**Изображение выглядит как логотип

Автоматически созданное описаниеМИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**(национальный исследовательский университет)»

**Институт (Филиал):** № 8 «Компьютерные науки и прикладная математика»

**Кафедра:** 806

**Группа:** М8О-401Б-2

**Направление подготовки:** 01.03.02 Прикладная математика и информатика

**Профиль:** Информатика

**Квалификация: бакалавр**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

на тему: «Разработка веб-приложения для решения задач смешано- целочисленного линейного программирования»

Автор ВКРБ: Знай Артемий Олегович (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Руководитель: Судаков Владимир Анатольевич (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Консультант: (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Консультант: (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Рецензент: (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**К защите допустить**

Заведующий кафедрой № 806 «Вычислительная математика  
и программирование» Крылов Сергей Сергеевич (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Москва 202**Изображение выглядит как логотип

Автоматически созданное описание**5

## **РЕФЕРАТ**

**СОДЕРЖАНИЕ**

[РЕФЕРАТ 2](#_Toc168398014)

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 4](#_Toc168398015)

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5](#_Toc168398016)

[ВВЕДЕНИЕ 6](#_Toc168398017)

[1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 8](#_Toc168398018)

[1.1 Компания 8](#_Toc168398019)

[1.2 Сотрудник 9](#_Toc168398020)

[1.3 Контакт 10](#_Toc168398021)

[1.4 Объявление 10](#_Toc168398022)

[1.5 Сделка 11](#_Toc168398023)

[1.6 Уведомление 13](#_Toc168398024)

[2 АРХИТЕКТУРА ПРОЕКТА 15](#_Toc168398025)

[2.1 Монолитная и микросервисная архитектуры 15](#_Toc168398026)

[2.2 Чистая архитектура 16](#_Toc168398027)

[2.3 Используемые технологии 20](#_Toc168398028)

[2.4 Взаимодействие между сервисами 25](#_Toc168398033)

[2.5 Авторизация и аутентификация 28](#_Toc168398036)

[2.6 Сервисы CRM-системы 30](#_Toc168398037)

[3 ОПИСАНИЕ РАБОТЫ 38](#_Toc168398045)

[3.1 Реализованная архитектура 38](#_Toc168398046)

[3.2 Взаимодействие с клиентом 38](#_Toc168398047)

[ЗАКЛЮЧЕНИЕ 42](#_Toc168398052)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc168398053)

[ПРИЛОЖЕНИЕ А Схема архитектуры проекта 44](#_Toc168398054)

[ПРИЛОЖЕНИЕ Б QR-код репозитория с исходным кодом 45](#_Toc168398055)

# **ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

# **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

# **ВВЕДЕНИЕ**

В современных условиях глобальной конкуренции и постоянного усложнения процессов принятия решений оптимизация деятельности предприятий приобретает всё большую значимость. Одной из ключевых направления в этой области является решение задач смешано-целочисленного линейного программирования (MILP), позволяющих моделировать и оптимизировать широкий спектр производственных, логистических и финансовых процессов. MILP-задачи характеризуются наличием как непрерывных, так и дискретных переменных, что делает их решение крайне полезным для систем, где принимаются как решения по объёму производства, так и по выбору конкретных стратегий распределения ресурсов.

Современные методы оптимизации, основанные на математическом моделировании, давно показали свою эффективность при решении сложных задач управления, однако их применение зачастую ограничено трудностями интеграции с реальными информационными системами предприятия. В этом контексте разработка веб-приложения для решения задач MILP представляет особый интерес, поскольку оно позволяет объединить мощные вычислительные возможности специализированных решателей (таких как GLPK, CBC, SCIP) с удобным и доступным интерфейсом, что облегчает практическое использование методов оптимизации даже пользователями без глубоких математических знаний.

Разрабатываемое веб-приложение призвано обеспечивать доступ к современным алгоритмам оптимизации через единый портал, где пользователь может загрузить исходные данные в виде таблиц (например, Excel-файлов), выбрать решатель и получить оптимальное решение задачи с анализом чувствительности. Такая интеграция способствует не только автоматизации процессов принятия решений, но и позволяет проводить детальный анализ параметров модели, оценивать влияние изменений входных данных на итоговую оптимизацию и, таким образом, принимать более обоснованные решения в условиях неопределенности и динамичного развития рынка.

Ключевой особенностью работы является построение архитектуры веб-приложения, которая объединяет фронтенд, реализованный с использованием современного фреймворка React, и бэкенд на Python с использованием фреймворка Flask и библиотеки Pyomo. Такой подход позволяет реализовать масштабируемую систему, способную работать с большими объёмами данных и предоставлять гибкие настройки параметров оптимизации. Важным элементом является реализация механизма обмена данными между клиентской и серверной частями с использованием протокола SSE, что обеспечивает оперативное получение промежуточных результатов и обратную связь пользователю.

Настоящая работа направлена на исследование и разработку современных технологий для решения MILP-задач с применением веб-приложений. В её рамках проводится анализ существующих методов оптимизации, разрабатывается и реализуется программное решение, позволяющее не только решать задачи оптимизации, но и проводить анализ чувствительности решения к изменениям исходных параметров модели. Ожидается, что результаты данной работы будут иметь практическое значение для предприятий, стремящихся повысить эффективность управления своими ресурсами, а также станут вкладом в развитие методов прикладной оптимизации и интеграции математического моделирования в информационные системы.

Таким образом, разработка веб-приложения для решения задач смешано-целочисленного линейного программирования является актуальной с научной и практической точек зрения, способствуя созданию удобных инструментов для оптимизации и принятия обоснованных решений в условиях сложных и динамичных производственных процессов.

# **1 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ РЕШЕНИЯ ЗАДАЧ СМЕШАНО-ЦЕЛОЧИСЛЕННОГО ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ**

## **1.1 Веб-приложение**

В последние десятилетия информационные технологии кардинально изменили способы взаимодействия человека с вычислительными системами. Одним из главных достижений этого периода стало появление и массовое распространение веб-приложений. Веб-приложение – это программный продукт, который работает в сети Интернет и предоставляет пользователю функциональность в виде интерактивного интерфейса, доступного через стандартный веб-браузер. В отличие от традиционных настольных программ, веб-приложения не требуют установки на устройство пользователя, что значительно упрощает их развёртывание, обновление и масштабирование.

Исторически развитие веб-приложений началось с простой публикации статических веб-страниц на сервере. С появлением технологий CGI (Common Gateway Interface) и ранних серверных языков, таких как Perl и ASP, началась эра динамических сайтов, способных генерировать содержимое в зависимости от пользовательского ввода. В дальнейшем, с появлением языка JavaScript и фреймворков на стороне клиента, а затем серверных платформ вроде PHP, Python, Ruby, Java, веб-приложения эволюционировали в полнофункциональные системы, способные удовлетворять самые разнообразные потребности – от электронной коммерции до корпоративных систем управления.

Ключевым отличием веб-приложений является их распределённая архитектура. Обычно такие системы состоят из фронтенда, реализованного с использованием HTML, CSS и JavaScript, а также бэкенда, который отвечает за бизнес-логику, хранение данных и обработку запросов пользователей. Благодаря этому разделению разработчики могут обновлять или масштабировать как интерфейс, так и серверную часть независимо друг от друга. В последние годы активно развивается концепция микросервисной архитектуры, когда система разбивается на самостоятельные, слабо связанные компоненты, что повышает отказоустойчивость и упрощает поддержку крупных приложений.

