

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГУ»)

Факультет Компьютерных наук  
Кафедра программирования и информационных технологий

Техническое задание  
на разработку мобильного приложения  
«Мобильное приложение для самоорганизации»

Исполнители

\_\_\_\_\_ А.М. Косенко  
\_\_\_\_\_ А.М. Лаптев  
\_\_\_\_\_ Е.Е. Малыхина  
\_\_\_\_\_ Е.В. Саломатова  
\_\_\_\_\_ Д.Д. Шульга  
\_\_\_\_\_ Н.О. Шульга

Заказчик

\_\_\_\_\_ В.С. Тарасов

Воронеж 2024

## Содержание

1	Общие сведения.....	4
1.1	Наименование приложения.....	4
1.2	Заказчик.....	4
1.3	Исполнитель.....	4
1.4	Перечень документов, на основании которых создается система.....	4
1.5	Плановые сроки начала и окончания работ.....	5
1.6	Термины и сокращения.....	5
2	Назначение и цели создания.....	7
2.1	Назначение приложения.....	7
2.2	Цели создания приложения.....	7
3	Характеристика объектов автоматизации.....	8
3.1	Пользовательские роли.....	8
3.2	Пользовательские сценарии.....	9
3.2.1	Добавление задачи.....	9
3.2.2	Просмотр списка задач.....	10
4	Требования к системе.....	11
4.1	Требования к системе в целом.....	11
4.1.1	Перечень подсистем.....	11
4.1.2	Требования к способам обеспечения взаимодействия.....	11
4.1.3	Требования к режимам функционирования.....	11
4.1.4	Требования по диагностированию системы.....	12
4.1.5	Перспективы развития, модернизации системы.....	12
4.2	Требования к функциям, выполняемым системой.....	12
4.3	Требования к видам обеспечения.....	13
4.3.1	Требования к информационному обеспечению.....	13
4.3.2	Требования к лингвистическому обеспечению.....	14
4.3.3	Требования к программному обеспечению приложения.....	14
4.4	Общие технические требования к системе.....	15
4.4.1	Требования к надёжности.....	15

4.4.2	Требования к безопасности.....	15
5	Состав и содержание работ по созданию системы.....	16
6	Порядок контроля и приёмки системы.....	17
7	Требования к документированию.....	18
8	Источники разработки.....	19

## **1 Общие сведения**

### **1.1 Наименование приложения**

Полное наименование: Taskbench.

### **1.2 Заказчик**

Старший преподаватель Тарасов Вячеслав Сергеевич, кафедра программирования и информационных технологий

### **1.3 Исполнитель**

Студент Косенко Алексей Михайлович, кафедра программирования и информационных технологий.

Студент Лаптев Александр Михайлович, кафедра программирования и информационных технологий.

Студентка Малыхина Елена Евгеньевна, кафедра программирования и информационных технологий.

Студентка Саломатова Елизавета Викторовна, кафедра программирования и информационных технологий.

Студентка Шульга Дарья Дмитриевна, кафедра программирования и информационных технологий

Студент Шульга Никита Олегович, кафедра программирования и информационных технологий.

### **1.4 Перечень документов, на основании которых создается система**

- Федеральный закон «Об информации, информационных технологиях и о защите информации» от 27.07.2006 N 149–ФЗ;
- Федеральный закон «О персональных данных» от 27.07.2006 N 152–ФЗ;

- Федеральный закон «О коммерческой тайне» от 29.07.2004 N 98–ФЗ;
- Закон РФ от 07.02.1992 N 2300-1 (ред. от 11.06.2021) «О защите прав потребителей».

### 1.5 Плановые сроки начала и окончания работ

Срок начала работ — 20 февраля 2025.

Плановый срок окончания работ — 1 июня 2025.

### 1.6 Термины и сокращения

- **Python** — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью.
- **Django** — фреймворк на языке программирования Python, который позволяет быстро создавать безопасные веб-приложения.
- **Compose UI** — декларативный фреймворк на языке программирования Kotlin для графического интерфейса под Android-устройства с возможностью расширения под разные платформы.
- **GitHub** - крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.
- **REST API** — стиль архитектуры программного обеспечения для построения распределённых масштабируемых веб-сервисов.
- **PostgreSQL** — реляционная база данных с открытым исходным кодом с большими возможностями для масштабирования.

- **Клиентская сторона** — программно–аппаратная часть веб-приложения, работающая на устройстве пользователя. Отвечает за отображение интерфейса, обработку ввода и взаимодействие с сервером.
- **Сервер, серверная сторона** — компьютер, обслуживающий другие компьютеры (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач.
- **Фреймворк** — Программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.
- **Getting Things Done** — Методика повышения личной эффективности посредством перенесения списка текущих задач на внешний носитель, созданная Дэвидом Алленом.

