отдельного слоя репозитория позволяет сервисам не обращаться к базе данных напрямую.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.05-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.5. Сервисный слой серверной части

В каталоге `backend/app/services` реализована основная бизнес-логика приложения. `AuthService` отвечает за регистрацию и вход пользователя. `TaskService` выполняет операции с задачами, включая создание, обновление, завершение и разбиение на подзадачи. `CityService` формирует состояние города и обновляет прогресс зданий. `ShopService` отвечает за каталог товаров, покупку и размещение объектов. `RewardService` рассчитывает награды XP и Coins. `AIService` связывает бизнес-логику задач с ИИ-подсистемой.

Сервисный слой отделен от слоя API и слоя хранения данных. Это позволяет использовать одну и ту же бизнес-логику как при обработке HTTP-запросов, так и в автоматических тестах.

1.6. Дополнительные архитектурные модули backend

В каталоге `backend/app/strategies` реализована стратегия работы с ИИ-провайдером. В текущей версии используется `GigaChatStrategy`, которая обращается к GigaChat API и возвращает подзадачи. При ошибке внешнего сервиса предусмотрен fallback-результат, чтобы пользовательский сценарий не прерывался полностью.

В каталоге `backend/app/builders` находится `CityBuilder`, который собирает итоговое состояние города из данных пользователя и построек. В каталоге `backend/app/factories` находится `BuildingFactory`, отвечающая за создание объектов зданий. В каталоге `backend/app/core` расположены вспомогательные архитектурные компоненты: политика прогрессии города, контейнер зависимостей, Event Bus и наблюдатели событий.

1.7. Клиентская часть

Код клиентской части находится в каталоге `productivity_city`. Клиентская часть реализована на Flutter и предназначена для запуска на Android-устройствах. Основной код приложения находится в каталоге `productivity_city/lib`.

Структура клиентской части:

- `productivity_city/lib/main.dart` - точка входа Flutter-приложения;
- `productivity_city/lib/app` - настройка приложения, маршрутизация и тема;
- `productivity_city/lib/features` - пользовательские экраны и функциональные разделы;
- `productivity_city/lib/shared` - общие модели, провайдеры, сетевой клиент, сессия, mock-данные и переиспользуемые виджеты;
- `productivity_city/assets` - графические ресурсы приложения;
- `productivity_city/test` - тесты клиентской части;
- `productivity_city/pubspec.yaml` - зависимости Flutter-приложения и подключение assets.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.05-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.8. Слой app клиентской части

В каталоге `productivity_city/lib/app` находятся общие настройки клиентского приложения. Файл `app.dart` содержит корневой виджет приложения. Файл `router.dart` отвечает за маршрутизацию между экранами, включая экран загрузки, онбординг, регистрацию, вход и основные разделы. В каталоге `theme` описаны цвета, отступы, радиусы, текстовые стили и общая тема интерфейса.

Такое разделение позволяет хранить навигацию и визуальный стиль отдельно от экранов конкретных функций. Благодаря этому разные разделы приложения используют единые элементы оформления.

1.9. Слой features клиентской части

В каталоге `productivity_city/lib/features` находятся пользовательские разделы приложения:

- `splash` - экран загрузки приложения;
- `onboarding` - вводные экраны;
- `auth` - экраны входа и регистрации;
- `home` - главный экран города;
- `tasks` - список задач, форма задачи, карточка задачи и компоненты подзадач;
- `calendar` - календарь активности;
- `shop` - магазин и карточка товара;
- `achievements` - достижения пользователя;
- `profile` - профиль и статистика;
- `notifications` - центр уведомлений;
- `settings` - настройки приложения.

В разделе `home` находятся главный экран города и виджеты изометрической карты. Файл `isometric_city_map.dart` отвечает за отрисовку карты, дорог, зданий и декоративных объектов. Файл `city_map_specs.dart` содержит спецификации зданий и объектов: изображения, уровни, `footprint` и `anchor`-точки для точного позиционирования на сетке.

Раздел `tasks` содержит основные экраны работы с задачами. Файл `tasks_screen.dart` отображает список задач, поиск и фильтрацию. Файл `task_form_screen.dart` реализует форму создания и редактирования задачи. Файл `task_details_screen.dart` отображает подробную карточку задачи и список подзадач. В каталоге `widgets` находятся переиспользуемые карточки задач и элементы подзадач.

1.10. Графические ресурсы

Графические ресурсы приложения находятся в каталоге `productivity_city/assets`. Они включают изображения зданий разных уровней, декоративные объекты магазина, иконки, иллюстрации для

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.05-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

онбординга и другие элементы интерфейса. Пути к ресурсам централизованы в файле `productivity_city/lib/shared/assets/asset_paths.dart`.

Для изометрического города используются pixel-art изображения. Позиционирование зданий на карте задается не произвольными координатами экрана, а спецификациями с anchor-точками, которые позволяют привязать конкретную точку изображения к ячейке карты.

1.11. Тесты

Автоматические тесты серверной части находятся в каталоге `backend/tests`. В них проверяются ключевые сценарии: обновление города, покупка товаров магазина, завершение задач и разбиение задач на подзадачи. Тесты запускаются с помощью `pytest`.

Тесты клиентской части находятся в каталоге `productivity_city/test`. Они проверяют запуск приложения и базовое отображение стартового экрана. Дополнительно клиентская часть проверяется командой `flutter analyze` и ручным тестированием основных экранов на Android-устройстве.

1.12. Конфигурационные файлы

В корне проекта расположен файл `docker-compose.yml`, который используется для локального запуска `backend` и PostgreSQL. В `backend/.env.example` приведен пример переменных окружения для серверной части. В `backend/requirements.txt` перечислены зависимости Python. В `productivity_city/pubspec.yaml` перечислены зависимости Flutter-приложения и подключенные графические ресурсы.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.05-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ 19.101-77: Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77: Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77: Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78: Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78: Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78: Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.401-78: Текст программы. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78: Общие правила внесения изменений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78: Правила внесения изменений в программные документы, выполненные печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.05-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

[illegible]