Существуют различные виды веб-приложений, каждый из которых имеет свои особенности:

* Одностраничные приложения (Single Page Applications, SPA) – позволяют изменять содержимое страницы без её полной перезагрузки. Это обеспечивает более плавное и отзывчивое взаимодействие с пользователем. К числу популярных технологий относятся React, Angular, Vue.js и другие.
* Традиционные многостраничные приложения – работают по классической схеме, когда каждый запрос пользователя приводит к загрузке новой HTML-страницы. Такой подход прост в реализации, однако может уступать по скорости и интерактивности одностраничным приложениям.
* Гибридные приложения – сочетают в себе возможности веб-приложений и нативных мобильных решений. Такие приложения часто создаются с использованием фреймворков типа Ionic или React Native, позволяющих использовать веб-технологии для разработки мобильных программ.
* Прогрессивные веб-приложения (PWA) – это современные решения, объединяющие преимущества веб- и мобильных приложений. PWA предлагают оффлайн-режим, нотификации, возможность установки на рабочий стол и удобство обновлений, что делает их высокофункциональными и удобными для конечного пользователя.

Особенности веб-приложений многогранны. Среди основных преимуществ можно выделить:

* Доступность. Пользователю не требуется установка специального программного обеспечения, достаточно стандартного браузера.
* Кросс-платформенность. Веб-приложение работает на различных устройствах и операционных системах, что существенно расширяет аудиторию.
* Централизованное управление. Обновления и исправления применяются на сервере, и все пользователи сразу получают актуальную версию.
* Масштабируемость. Благодаря распределённой архитектуре и возможности использования облачных вычислений, веб-приложения легко масштабируются для обработки растущего объёма данных и запросов.

Однако есть и определённые минусы. Среди недостатков можно отметить:

* Зависимость от соединения с Интернетом. Для корректной работы требуется стабильный доступ к сети, что не всегда возможно в отдалённых регионах или при перебоях.
* Вопросы безопасности. Веб-приложения становятся объектом атак из-за публичного доступа, поэтому обеспечение безопасности данных и защиты от внешних угроз становится критически важным.
* Производительность. В некоторых случаях, особенно при работе с тяжелым клиентским JavaScript или большим объёмом данных, может наблюдаться снижение производительности по сравнению с нативными приложениями.

Таким образом, веб-приложения представляют собой гибкий и доступный инструмент для реализации самых разнообразных бизнес-задач, от простых сервисов до сложных корпоративных систем. Их архитектура и возможности постоянно развиваются, что позволяет интегрировать в них новые технологии и подходы, включая современные методы оптимизации и анализа, такие как задачи смешано-целочисленного линейного программирования. Именно поэтому разработка веб-приложения для решения таких задач является актуальной и перспективной областью исследований, способной существенно улучшить процессы принятия решений и оптимизации в различных сферах деятельности.

## **1.2 Задачи смешано-целочисленного линейного программирования**

Задачи смешано-целочисленного линейного программирования представляют собой класс оптимизационных задач, где целевая функция и ограничения выражены линейными зависимостями, а переменные могут быть как непрерывными, так и дискретными (обычно целочисленными). Такое смешение типов переменных позволяет моделировать реальные ситуации, в которых помимо количественных (непрерывных) решений требуется также принимать решения о выборе вариантов или о допустимости целых значений – например, решение о том, включать ли конкретный производственный цех в план или нет, или сколько единиц продукции производить в условиях ограничения на ресурсы.

Особенности MILP-задач заключаются в следующем:

* Линейность модели. Как целевая функция, так и все ограничения выражаются через линейные комбинации переменных. Это позволяет применять теоретические методы линейного программирования, но усложняется наличие целочисленных переменных.
* Смешанный характер переменных. Наличие целочисленных переменных делает задачи NP-трудными, так как пространство возможных решений дискретное и обычно экспоненциально растёт с увеличением числа таких переменных.
* Применимость для широкого спектра задач. MILP модели находят применение в логистике, планировании производства, управлении запасами, финансовом моделировании и многих других сферах, где необходимо учитывать как количественные, так и дискретные критерии.

Применение MILP-моделирования очень популярно в тех случаях, когда необходимо принять решения при наличии ограниченных ресурсов. Например, в сфере транспортной логистики MILP позволяет определить оптимальные маршруты грузоперевозок с учётом целых маршрутов и выбором из множества альтернатив, в планировании производства – оптимально распределить задания на производственные линии, где некоторые переменные представляют собой «да» или «нет» решения, а другие – непрерывные потоки материалов.

Среди методов решения MILP-задач можно выделить:

* Классический метод ветвей и границ (branch and bound), который последовательно делит большое пространство решений на подсистемы и отсеивает неэффективные ветви.
* Метод ветвей и усечений (branch and cut), представляющий собой улучшение метода branch and bound с использованием дополнительных неравенств (усечений), которые ускоряют процесс поиска.
* Различные эвристические и метаэвристические алгоритмы, такие как генетические алгоритмы или алгоритм имитации отжига, применяемые для очень больших или сложных MILP-задач, когда точные методы оказываются вычислительно затратными.

Стоит отметить, что MILP-задачи, будучи NP-трудными, в общем случае не имеют полиномиального алгоритма решения. Современные решатели, такие как CBC, SCIP, Gurobi и CPLEX, используют комбинацию различных методов, включая техники ветвей и границ, предусекционные методы и мультипликаторы Лагранжа для уменьшения исследуемого пространства, что позволяет находить оптимальные или приближённые решения за разумное время даже для относительно крупных моделей.

На практике эффективность MILP моделей определяется не только качеством математической формулировки задачи, но и выбранным решателем, а также приемами предобработки и анализа чувствительности модели. Именно поэтому разработка веб-приложения для решения таких задач требует гибкого подхода: оно должно предоставлять возможность выбора решателя, настройки параметров модели и демонстрации анализа влияния изменений входных данных на оптимальное решение.

Таким образом, MILP-проблемы являются универсальным инструментом для оптимизации в различных сферах деятельности. Они позволяют интегрировать дискретные и непрерывные решения в единую модель и, благодаря современным алгоритмам, находить эффективные решения для комплексных и многокомпонентных ситуаций, что делает их незаменимыми как в исследовательской, так и в прикладной деятельности.

# **2 АРХИТЕКТУРА ПРОЕКТА**

**2.1 Архитектура бэкенда**

Архитектура бэкенда представляет собой системное решение, обеспечивающее обработку бизнес-логики, хранение данных, взаимодействие с внешними сервисами и предоставление API для клиентских приложений. При разработке бэкенд-части существует несколько вариантов архитектурных подходов, каждый из которых имеет свои преимущества и ограничения.

Монолитная архитектура.

Суть подхода: вся бизнес-логика, обработки запросов, работа с базой данных и коммуникация с внешними сервисами реализуются в одном приложении, развёрнутом как единое целое.

Преимущества:

* Простота разработки и развёртывания, особенно для небольших и средних проектов.
* Нет дополнительных накладных расходов для межсервисных коммуникаций, всё находится в одном кодовом пространстве.
* Лёгкость отладки и тестирования за счёт наличия единой кодовой базы.

Недостатки:

* С увеличением проекта кодовая база становится сложной для поддержки и масштабирования.
* Любое изменение или обновление требует перезагрузки всего приложения, что может привести к простоям.
* Ограниченные возможности масштабирования: масштабируется вся система, даже если требуются дополнительные ресурсы только для отдельных функций.

Микросервисная архитектура.

Суть подхода: функциональность разделяется на независимые сервисы, каждый из которых отвечает за конкретную задачу (например, аутентификация, обработка оптимизационных вычислений, управление базой данных).

Преимущества:

* Возможность масштабирования каждого сервиса отдельно в зависимости от нагрузки.
* Повышенная отказоустойчивость: сбой одного сервиса не приводит к краху всей системы.
* Более гибкая разработка, так как команды могут работать независимо над различными компонентами.

