  
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

(ДВФУ)

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ (ШКОЛА)**

**Департамент программной инженерии и искусственного интеллекта**

РАЗРАБОТКА ВЕБ-СЕРВИСА «FITFUEL» С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ КОЛЛЕКТИВНОЙ ПРОМЫШЛЕННОЙ РАЗРАБОТКИ

**КУРСОВАЯ РАБОТА**

по дисциплине «Технологии коллективной промышленной разработки информационных систем» по образовательной программе подготовки бакалавров по направлению

09.03.04 «Программная инженерия» профиль «Программная инженерия»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | |  | | | Студенты группы Б9121-09.03.04 | | | | | | | | | |
|  | | | | | | | |  | | |  | | | | | Костюченко А. И. | | | | |
|  | | | | | | | |  | | | (подпись) | | | | | Зеленин К. В. | | | | |
|  | | | | | | | |  | | | (подпись) | | | | | Шешеня Е. О. | | | | |
|  | | | | | | | |  | | | (подпись) | | | | | Иванов Д. И. | | | | |
|  | | | | | | | |  | | | (подпись) | | | | |  | | | | |
|  | | | | | | | |  | |  | Руководитель: | | | | | | | | | |
|  | | | | | | | |  | |  | ст. преподаватель ДПИиИИ | | | | | | | | | |
|  | | | | | | | |  | |  | ученая степень, должность | | | | | | |  | Иваненко Ю. С. | |
|  | | | | | | | |  | |  | (подпись) | | | | | | |  | (ФИО) | |
|  | | | | | | | | | | | | | | | | | | | | |
| Регистрационный № | | | | | | | |  | |  | Защищен с оценкой | | | | | | | | | |
|  | | | | | | | |  | |  |  | | | | | | | | | |
|  | | | | |  |  | |  | |  | « | |  | » |  |  | | | | 2024 г. |
| (подпись) | | | | |  | (ФИО) | |  | |  |  | | | | | | | | | |
| « | |  | » |  | | | 2024 г. |  | |  |  | | | | | | | | | |

г. Владивосток  
2025

Оглавление

[Введение 3](#_Toc185188732)

[1. Разработка плана проекта 5](#_Toc185188733)

[2. Разработка регламента проведения инспекции 6](#_Toc185188734)

[3. Разработка модели состояний задач 9](#_Toc185188742)

[4. Разработка презентации проекта 10](#_Toc185188745)

[5. Разработка требований к проекту 14](#_Toc185188746)

[6. Разработка архитектуры проекта 26](#_Toc185188747)

[7. Разработка измерений проекта 29](#_Toc185188748)

[8. Разработка перечня задач проекта 31](#_Toc185188749)

[9. Разработка рекомендаций по кодированию 33](#_Toc185188750)

[10. Разработка плана тестирования проекта 36](#_Toc185188751)

[Заключение 40](#_Toc185188752)

[Список литературы 41](#_Toc185188753)

# Введение

В современном мире, где вопросы здоровья и правильного питания становятся всё более актуальными, веб-сервис FitFuel предлагает удобное решение для тех, кто хочет взять под контроль свой рацион. FitFuel — это инновационный инструмент, предназначенный для подсчёта калорий и создания персонализированных меню с учётом индивидуальных потребностей пользователей.

Программный продукт FitFuel идеально подходит для следующих категорий пользователей:

1. **Тех, кто хочет похудеть**: Приложение помогает создать меню с учётом целевых калорий и пищевых предпочтений, что облегчает процесс снижения веса и делает его более эффективным;
2. **Спортсменов и фитнес-энтузиастов**: FitFuel предоставляет возможность подобрать оптимальное меню для поддержания энергии на тренировках и быстрого восстановления после физических нагрузок;
3. **Людей с особыми диетическими потребностями**: Приложение учитывает различные диетические ограничения, такие как веганство, безглютеновая диета, пищевые аллергии, и предлагает соответствующие варианты меню;
4. **Тех, кто хочет сэкономить время и деньги**: Сервис помогает планировать питание на основе бюджета и предпочтений, исключая необходимость долгих раздумий о том, что приготовить;
5. **Людей, стремящихся к здоровому образу жизни**: FitFuel предоставляет подробную информацию о питательной ценности продуктов, что способствует созданию сбалансированного рациона для улучшения общего состояния здоровья.

Разработка FitFuel строится на принципах удобства, точности и персонализации. Приложение включает функционал автоматического расчёта калорийности, учета диетических ограничений. Такой подход обеспечивает комплексное решение задач, связанных с питанием и здоровьем, делая процесс достижения целей пользователя простым и комфортным.