## **2 Назначение и цели создания**

### **2.1 Назначение приложения**

Приложение предназначено для автоматизации и упрощения процесса самоорганизации пользователя. Чтобы улучшить пользовательский опыт предлагается провести следующие действия:

- Ускорить и упростить ввод и создание новой задачи;
- Автоматически классифицировать создаваемые задачи в общем списке.

### **2.2 Цели создания приложения**

Цели создания:

- Внедрение технологии искусственного интеллекта в систему обработки ввода пользователя;
- Создание базы данных для хранения пользовательских записей для доступа к ним с различных устройств;
- Предоставление возможности анализа пользовательской продуктивности с выводом статистических данных пользователю.

### **3 Характеристика объектов автоматизации**

#### **3.1 Пользовательские роли**

В системе можно выделить четыре вида пользователей с разными возможностями:

- Неавторизованный пользователь;
- Авторизованный пользователь;
- Премиум пользователь;
- Администратор.

Система подразумевает, что пользователи не видят рабочее пространство других пользователей, то есть рассчитана на личное пользование, поэтому возможности неавторизованного пользователя крайне ограничены. Неавторизованным пользователям доступны следующие действия:

- Создание аккаунта и регистрация в системе;
- Вход в существующий аккаунт.

Авторизованный пользователь является основным видом пользователя. Ему доступны следующие действия:

- Все действия, допустимые для неавторизованных пользователей, за исключением создания аккаунта;
- Просмотр своих задач;
- Добавление, изменение, удаление своих задач;
- Отмена уже выполненных задач;



- Присвоение задаче срока окончания, приоритета и (или) категории;
- Создание категории;
- Просмотр своей статистики;
- Оплата подписки для перехода в категорию премиум-пользователей.
- Изменение сохранённых персональных данных в настройках.

Премиум-пользователю доступны следующие действия:

- Все действия, допустимые для авторизованных пользователей;
- Использование автоматической обработки ввода на основе искусственного интеллекта для автоматического выделения подзадач, присвоения сроков, категории и приоритета создаваемым задачам.

Администратор не взаимодействует с пользователями напрямую в целях сохранения конфиденциальности. Ему доступны следующие действия:

- Просмотр общей статистики использования приложения;
- Просмотр количества премиум-пользователей.

## **3.2 Пользовательские сценарии**

### **3.2.1 Добавление задачи**

«Как пользователь, я хочу добавлять задачи в приложение, чтобы не забывать о делах.»

- Пользователь должен иметь возможность ввести заголовок задачи (обязательно);
- Пользователь должен иметь возможность добавить подзадачи (необязательно);
- Пользователь должен иметь возможность указать дату и время выполнения (необязательно);
- Пользователь должен иметь возможность указать категорию задачи (по умолчанию общая категория);
- После добавления задача появляется в общем списке задач;
- Пользователь должен иметь возможность отменить добавление до сохранения.

### **3.2.2 Просмотр списка задач**

«Как пользователь, я хочу просматривать список задач, чтобы понимать, что мне нужно сделать.»

- Пользователь должен видеть задачи в порядке их приоритета (по умолчанию);
- Пользователь должен иметь возможность выбрать другой способ сортировки (по дате, категории и т. д.);
- Выполненные задачи автоматически скрываются из списка;
- Нажатие на задачу должно открывать ее полное описание.

## **4 Требования к системе**

### **4.1 Требования к системе в целом**

#### **4.1.1 Перечень подсистем**

В системе можно выделить несколько подсистем:

- Мобильное приложение;
- Панель администрирования;
- Серверная часть;
- База данных;
- Сервис искусственного интеллекта.

#### **4.1.2 Требования к способам обеспечения взаимодействия**

Пользователь взаимодействует с системой через мобильное приложение, которое связано с серверной частью посредством REST API. Сервер обрабатывает входящие запросы мобильного приложения, используя фреймворк Django. В качестве базы данных используется PostgreSQL. Сервер связывается с внешним сервисом искусственного интеллекта посредством публичного REST API и при необходимости запрашивает обработку данных.

#### **4.1.3 Требования к режимам функционирования**

Основным режимом функционирования является нормальный режим. В таком режиме работа приложения и всех серверов должна поддерживаться круглосуточно с возможными перерывами на техническое обслуживание.

В случае отказа компонента программного обеспечения по возможности необходимо совершить резервное копирование данных и

завершить работу приложения, после чего устранить проблему и перезапустить все процессы.

#### **4.1.4 Требования по диагностированию системы**

Компоненты должны предоставлять возможности для отслеживания процесса выполнения программы. В случае ошибки или отказа какого-либо компонента, система должна сохранять информацию, необходимую для устранения этой ошибки.