Недостатки:

* Усложнение архитектуры за счёт межсервисного взаимодействия (например, через REST, gRPC или message queues).
* Усиленные требования к инфраструктуре (контейнеризация, оркестрация, мониторинг).
* Трудности при отладке и тестировании интеграционных сценариев.

В рамках настоящего проекта для решения задач смешано-целочисленного линейного программирования выбрана монолитная архитектура бэкенда. Такой выбор обусловлен следующими соображениями:

* Простота разработки. Проект ориентирован на решение узкой прикладной задачи – оптимизации MILP, поэтому реализация всей бизнес-логики на одном серверном приложении позволяет быстро воплотить функциональные требования, не усложняя архитектуру межсервисными коммуникациями.
* Централизованность функционала. В рамках монолита удобно объединить обработку загрузки файлов, построение математических моделей с помощью Pyomo, выбор и интеграцию решателей (GLPK, CBC, SCIP и т.д.), а также работу с базой данных и механизм обмена данными посредством SSE. Такой подход облегчает единое управление состоянием и упрощает отладку.
* Масштабируемость на начальном этапе. При умеренной нагрузке на систему монолитное приложение позволяет эффективно работать, а если в будущем возникнет необходимость расширения, его можно будет постепенно переписывать в микросервисную архитектуру.

Таким образом, монолитная архитектура является оптимальным выбором для данного проекта с точки зрения скорости разработки, простоты поддержки и достаточной масштабируемости для текущих задач. В будущем, при расширении проекта, возможно разделение на микросервисы по мере необходимости, но на данном этапе интегрированное решение позволяет сфокусироваться на основной функциональности – решении MILP-задач в веб-среде.

**2.2.1 Технологии бэкенда**

В рамках реализации веб-приложения для решения задач смешано-целочисленного линейного программирования был выбран стек технологий, основанный на языке программирования Python. Это объясняется не только его выразительностью и богатой экосистемой, но и возможностью быстрого прототипирования оптимизационных моделей. В данном проекте задействованы следующие ключевые технологии:

Python.

Язык Python выбран благодаря своей универсальности и широкому спектру библиотек для численных расчетов, аналитики и веб-разработки. Его читаемость, динамическая типизация и большое сообщество разработчиков позволяют оперативно внедрять новые решения и поддерживать актуальность проекта.

В проекте используются современные версии Python, что обеспечивает совместимость с актуальными библиотеками и инструментами разработки.

Flask.

Фреймворк Flask обеспечивает легковесную и гибкую архитектуру веб-приложения. Благодаря своей минималистичной природе Flask позволяет разработчику самостоятельно выбирать необходимые компоненты и интегрировать их в единое решение.

В проекте Flask служит основой для обработки HTTP-запросов, организации маршрутизации, подключения middleware и реализации REST API, через которое осуществляется взаимодействие с фронтендом.

Преимущество Flask заключается в простоте настройки и широких возможностях расширения за счёт сторонних расширений.

Flask-CORS.

Для обеспечения корректного междоменных запросов (CORS – Cross-Origin Resource Sharing) используется расширение Flask-CORS. Оно позволяет разрешить доступ к API из различных источников, что особенно актуально при разработке распределённых систем, где фронтенд и бэкенд могут находиться на разных доменах или портах.

Это расширение минимальными усилиями устраняет проблемы, связанные с политикой одного источника браузеров, обеспечивая стабильное взаимодействие между клиентской и серверной частями.

Pyomo.

Основной инструмент для математического моделирования – это библиотека Pyomo, предназначенная для формулирования и решения задач оптимизации. Она позволяет описывать модели смешанного целочисленного линейного программирования в декларативном виде, а также интегрируется с различными решателями, такими как GLPK, CBC или SCIP.

Pyomo удовлетворяет требованиям по гибкости и масштабируемости модели, предоставляя возможность работы с линейными ограничениями, целевыми функциями и синтаксическим оформлением модели в виде объекта Python.

Решатели оптимизации (GLPK, CBC, SCIP)

По умолчанию в проекте используется решатель GLPK, однако предусмотрена возможность поддержания альтернативных решателей, таких как CBC и SCIP. Это даёт возможность пользователю выбирать наиболее подходящий инструмент в зависимости от специфики решаемой задачи.

Использование разных решателей позволяет провести сравнительный анализ эффективности, а также обеспечить более надёжное и гибкое решение MILP-задач при разных условиях.

SQLAlchemy и PostgreSQL.

Для работы с базой данных применяется SQLAlchemy – ORM-фреймворк, позволяющий абстрагироваться от специфики SQL-синтаксиса и обеспечить удобное манипулирование данными через объектную модель.

В качестве СУБД используется PostgreSQL, что обеспечивает высокую надёжность, масштабируемость и гибкость при хранении результатов оптимизационных расчётов, условий задач и логов работы приложения.

Такой подход позволяет реализовать централизованное ведение данных, а также легко интегрировать дополнительные возможности по аналитике и отчетности.

Pandas и openpyxl.

Для чтения и обработки входных Excel-файлов задействованы библиотеки Pandas и openpyxl. Pandas обеспечивает мощные инструменты работы с табличными данными, а openpyxl служит движком для чтения файлов формата xlsx.

Эти технологии позволяют корректно обрабатывать, валидировать и конвертировать данные, загруженные пользователем, что является важным этапом подготовки данных для оптимизационной модели.

Многопоточность.

В проекте используется модуль concurrent.futures с ThreadPoolExecutor для реализации фоновой обработки длительных вычислений. Такой подход позволяет освобождать основной поток сервера для обработки новых запросов, одновременно выполняя вычислительно затратные задачи в фоне.

Это позитивно сказывается на отклике системы и обеспечивает удобную реализацию SSE-уведомлений о ходе выполнения оптимизации.

В итоге выбранный стек технологий сочетает в себе проверенные временем инструменты и современные подходы, позволяющие создать надёжное и масштабируемое веб-приложение. Монолитная архитектура, построенная на Flask, демонстрирует высокую скорость разработки, централизованное управление бизнес-логикой и интеграцию с мощной библиотекой Pyomo, что позволяет реализовывать сложные MILP задачи. Использование ORM SQLAlchemy и СУБД PostgreSQL обеспечивает устойчивость хранения данных, а применение Pandas и openpyxl делает возможным оперативное считывание данных из Excel-файлов. В совокупности, данные технологии создают основу для разработки эффективного и гибкого веб-приложения, способного решать задачи оптимизации в реальном времени и удовлетворять потребности современных пользователей.

**2.2.1 Реализация серверной части**

При инициализации приложения происходит создание приложения с помощью фреймворка Flask, CORS настройка и подключение к базе данных PostgreSQL. Адрес СУБД берется из переменной окружения DATABASE\_URL. Модель таблицы tasks в базе данных имеет следующий вид:

* Поле id – id записи в таблице.
* Поле task\_id – id задачи, который назначается каждой задачи, при получении условий на сервере.
* Поле conditions – исходное условие задачи. Формат хранения – JSON.
* Поле solution – решение задачи. Формат хранения – JSON. Может произойти, например, ошибка, тогда решатель не вернет нам валидное решение, поэтому это поле в таблице может быть пустым.
* Поле solver – решатель, который выбрал пользователь.
* Поле conditions\_excel - поле для хранения сгенерированного Excel файла в виде base64 строки.
* Поле upload\_time - время загрузки задачи на сервер.
* Поле solve\_time – время завершения решения задачи.
* Поле canceled – булевый флаг, который символизирует об отмене пользователем решения задачи.