Целью проекта FitFuel является создание универсального инструмента для управления рационом, который будет полезен как отдельным пользователям, так и профессионалам в области диетологии и фитнеса. В процессе разработки используются современные технологии, обеспечивающие высокую производительность, стабильность и безопасность приложения, что делает его надёжным помощником для пользователей, стремящихся к улучшению своего здоровья и качества жизни.

Для достижения поставленной цели необходимо решить следующие задачи:

* разработать план проекта;
* разработать регламент проведения инспекции;
* разработать модель состояний задач;
* разработать презентацию проекта;
* разработать требования к проекту;
* разработать архитектуру проекта;
* разработать измерения проекта;
* разработать перечь задач проекта;
* разработать рекомендации по кодированию;
* разработать план тестирования проекта.

# 1. Разработка плана проекта

План проекта – это документ, содержащий подробную информацию о проекте: исполнителях, задачах и сроках. Документ является конечным результатом этапа планирования, утверждается до начала любых работ и становится самым главным и достоверным источником информации о грядущем проекте.

В нашем случае исполнителями являются следующие лица:

* Team Leader, Coder, Build Engineer – Костюченко Антон;
* Coder, Technical – Зеленин Кирилл;
* **Frontend Developer** – Шешеня Евгений;
* **Technical Writer, Frontend Developer – Иванов Дмитрий.**

На рисунке 1 представлен перечень задач для выполнения и примерные сроки их реализации.

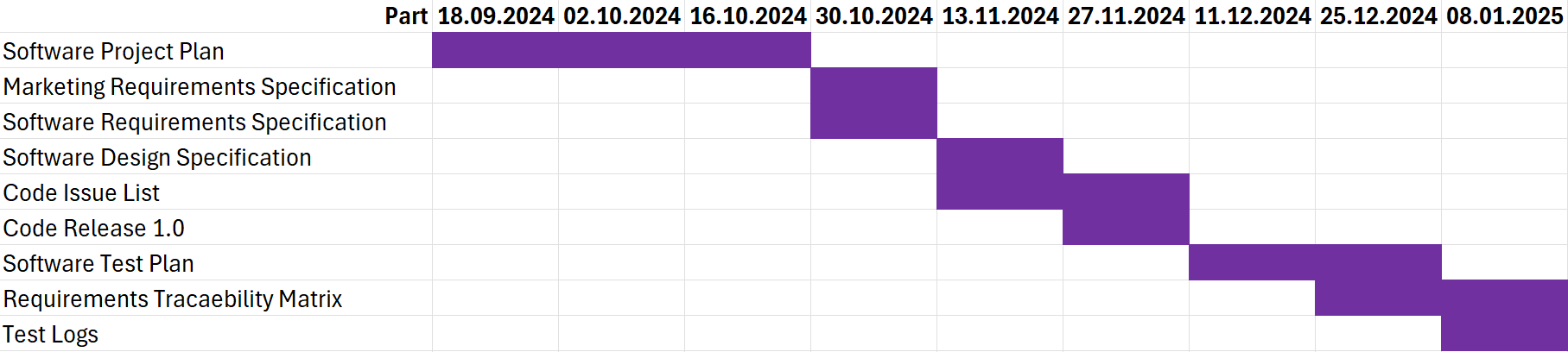


Рисунок 1 – план проекта

# 2. Разработка регламента проведения инспекции

Верификация рабочих продуктов является неотъемлемой частью процесса по обеспечению их качества. Современной технологией программирования выработаны специальные стандарты, подходы и механизмы проведения верификаций рабочих продуктов в формате так называемых инспекций (peer reviews).

Инспекция – это мероприятие по обеспечению качества рабочих продуктов проектов по разработке ПО и иной деятельности, которая проводится разработчиками, возможно - с участием представителей заказчика.

Концептуально инспекция имеет следующие цели:

* обнаружить ошибки в функциях, логике, содержании или реализации рабочих продуктов на ранних этапах их разработки и предотвратить их наследование;
* рационально донести замысел или реализацию продукта до всех заинтересованных лиц (через их участие);
* оптимизировать, оценить или улучшить рабочий продукт.

Критерии отнесения к формальной инспекции:

* Разработана новая версия документации или исправлена существующая документация;
* Создание pull request в master в репозиторий на платформе GitHub.