#### **4.1.5 Перспективы развития, модернизации системы**

Исходный код компонентов системы должен иметь возможность дальнейшего расширения.

При дальнейшем росте количества пользователей возможно увеличение нагрузки на серверную часть приложения. В таком случае возможно увеличение мощности системы с использованием инструментов контейнеризации и балансировки трафика.

### **4.2 Требования к функциям, выполняемым системой**

Система должна предоставлять следующие функции:

- Создание аккаунта в базе данных;
- Удаление аккаунта из базы данных со всем контентом, созданным этим аккаунтом;
- Изменение персональных данных аккаунта в базе данных;
- Оплата подписки и присвоение аккаунту категории премиум-пользователя;
- Создание задачи в базе данных и присвоения её конкретному аккаунту;

- Редактирование задачи;
- Удаление задачи;
- Отметка задачи выполненной;
- Добавление подзадачи к задаче;
- Присвоение задаче срока;
- Присвоение задаче приоритета;
- Присвоение задаче категории;
- Автоматическое добавление задаче подзадачи, присвоения срока, категории и приоритета с использованием искусственного интеллекта;
- Просмотр задач пользователя из определенной категории, задач за определенный срок или задач определенного приоритета;
- Просмотр статистики по выполненным за последнюю неделю задачам;
- Просмотр статистики по самому продуктивному дню за последнюю неделю;
- Просмотр количества всех аккаунтов и количества премиум-пользователей.

## **4.3 Требования к видам обеспечения**

### **4.3.1 Требования к информационному обеспечению**

Для хранения данных применяется система управления базами данных PostgreSQL. Между компонентами системы данные передаются по REST API в формате JSON.

#### **4.3.2 Требования к лингвистическому обеспечению**

Приложение должно поддерживать русский язык.

#### **4.3.3 Требования к программному обеспечению приложения**

Требования к программному обеспечению клиентской части:

- Мобильное приложение должно функционировать на устройствах с операционной системой Android версией 8.0 или выше.

Требования к программному обеспечению серверной части:

- Серверная часть приложения должна быть реализована на языке программирования Python с использованием фреймворка Django;
- В качестве внешнего сервиса искусственного интеллекта должен использоваться сервис GigaChat, созданный ПАО СберБанк;
- В качестве СУБД должна использоваться PostgreSQL.

## **4.4 Общие технические требования к системе**

### **4.4.1 Требования к надёжности**

Система должна выдерживать нагрузку для базовой аудитории в 1000 пользователей. При будущем росте аудитории, требования будут повышаться.

### **4.4.2 Требования к безопасности**

Система должна обеспечивать безопасность данных:

- Защита от SQL-инъекций посредством встроенных во фреймворк Django инструментов для безопасного взаимодействия с базой данных;
- Аутентификация пользователей при авторизации с помощью безопасных протоколов;

## **5 Состав и содержание работ по созданию системы**

Работа по созданию системы состоит из следующих этапов:

- Фаза предпроектного исследования, включающая в себя анализ конкурентов, анализ целевой аудитории и рынка, разработку финансовой модели и дорожной карты, а также проработку функциональных требований и начальной архитектуры, создание макетов пользовательского графического интерфейса;
- Этап разработки минимально жизнеспособного продукта, который включает в себя разработку и развёртывание необходимых модулей приложения;
- Проведение тестирования, исправление найденных ошибок.



## **6 Порядок контроля и приёмки системы**

Поэтапные отчёты по работе будут проводиться во время рубежных аттестаций:

- Первая аттестация: пройден первый этап работ по созданию системы, подразумевающий проведение предпроектного исследования, разработку начальной архитектуры системы и организацию рабочих процессов;

- Вторая аттестация: пройден второй этап работ по созданию системы, подразумевающий наличие всех модулей приложения и взаимодействия между ними;

- Третья аттестация: пройден третий этап работ по созданию системы, подразумевающий соответствие протестированного приложения итоговым требованиям, а также наличие отчёта по курсовой работе.

## **7 Требования к документированию**

Документация к приложению должна быть написана на русском языке и описывать аспекты, необходимые для понимания архитектуры и взаимодействия модулей приложения.

Документация должна включать:

- Описание REST API в формате OpenAPI;
- Описание базы данных в формате ER-диаграммы;
- Инструкции по развёртыванию приложения.

## 8 Источники разработки

Во время разработки должен учитываться опыт схожих сервисов, таких как:

- TickTick (Appest Limited);
- Akiflow (Akiflow);
- Sunsama (Summay Inc.);
- Routine (Routine SAS);
- Notion (Notion Labs, Inc.);
- Nirvana (Nirvanahq Inc.);
- Things (Cultured Code GmbH & Co. KG);
- Omnifocus (Omni Development Inc.);
- Taskade (Taskade);
- Usemotion (Nexusbird, Inc).