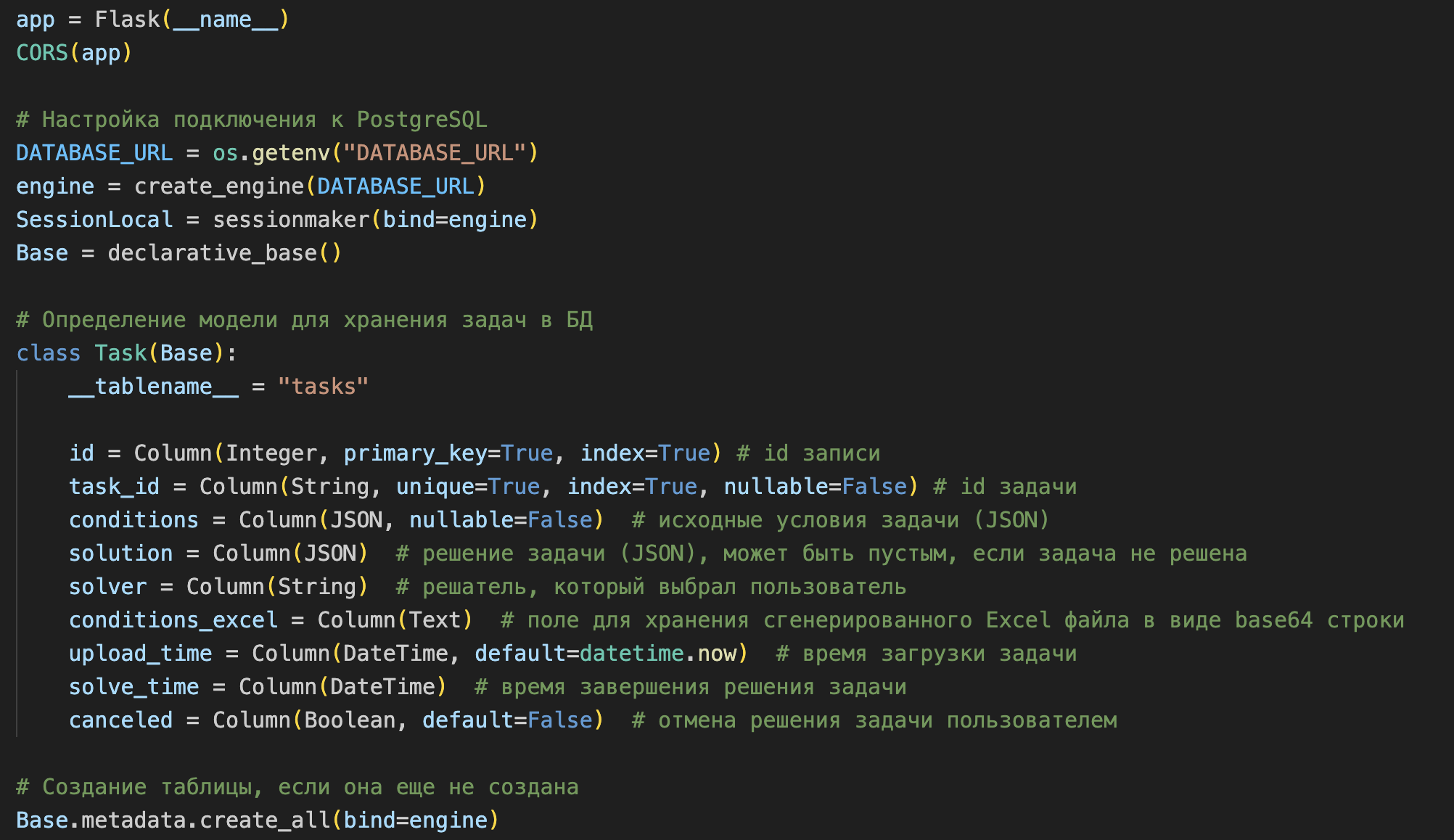


Рисунок 1 — Создание приложения и первичная настройка

Для загрузки задачи, которая была создана с помощью полей ввода на сайте, создан эндпоинт «/task» и методом POST. В функции обработчике этого эндпоита происходит извлечение из запроса данных о самой задаче и о том какой решатель будет использоваться. Далее из данных о задаче создается модель, чтобы далее передать ее в выбранный решатель библиотеки Pyomo. Также в функции происходит генерация task\_id, конвертация условий в формат excel, запись всей текущей информации в базу данных и вызов фоновой функции решателя. Если все исполнилось корректно, то возвращаем ответ, в котором содержится task\_id, чтобы клиент мог опрашивать сервер о статусе выполнения конкретной задачи.