## Перечень ролей участников инспекции и их обязанности

|  |  |
| --- | --- |
| **Роль** | **Обязанности** |
| Автор | Создание инспектируемых изменений в  существующем рабочем продукте |
| Инспектор, проверяющий | Ответственность за эффективную проверку инспектируемого рабочего продукта. |
| Модератор (Тимлид) | Проведение подготовительного совещания, координация работы участников; отслеживание выполнения задач и координированной работы. |

Таблица 1 – Роли участников инспекции

Роли не фиксированы, за исключением одной – Модератор, модератором выступает Костюченко Антон.

## Этапы инспекции:

1. Инициация – создание автором запроса на внесение изменений в продукт и назначение инспектора.
2. Подготовка и непосредственная инспекция – анализ изменений и выявление недочетов.
3. Подведение итогов – вердикт о внесении изменений.

## Порядок организации

Автор выгружает код продукта на GIT и приглашает Инспектора. Далее Инспектор готовится к инспекции и составляет протокол ошибок, которые отправляет автору. После исправления ошибок автор высылает исправленную версию инспектору.

## Порядок подготовки и проведения

Автор оповещает инспектора о проведении инспекции.

Проведение инспекции проводится в течение 4-7 дней. Затем замечания подаются в GIT, обозначая при этом их важность, если замечания имеются, то работа передается автору на внесение изменений, иначе инспекция считается успешной и завершенной.

## Перечень статусов и степени важности замечаний

1. Ошибка – важное замечание, требующее доработки в обязательном порядке.
2. Комментарий – рекомендация по улучшению проекта(не является обязательным).

## Порядок верификации учёта замечаний

После конечных изменений инспектор проверяет замечания и выносит вердикт о продолжении выполнении проекта или наоборот выносит новые замечания.

## Метрики, характеризующие эффективность инспекций

**Эффективность устранения дефектов (Defect Removal Efficiency)**

DRE характеризует эффективность инспекции. Чем больше DRE, тем эффективнее инспекция.

* **Стратегическая цель метрики** – повысить качество разрабатываемого ПО за счёт эффективного устранения дефектов на ранних этапах разработки;
* **Изучаемый объект метрики** – процесс устранения дефектов;
* **Измеряемый атрибут** – процент устранённых дефектов от общего числа найденных;
* **Единица измерения** – процент (%);
* **Цель предприятия** – повышение DRE: чем выше DRE, тем лучше, так как это свидетельствует о высоком уровне эффективности работы с обнаруженными дефектами.

# 3. Разработка модели состояний задач

Каждая задача, являясь отражением делового процесса, проходит определенные состояния. Сначала идет создание задачи, потом идет выполнение работ по задаче, после выполнения задача завершается.

## Перечень состояний задач:

* Created – задача создана и зарегистрирована в системе. Ожидает назначения исполнителя;
* Assigned – задача назначена конкретному исполнителю или команде. Исполнитель уведомлен и готовится к выполнению;
* In Progress – задача находится в процессе выполнения;
* Under Review – работа по задаче завершена, результат передан на проверку ответственному лицу;
* Rework – задача возвращена на доработку после проверки;
* Completed – задача успешно выполнена, проверена и утверждена;
* Cancelled – задача отменена без выполнения из-за изменений требований или ненадобности;

## Правила перехода задачи из состояния в состояние

- В качестве системы отслеживания используется GitHub.

- Созданная задача имеет состояние «Created».

- Тимлид назначает исполнителя и инспектора (по желанию любой участник может взять роль исполнителя на себя) – «Assigned».

- Когда участник приступил к работе задаче присваивается статус «In Progress»

- После окончания работы результат передается ответственному лицу и состояние меняется на «Under Review».

- Если присутствуют какие-либо недочеты, то задаче присваивается статус «Rework» и вновь становится «In Progress».

- Если задача успешно выполнена и проверена, то статус «Completed».

- В случае изменения требований к задаче или за ее ненадобностью задача становится неактивной и присваивается статус «Cancelled»

# 4. Разработка презентации проекта

На рисунке 2 представлена титульная страница презентации, а также состав команды разработчиков.

Изображение выглядит как текст, снимок экрана, графический дизайн, Графика

Автоматически созданное описание

Рисунок 2 – Титульная страница

Описание проекта на рис. 3 – 4.

Изображение выглядит как текст, фрукт, яблоко, Продовольственная группа

Автоматически созданное описание

Рисунок 3 – описание

Изображение выглядит как текст, обувь, Фитнес, спорт

Автоматически созданное описание

Рисунок 4 – продолжение описания

Преимущества калькулятора калорий расположены на рисунке 5.