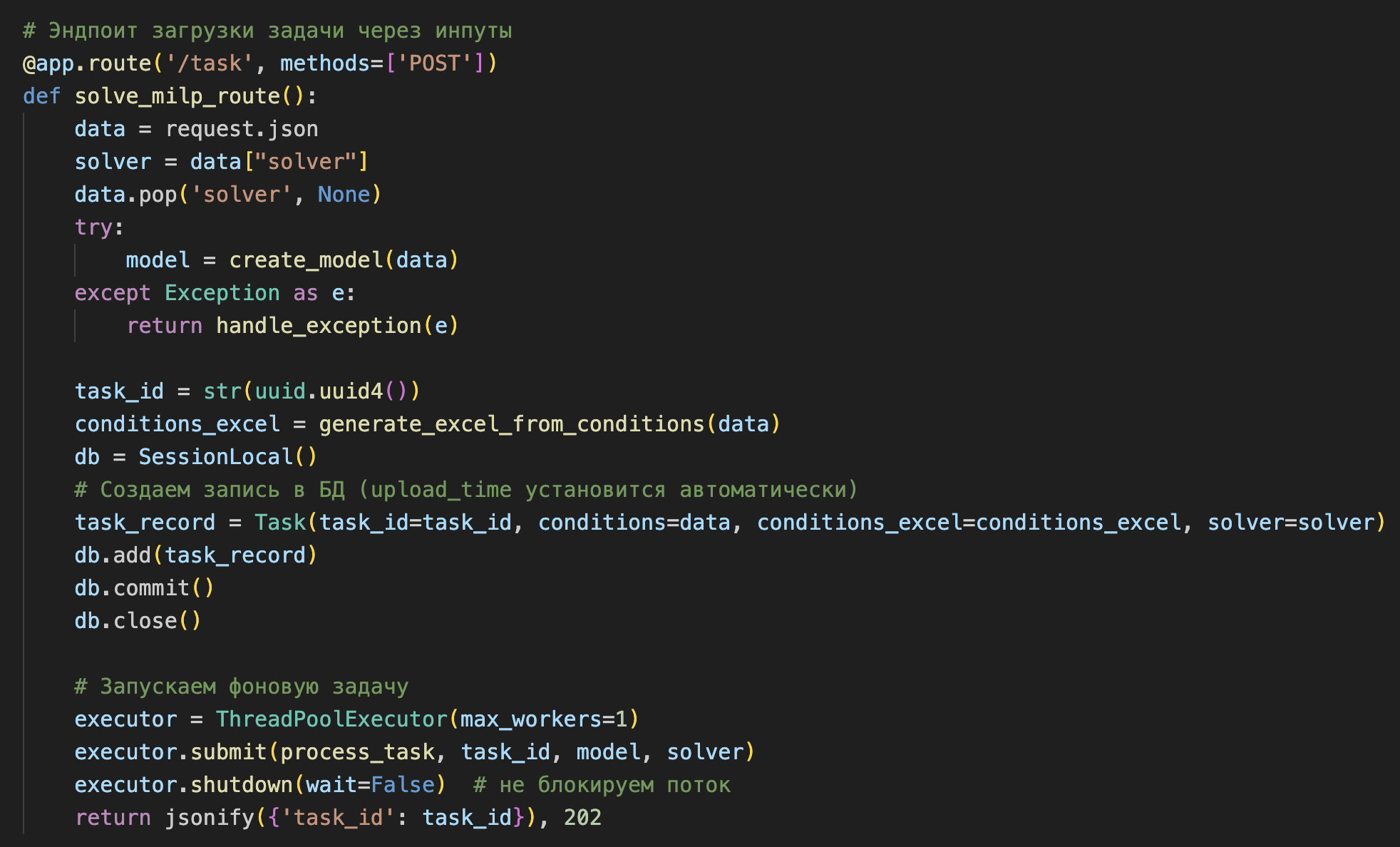


Рисунок 2 — Эндпоинт загрузки задачи через инпуты

Еще один способ загрузки задачи – с помощью файла формата xlsx. Для этого был разработан отдельный эндпоинт, доступный по пути «/task/excel», с методом POST. В функции обработчике происходит парсинг данных с помощью pandas, валидация данных, конвертация в json формат, создание модели, конвертация условия в base64 строку, запись всей текущей информации в базу данных и вызов той же фоновой функции решателя, что и в предыдущем эндпоинте. Если все прошло успешно, то возвращается task\_id задачи.

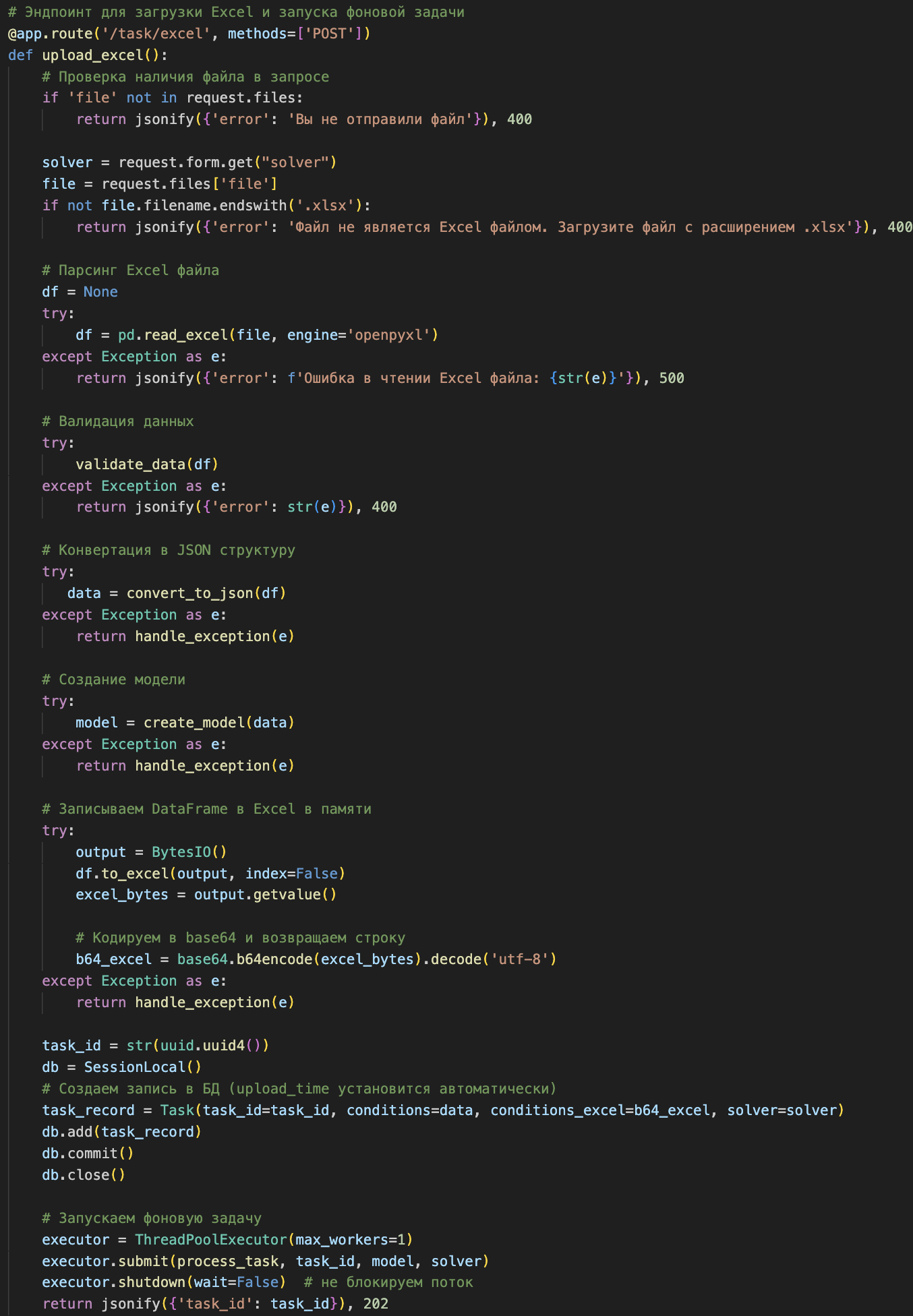


Рисунок 2 — Эндпоинт загрузки задачи через файл

Для получения информации о текущем статусе решении задачи, доступен эндпоинт «/task/task\_progress/<task\_id>» с методом GET. Клиент создает SSE соединение по этому адресу и получает каждую секунду состояние решения задачи. Для этого в функции event\_stream каждую секунду происходит обращение к базе данных по task\_id, если время решения не установлено, значит задача в процесс решения, и на клиент отправляется соответствующее сообщение. Если solve\_time известен, то на клиент отправляется информация о решении, условии задачи, решатели, которым производилось решение и времени решении. В случае обнаружения ошибки, она отправляется на клиент.