Изображение выглядит как текст, снимок экрана, визитная карточка, Шрифт

Автоматически созданное описание

Рисунок 5 – преимущества калькулятора

Преимущества подборщика еды/диет/блюд расположены на рисунке 6.

Изображение выглядит как текст, овощи, производить, Веганское питание

Автоматически созданное описание

Рисунок – преимущества подборщика

Варианты улучшения веб-сервиса (параметры пользователя) представлены на рис. 7

Изображение выглядит как текст, человек, Человеческое лицо, снимок экрана

Автоматически созданное описание

Рисунок 7 – варианты улучшения (параметры пользователя)

Варианты улучшения (обратная связь) находятся на рис. 8

Изображение выглядит как текст, снимок экрана, млекопитающее

Автоматически созданное описание

Рисунок 8 – варианты улучшения (обратная связь)

# 5. Разработка требований к проекту

Программный продукт «FitFuel» состоит из следующих подсистем:

1. Пользовательский интерфейс (UI - User Interface);
2. Подсистема подсчета калорий (CC - Calorie Counting) ;
3. Подсистема управления пользователями и личными кабинетами (UM - User Management);
4. Подсистема подбора меню (MM - Menu Management);
5. Подсистема интеграции и API (IA - Integration and API).

**Требования к подсистеме «Пользовательский интерфейс»**

*Требование REQ\_UI\_001*

В шапке сайта должны быть расположены следующие элементы:

* Логотип веб-сервиса, который является ссылкой на главную страницу;
* Кнопки навигации: "Главная", "Подбор меню", "Личный кабинет";
* Кнопка "Вход" или "Регистрация" для неавторизованных пользователей;
* Имя пользователя и кнопка "Выход" для авторизованных пользователей.

Кнопки навигации должны быть легко доступны, визуально выделены и иметь интуитивно понятные названия.

*Требование REQ\_UI\_002*

На странице "Главная" должен быть расположен калькулятор калорий, занимающий большую часть страницы. Калькулятор должен содержать следующие элементы:

* Поле для ввода названия продукта или блюда с автодополнением и возможностью выбора из списка;
* Поле для ввода количества продукта или порций блюда с возможностью выбора единиц измерения;
* Кнопка "Рассчитать" для запуска расчета калорийности и питательных веществ;
* Блок с результатами расчета, отображающий калорийность, белки, жиры, углеводы и другие доступные параметры;
* Блок с рекомендациями по потреблению продукта или блюда, основанными на целях и предпочтениях пользователя;
* Кнопка "Добавить в дневник" для сохранения результатов расчета в истории пользователя.

*Требование REQ\_UI\_003*

На странице "Подбор меню" должна быть форма для ввода параметров подбора меню, содержащая следующие элементы:

* Поле для ввода желаемого бюджета на меню с возможностью выбора валюты;
* Поле для ввода желаемой калорийности меню с возможностью выбора единиц измерения;
* Выбор времени приема пищи (завтрак, обед, ужин, перекус) с возможностью множественного выбора;
* Чекбоксы для указания наличия определенных продуктов или ингредиентов;
* Чекбокс "Веганское меню";
* Чекбокс "Учитывать аллергии и предпочтения из личного кабинета";
* Кнопка "Подобрать меню" для запуска процесса подбора меню;
* Блок с результатами подбора меню, отображающий предложенные блюда, их ингредиенты, калорийность и стоимость;
* Возможность сохранения подобранного меню в личном кабинете или экспорта в PDF.

*Требование REQ\_UI\_004*

На странице "Личный кабинет" должна быть форма для ввода и редактирования информации о пользователе, содержащая следующие элементы:

* Поля для ввода персональных данных (имя, возраст, пол, рост, вес) с валидацией и подсказками;
* Поля для ввода предпочтений в еде и аллергий с возможностью выбора из списка и добавления своих вариантов;
* Выбор цели (набор массы, сброс веса, поддержание веса) с возможностью указания желаемого темпа изменения веса.

**Требования к подсистеме «Подсчет калорий»**

*Требование REQ\_CC\_001*

Подсистема должна обеспечивать расчет калорийности и питательных веществ (белки, жиры, углеводы) для продуктов и блюд на основе данных, введенных пользователем.

*Требование REQ\_CC\_002*

Подсистема должна иметь базу данных продуктов и блюд с информацией о их калорийности и содержании питательных веществ. База данных должна включать:

* Название продукта или блюда;
* Калорийность на 100 граммов или на порцию;
* Содержание белков, жиров и углеводов на 100 граммов или на порцию;
* Дополнительную информацию (витамины, минералы, клетчатка и т. д.) при наличии.

*Требование REQ\_CC\_003*  
Подсистема должна предоставлять возможность поиска продуктов и блюд в базе данных по названию или ключевым словам. Результаты поиска должны отображаться в виде списка с возможностью выбора нужного элемента.

*Требование REQ\_CC\_004*

Подсистема должна обеспечивать возможность ввода пользовательских продуктов и блюд, отсутствующих в базе данных. Пользователь должен иметь возможность указать:

* Название продукта или блюда;
* Калорийность на 100 грамм или на порцию;
* Содержание белков, жиров и углеводов на 100 грамм или на порцию;
* Дополнительную информацию (витамины, минералы, клетчатка и т.д.) при наличии.

*Требование REQ\_CC\_005*

Подсистема должна производить расчет калорийности и питательных веществ для указанного количества продукта или порций блюда. Результаты расчета должны отображаться в удобном для восприятия виде (таблица, график, диаграмма).

*Требование REQ\_CC\_006*

Подсистема должна предоставлять возможность сохранения результатов расчета в истории пользователя. Сохраненные записи должны включать:

* Дату и время расчета;
* Название продукта или блюда;
* Количество продукта или порций блюда;
* Результаты расчета (калорийность, белки, жиры, углеводы).

*Требование REQ\_CC\_007*

Подсистема должна обеспечивать возможность просмотра и анализа истории расчетов пользователя. Пользователь должен иметь возможность:

* Просматривать список сохраненных записей;
* Фильтровать записи по дате, времени приема пищи или типу продукта/блюда;
* Просматривать суммарную калорийность и содержание питательных веществ за выбранный период;
* Экспортировать историю расчетов в формате PDF или CSV.

*Требование REQ\_CC\_008*

Подсистема должна предоставлять рекомендации по потреблению продуктов и блюд на основе целей и предпочтений пользователя, указанных в личном кабинете. Рекомендации могут включать:

* Суточную норму потребления калорий и питательных веществ;
* Распределение калорий и питательных веществ по приемам пищи;
* Предупреждения о превышении нормы потребления отдельных веществ (например, сахара или насыщенных жиров).

**Требования к подсистеме «Управление пользователями и личными кабинетами»**

*Требование REQ\_UM\_001*Подсистема должна обеспечивать возможность регистрации новых пользователей. При регистрации пользователь должен указать:

* Имя пользователя (логин);
* Адрес электронной почты;
* Пароль;
* Подтверждение пароля.

Подсистема должна проверять корректность введенных данных и наличие уже зарегистрированных пользователей с таким же логином или адресом электронной почты.

*Требование REQ\_UM\_002*

Подсистема должна обеспечивать возможность авторизации зарегистрированных пользователей. Авторизация должна осуществляться по логину и паролю. После успешной авторизации пользователь должен получать доступ к своему личному кабинету и всем функциям веб-сервиса, доступным для авторизованных пользователей.

*Требование REQ\_UM\_003*

Подсистема должна обеспечивать возможность восстановления забытого пароля. Пользователь должен иметь возможность запросить восстановление пароля, указав свой адрес электронной почты. На указанный адрес должна быть отправлена ссылка для сброса пароля и установки нового.

*Требование REQ\_UM\_004*

Подсистема должна обеспечивать хранение и управление информацией о пользователях, включая:

* Персональные данные (имя, возраст, пол, рост, вес);
* Контактную информацию (адрес электронной почты, номер телефона);
* Предпочтения в еде и аллергии;
* Цель (набор массы, сброс веса, поддержание веса);
* Уровень физической активности;
* Историю расчетов калорийности и подобранных меню.

*Требование REQ\_UM\_005*

Подсистема должна предоставлять возможность редактирования информации о пользователе через личный кабинет. Пользователь должен иметь возможность:

* Изменять персональные данные;
* Изменять контактную информацию;
* Редактировать список предпочтений в еде и аллергий;
* Изменять цель и уровень физической активности;
* Удалять записи из истории расчетов и подобранных меню.

*Требование REQ\_UM\_006*

Подсистема должна обеспечивать конфиденциальность и безопасность хранения персональных данных пользователей. Доступ к информации о пользователе должен быть ограничен и предоставляться только самому пользователю и администраторам системы в случае необходимости.

*Требование REQ\_UM\_007*

Подсистема должна предоставлять возможность удаления учетной записи пользователя по его запросу. При удалении учетной записи должна быть удалена вся связанная с пользователем информация, включая историю расчетов и подобранных меню.