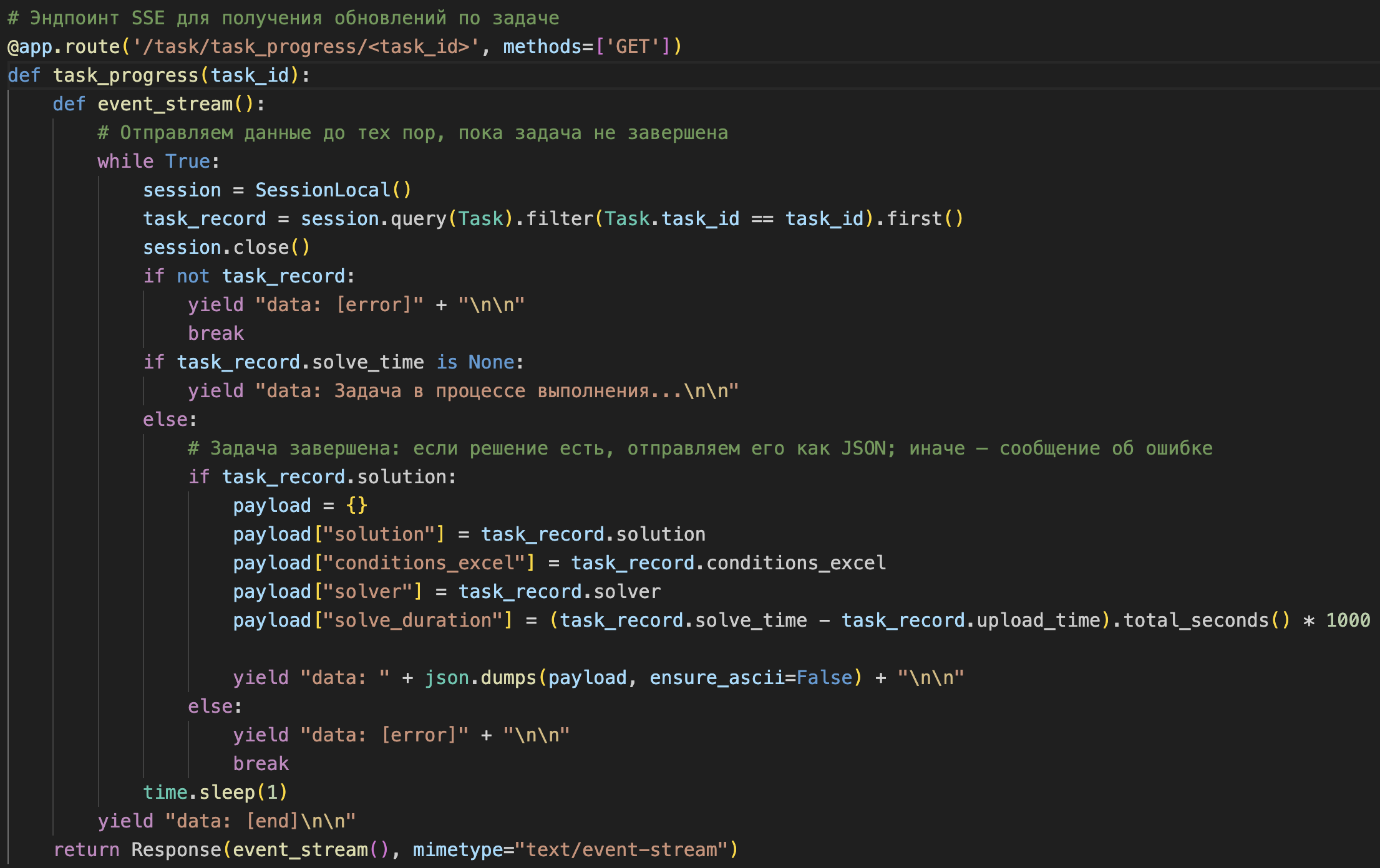


Рисунок 3 — Эндпоинт загрузки задачи через файл

У пользователя есть возможность остановить решение загруженной задачи, например, если его не устраивает, что задача решается слишком долго. Для этого создан эндпоинт «/task/cancel\_task/<task\_id>» c методом POST. Его функционал заключается в установлении булевого флага поля canceled в значение True. После чего клиенту возвращается сообщение об успешности отмены решения.

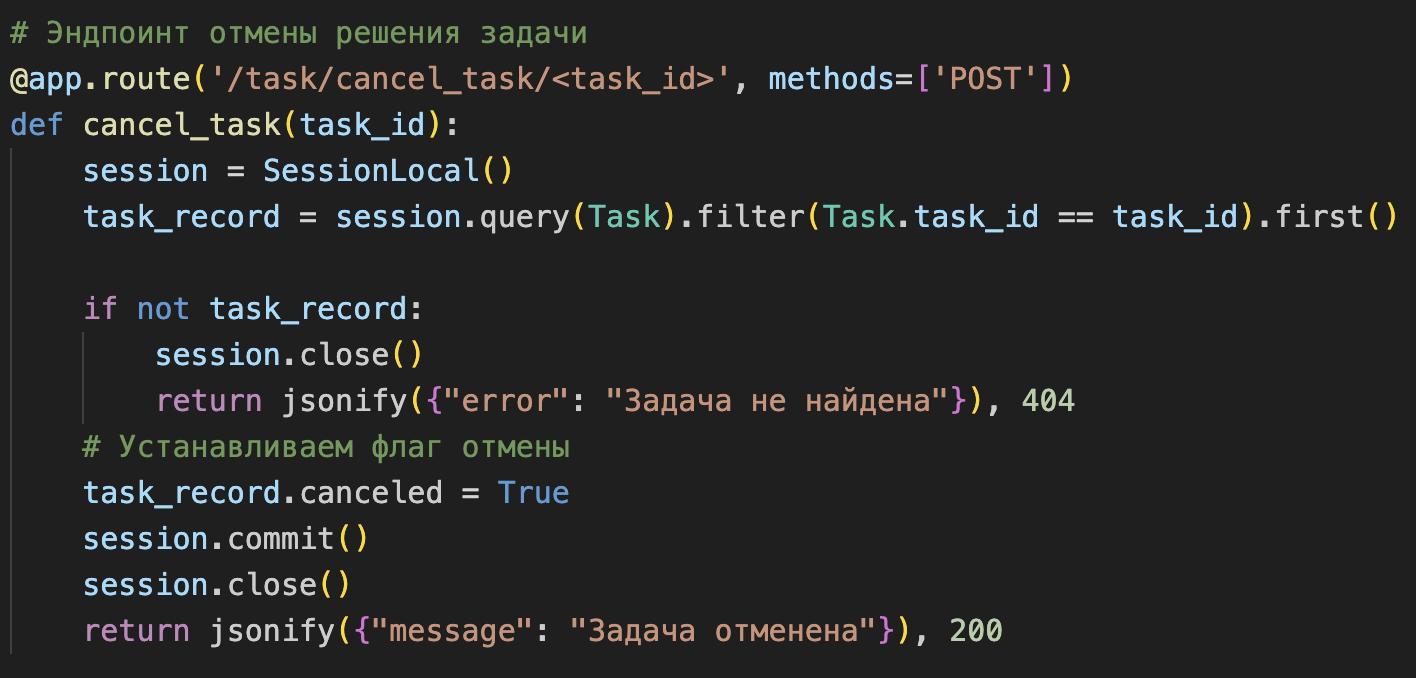
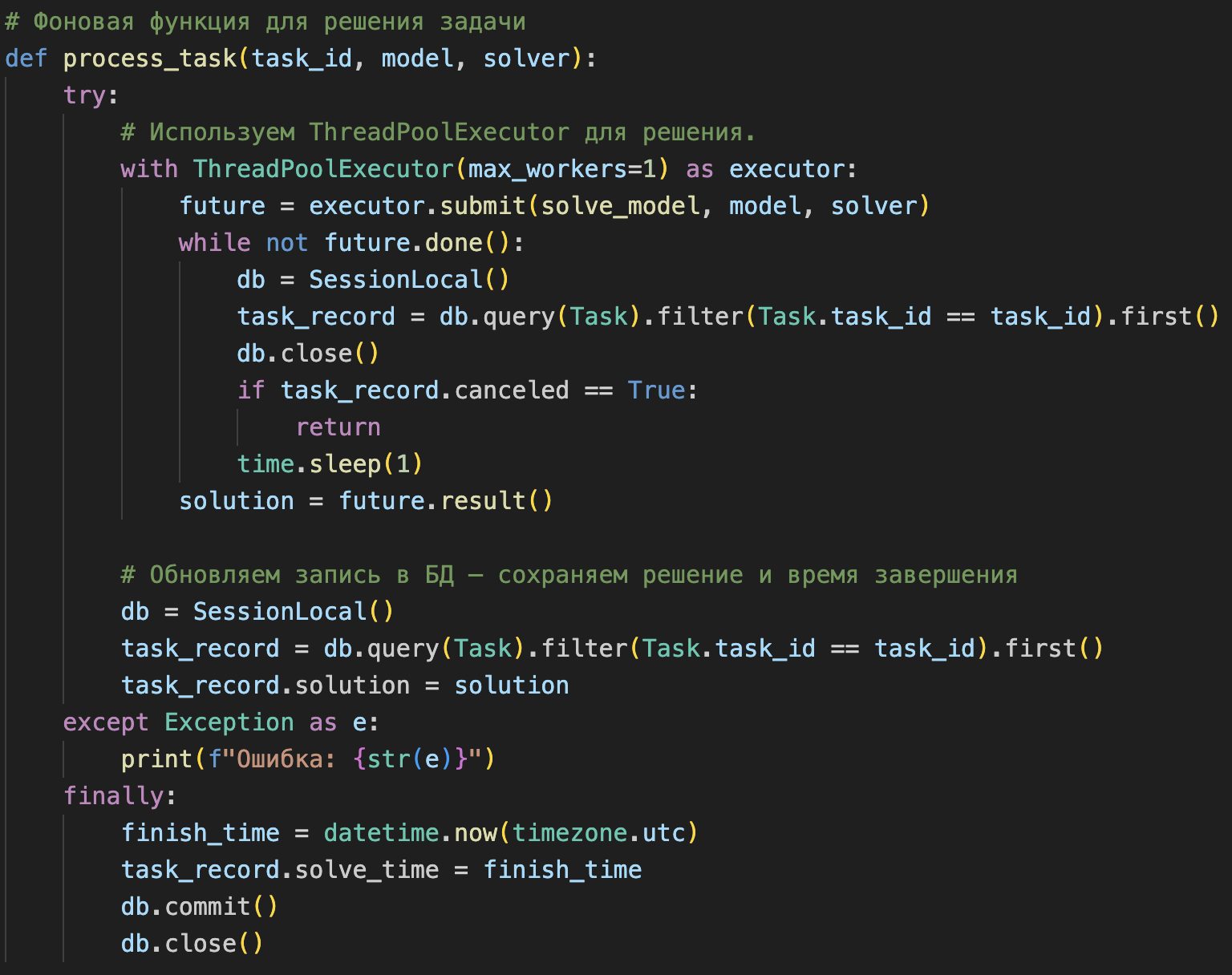