*Требование REQ\_UM\_008*

Подсистема должна обеспечивать возможность экспорта данных пользователя в формате PDF или CSV. Пользователь должен иметь возможность экспортировать:

* Персональные данные и контактную информацию;
* Список предпочтений в еде и аллергий;
* Историю расчетов калорийности и подобранных меню за выбранный период.

**Требования к подсистеме «Подбор меню»**

*Требование REQ\_MM\_001*

Подсистема должна обеспечивать возможность подбора меню на основе указанных пользователем параметров:

* Желаемая калорийность меню;
* Бюджет на меню;
* Время приема пищи (завтрак, обед, ужин, перекус);
* Наличие определенных продуктов или ингредиентов;
* Предпочтения в еде и аллергии пользователя;
* Цель пользователя (набор массы, сброс веса, поддержание веса).

*Требование REQ\_MM\_002*

Подсистема должна иметь базу данных рецептов блюд с информацией о:

* Названии блюда;
* Ингредиентах и их количестве;
* Калорийности и содержании питательных веществ (белки, жиры, углеводы);
* Времени приготовления;
* Уровне сложности приготовления;
* Стоимости ингредиентов.

*Требование REQ\_MM\_003*

Подсистема должна предоставлять возможность поиска рецептов блюд в базе данных по названию, ингредиентам или другим параметрам. Результаты поиска должны отображаться в виде списка с возможностью просмотра подробной информации о каждом блюде.

*Требование REQ\_MM\_004*

Подсистема должна обеспечивать возможность фильтрации рецептов блюд по различным критериям:

* Время приема пищи (завтрак, обед, ужин, перекус);
* Наличие определенных ингредиентов;
* Диапазон калорийности;
* Диапазон времени приготовления;
* Уровень сложности приготовления;
* Стоимость ингредиентов.

*Требование REQ\_MM\_005*

Подсистема должна производить подбор меню на основе указанных пользователем параметров и предпочтений. Алгоритм подбора должен учитывать:

* Желаемую калорийность меню;
* Бюджет на меню;
* Наличие определенных продуктов или ингредиентов;
* Предпочтения в еде и аллергии пользователя;
* Цель пользователя (набор массы, сброс веса, поддержание веса);
* Сбалансированность питания (соотношение белков, жиров и углеводов).

*Требование REQ\_MM\_006*

Подсистема должна предоставлять возможность просмотра подробной информации о каждом блюде в подобранном меню, включая:

* Название блюда;
* Ингредиенты и их количество;
* Калорийность и содержание питательных веществ;
* Рецепт приготовления;
* Время приготовления;
* Стоимость ингредиентов.

*Требование REQ\_MM\_007*

Подсистема должна обеспечивать возможность замены блюд в подобранном меню на альтернативные варианты. Пользователь должен иметь возможность:

* Просматривать список альтернативных блюд для каждого блюда в меню;
* Выбирать альтернативное блюдо из списка и заменять им исходное блюдо в меню;
* Просматривать обновленную информацию о калорийности, содержании питательных веществ и стоимости меню после замены блюд.

**Требования к подсистеме «интеграции и API»**

*Требование REQ\_IA\_001*

Подсистема должна предоставлять возможность интеграции с внешними сервисами и приложениями через API (Application Programming Interface). API должен быть реализован в соответствии с архитектурным стилем REST (Representational State Transfer).

*Требование REQ\_IA\_002*  
API должен предоставлять следующие основные функции:

* Аутентификация и авторизация пользователей;
* Получение информации о продуктах и блюдах из базы данных;
* Расчет калорийности и питательных веществ для указанного количества продукта или порций блюда;
* Подбор меню на основе указанных параметров и предпочтений пользователя;
* Получение информации о пользователе и его предпочтениях.
* Сохранение и получение истории расчетов калорийности и подобранных меню

*Требование REQ\_IA\_003*

API должен использовать стандартные методы HTTP (GET, POST, PUT, DELETE) для выполнения операций над ресурсами. Каждый ресурс должен иметь уникальный идентификатор (URI) и поддерживать соответствующие методы HTTP для выполнения CRUD-операций (Create, Read, Update, Delete).

*Требование REQ\_IA\_004*

Формат обмена данными через API должен быть JSON (JavaScript Object Notation). Все запросы и ответы должны содержать данные в формате JSON, если не указано иное.

*Требование REQ\_IA\_005*

API должен обеспечивать безопасность передачи данных. Все запросы должны выполняться через протокол HTTPS (HTTP Secure) для шифрования передаваемых данных. Аутентификация пользователей должна осуществляться с использованием токенов доступа (например, JWT - JSON Web Tokens).

*Требование REQ\_IA\_006*

API должен иметь подробную документацию, описывающую доступные ресурсы, методы, параметры запросов и форматы ответов. Документация должна быть доступна в удобном для разработчиков формате (например, OpenAPI/Swagger).

*Требование REQ\_IA\_007*

Подсистема должна обеспечивать возможность версионирования API. Различные версии API должны иметь отдельные URL-адреса и поддерживать обратную совместимость для существующих клиентов.

*Требование REQ\_IA\_008*

Подсистема должна обеспечивать обработку ошибок и возвращать соответствующие коды состояния HTTP в случае возникновения ошибок. Сообщения об ошибках должны быть информативными и содержать подробную информацию о причине ошибки.

*Требование REQ\_IA\_009*

Подсистема должна обеспечивать возможность ограничения доступа к API на основе ролей и прав пользователей. Различные роли пользователей (например, обычный пользователь, премиум-пользователь, администратор) должны иметь различные уровни доступа к функциям API.

*Требование REQ\_IA\_010*

Подсистема должна вести журналирование (логирование) всех запросов к API и ответов от него. Журналы должны содержать информацию о времени запроса, IP-адресе клиента, методе HTTP, URL-адресе, параметрах запроса и коде состояния HTTP ответа. Журналы должны быть доступны для анализа и мониторинга работ

# 6. Разработка архитектуры проекта

Создание плана архитектуры проекта является одним из важных этапов при разработке ПО, такое представления проекта помогает чётко разделить проект на отдельные задачи, и помогает на ранних этапах фиксировать структурные проблемы и решать их.

Для лучшего представления предметное области строятся модели отношения сущностей, такая модель показывает связи между сущностями предметной области, а также может демонстрировать предварительный вид базы данных ПО.(рис.9)

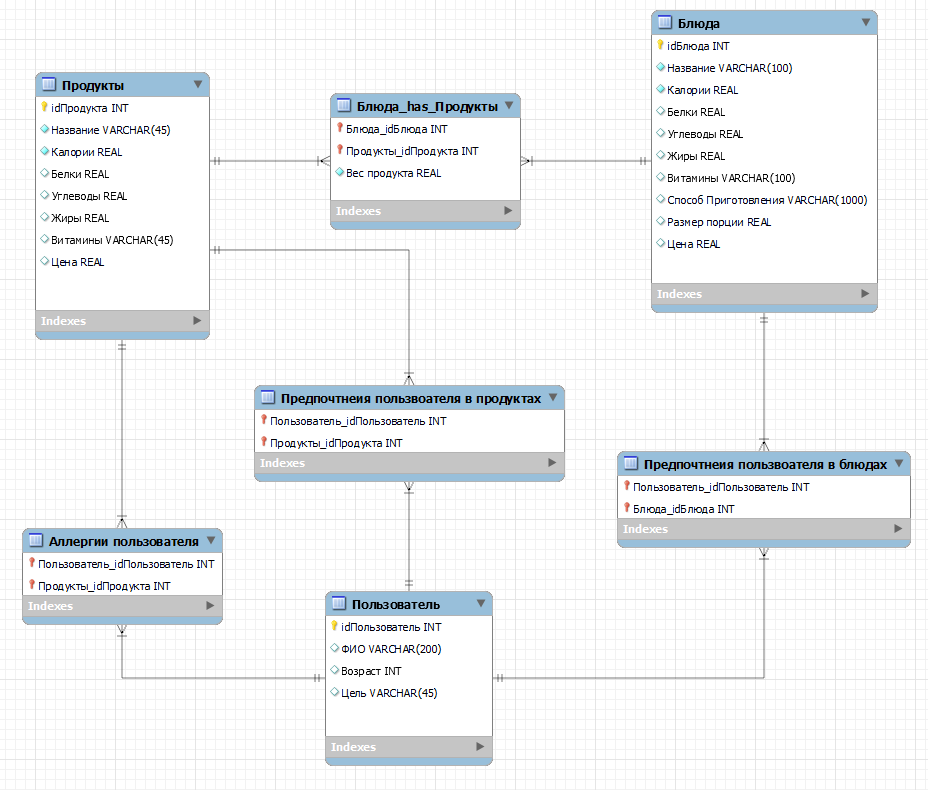


Рисунок 9 – модель отношения сущностей

Для представления архитектуры проекта является разумным составление контекстных диаграмм и диаграмм последовательностей, которые будут описывать потоки данных между процессами, всего ПО или конкретного модуля. На рисунках 10 и 11 представлены контекстная диаграмм и диаграмма последовательностей.  
**Контекстная диаграмма для подсистемы «Подсчёт калорий»**