Рисунок 3 — Эндпоинт отмены решения задачи

Непосредственно остановка решения происходит в фоновой функции решения - в цикле, пока решение задачи не завершено, проверяем у текущей задачи в таблице поле canceled. Если флаг равен True, то мы останавливаем решение. Если решение не прервали, мы сохраняем результаты в базу данных. При возникновении ошибки логируем ее.

 Рисунок 4 — Фоновая функция решения задачи

# **3 ОПИСАНИЕ РАБОТЫ**

## **3.1 Реализованная архитектура**

По окончании разработки серверной части получили систему из 9 микросервисов. Для запуска сервера используется Docker, вся серверная часть приложения занимает 17 контейнеров — 9 контейнеров отведены под сами сервисы, еще 8 — под их зависимости (базы данных различного типа). Четыре микросервиса из девяти отведены под операции над основными сущностями: core-сервис, сервис объявлений, сервис сделок, сервис уведомлений. У каждого из них есть отдельная база данных. Это необходимо, чтобы отказ одной базы данных не повлек за собой аварию более чем на одном сервисе, например, при аварии на сервисе сделок пользователь все еще смог бы просматривать объявления. Таким образом, приложение разделено на микросервисы с целью минимизации потери функционала при отказе одного сервиса, например, из-за неудачного релиза или из-за высокой нагрузки.

Схема архитектуры приложения приведена в приложении А, ссылка на репозиторий с исходным кодом приведена в приложении Б.

## **3.2 Взаимодействие с клиентом**

Из 9 микросервисов 4 имеют собственные HTTP-серверы для непосредственного взаимодействия с фронтендом:

* сервис регистрации;
* сервис аутентификации;
* API-сервис;
* сервис изображений.

Для удобства фронтенд-разработчика у каждого из этих сервисов есть Swagger-документация, которая позволяет взаимодействовать с API в удобном формате. Swagger-документация полезна бэкенд-разработчику при тестировании разработанного API и фронтенд-разработчику при ознакомлении с функционалом API, так как содержит описания всех возможных запросов, в том числе их адреса, методы, query- и path-параметры, форматы тел запроса и ответа. На рисунке 7 продемонстрирован интерфейс Swagger-документации.

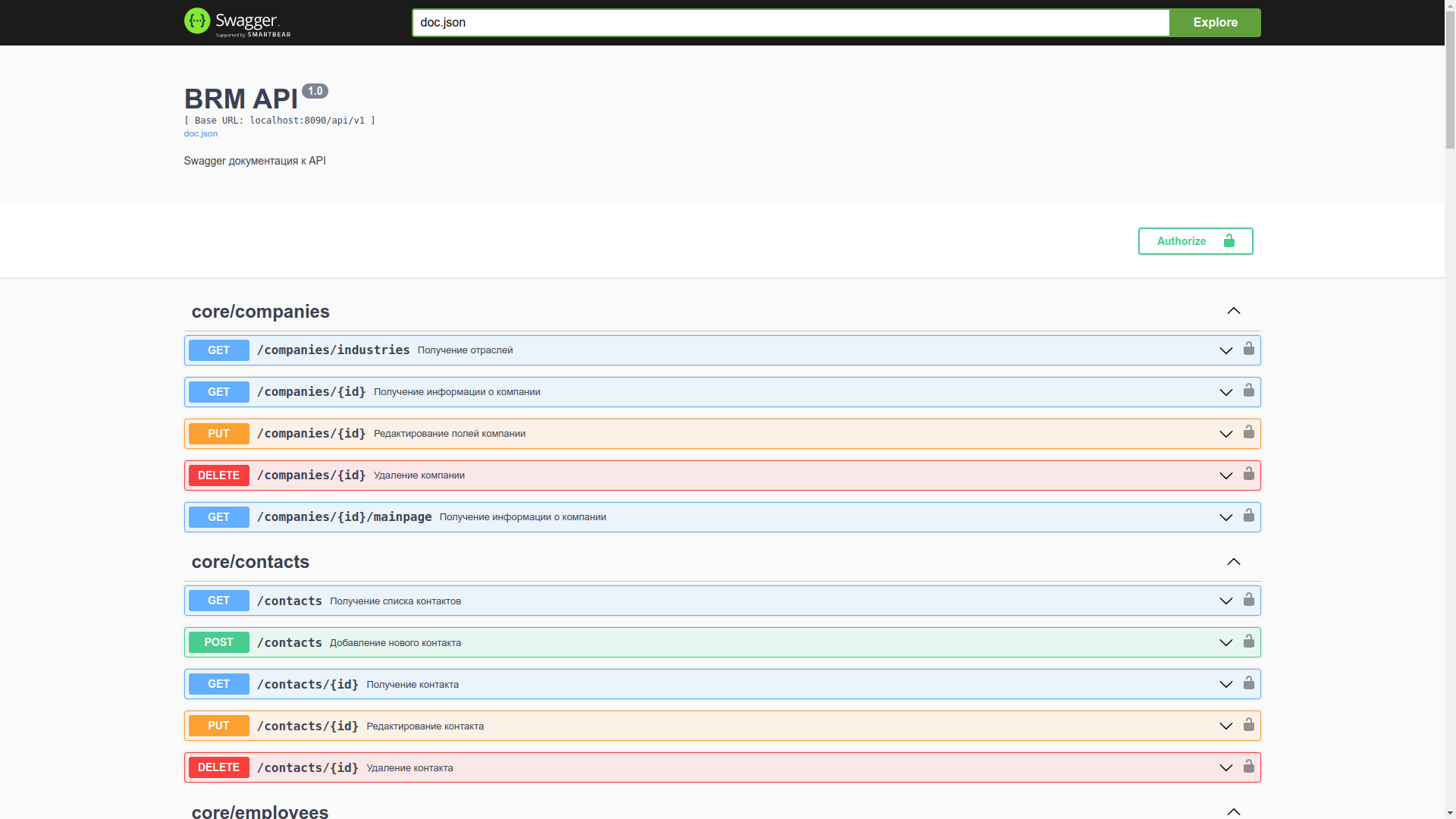


Рисунок 7 — Интерфейс Swagger-документации

### 3.2.1 Сервис регистрации

Процедуру регистрации проходят все новые клиенты BRM. Для прохождения регистрации не нужно иметь токен доступа, так как он выдается только существующим пользователям. Сервис регистрации поддерживает два типа запросов:

* добавление новой компании и владельца;
* получение списка отраслей, которые могут быть у компании.

### 3.2.2 Сервис аутентификации

С сервисом аутентификации фронтенд взаимодействует, когда пользователь вводит свои логин и пароль для входа в систему, или когда необходимо обновить access-токен пользователя. В этом случае взаимодействие с сервисом происходит незаметно для пользователя.

HTTP-сервер сервиса аутентификации поддерживает следующие запросы:

* получение пары токенов;
* удаление refresh-токена;
* обновление токенов.

### 3.2.3 API-cервис

Этот сервис предоставляет функционал для управления всеми компонентами CRM-системы. Для его использования пользователю необходим токен доступа, который можно получить в сервисе аутентификации. Токен необходимо передавать с каждым запросом в заголовке «Authorization» и с префиксом «Bearer ».

Запросы к API-сервису можно разделить на 6 логических частей:

* управление компаниями;
* управление контактами;
* управление сотрудниками;
* управление сделками;
* управление объявлениями;
* управление уведомлениями.

Для контактов, сотрудников и объявлений доступны все CRUDL-операции. Для компаний доступны операции получения, редактирования и удаления информации, создать компанию с помощью API-сервиса нельзя, для этого есть сервис регистрации. Для сделок доступны операции получения и редактирования, создать сделку с помощью API нельзя, так как сделка создается автоматически при отклике на объявление. Для управления уведомлениями в API доступны операции просмотра в виде списка или по отдельности и операция подтверждения закрытия сделки.

### 3.2.4 Сервис изображений

Взаимодействие фронтенда с этим сервисом происходит незаметно для пользователя каждый раз, когда пользователь запрашивает информацию, содержащую изображения.

HTTP-сервер сервиса изображений поддерживает следующие запросы:

* добавление изображения;
* получение изображения.

# **ЗАКЛЮЧЕНИЕ**

В рамках выполнения выпускной квалификационной работы бакалавра была разработана CRM-система, обеспечивающая взаимодействие компаний друг с другом. Компании различных размеров получили площадку для размещения объявлений и торговли друг с другом, что позволяет отнести разработанное приложение в сферу B2B. Тем не менее, приложение сохраняет историю всех сделок компании, предлагает некоторые аналитические функции и обеспечивает коммуникацию компании и ее клиентов, поэтому его можно классифицировать как CRM-систему.

Приложение разработано в соответствии с принципами микросервисной архитектуры. Это означает, что поддерживается горизонтальное масштабирование сервисов и обеспечена высокая отказоустойчивость, так как отказ одного из сервисов не влияет на работоспособность всех остальных. Благодаря средству контейнеризации Docker, можно легко запустить все сервисы и зависимости приложения.

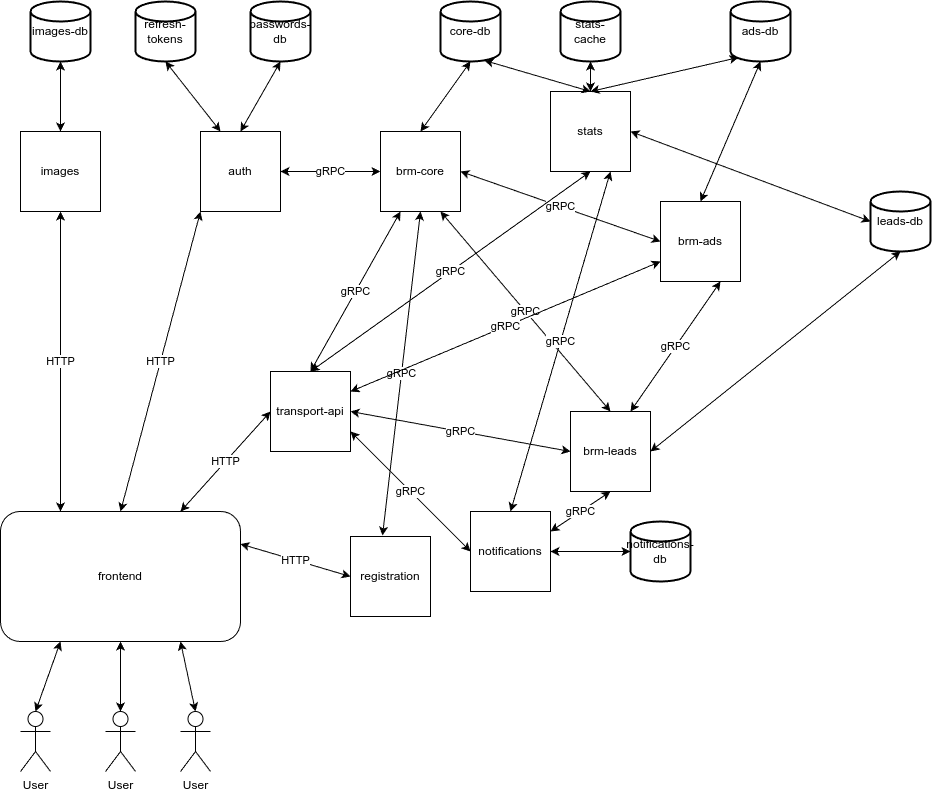
Разработку такой системы будет легко поддерживать в дальнейшем, так как каждый сервис разработан по правилам чистой архитектуры. Это значит, что внутренние компоненты сервисов могут быть легко заменены на другие в случае необходимости.

В процессе разработки был реализован весь планируемый функционал приложения. Предметная область включает в себя компанию, сотрудников, объявления и сделки. Благодаря микросервисной архитектуре, функционал приложения может быть легко расширен при необходимости.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения. // Питер, 2021. – 352 с.
2. Полное руководство по CRM-системам [Электронный ресурс]. — URL: https://www.oracle.com/cis/cx/what-is-crm/ (дата обращения 09.01.2024).
3. CRM [Электронный ресурс]. — URL: https://sendpulse.com/support/glossary/crm (дата обращения 09.01.2024).
4. Что такое CRM? [Электронный ресурс]. — URL: https://www.sap.com/central-asia-caucasus/products/crm/what-is-crm.html (дата обращения 09.01.2024).
5. CRM-система — что это и как работает [Электронный ресурс]. — URL: https://megaplan.ru/news/articles/chto-takoe-crm-sistema/ (дата обращения 09.01.2024).
6. The Best CRM Software for 2024 [Электронный ресурс]. — URL: https://www.pcmag.com/picks/the-best-crm-software (дата обращения 09.01.2024).
7. Microservices. Как правильно делать и когда применять [Электронный ресурс]. — URL: https://habr.com/ru/companies/dataart/articles/280083/ (дата обращения 02.02.2024).
8. 26 основных паттернов микросервисной разработки [Электронный ресурс]. — URL: https://cloud.vk.com/blog/26-osnovnyh-patternov-mikroservisnoj-razrabotki (дата обращения 02.02.2024).
9. A guide to Golang microservices [Электронный ресурс]. — URL: https://www.cortex.io/post/golang-microservices (дата обращения 02.02.2024).
10. Golang, Microservices, and Monorepo [Электронный ресурс]. — URL: https://dev.to/bastianrob/golang-microservices-and-monorepo-gp3 (дата обращения 02.02.2024).

# **ПРИЛОЖЕНИЕ А Схема архитектуры проекта**

****

# **ПРИЛОЖЕНИЕ Б QR-код репозитория с исходным кодом**

****