Изображение выглядит как текст, диаграмма, линия, Параллельный

Автоматически созданное описание

Рисунок 10 – контекстная диаграмма

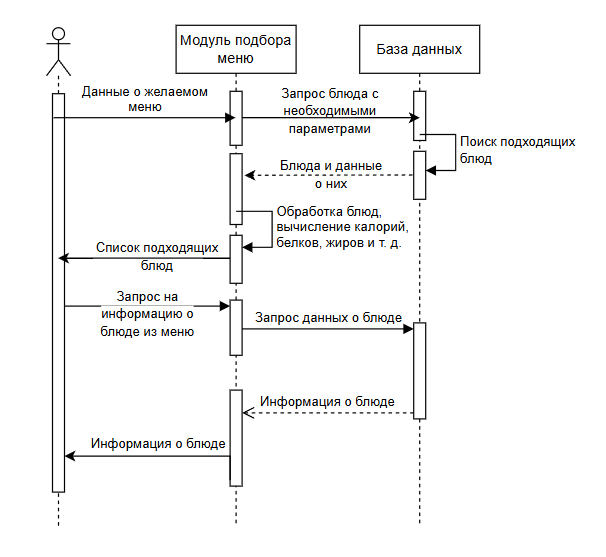
**Диаграмма последовательности для подсистемы «Подбор меню»  
**

Рисунок 11 – диаграмма последовательности

# 7. Разработка измерений проекта

**Оценка эффективности процесса разработки**

* **Productivity** = LOC / Рабочее время, затраченное на проект.
  1. **Стратегическая цель метрики:** Повысить производительность труда.
  2. **Изучаемый объект метрики:** проект.
  3. **Измеряемый атрибут:** производительность труда.
  4. **Единица измерения:** LOC / человеко-час.
* **Problem Resolution Rate (PRR)** = Количество дней на обработку задачи.
  1. **Стратегическая цель метрики:** Сократить сроки выполнения задач.
  2. **Изучаемый объект метрики:** задача.
  3. **Измеряемый атрибут:** время обработки.
  4. **Единица измерения:** день.

**Оценка качества программного продукта**

* **Product Fault Density (PFD)**
* **PFD-UI:** Число ошибок в интерфейсе / Количество страниц интерфейса.
* **PFD-CC:** Число ошибок в подсчете калорий / Количество обработанных запросов.
* **PFD-UM:** Число ошибок в управлении пользователями / Количество пользователей.
  1. **PFD-MM:** Число ошибок в подборе меню / Количество составленных меню.
  2. **PFD-IA:** Число ошибок в интеграции / Количество API-вызовов.
  3. **Стратегическая цель метрики:** Повысить качество разрабатываемого ПО.
  4. **Изучаемый объект метрики:** продукт.
  5. **Измеряемый атрибут:** плотность ошибок.
  6. **Единица измерения:** ошибка / единица размера.

# 8. Разработка перечня задач проекта

**Задачи для подсистемы «Пользовательский интерфейс»**

* **Задача TASK\_UI\_001:**
  1. **Название:** Проектирование интерфейса.
  2. **Описание:** На основе требований разработать интерфейс. Необходимый функционал: шапка сайта с кнопками навигации, страница с калькулятором калорий, адаптивный дизайн.

**Задачи для подсистемы «Подсчет калорий»**

* **Задача TASK\_CC\_001:**
  1. **Название:** Реализация калькулятора калорий.
  2. **Описание:** Разработать алгоритм подсчета калорий на основе введенных данных. Учитывать параметры продуктов (БЖУ, калорийность).

**Задачи для подсистемы «Управление пользователями»**

* **Задача TASK\_UM\_001:**
  1. **Название:** Разработка личного кабинета.
  2. **Описание:** Реализовать функционал регистрации, входа и управления профилем пользователя.

**Задачи для подсистемы «Подбор меню»**

* **Задача TASK\_MM\_001:**
  1. **Название:** Реализация алгоритма подбора меню.
  2. **Описание:** Разработать алгоритм на основе БЖУ, бюджета, предпочтений и ограничений пользователя.

**Задачи для подсистемы «Интеграция и API»**

1. **Задача TASK\_IA\_001:**
   1. **Название:** Реализация API для интеграции.
   2. **Описание:** Разработать API для взаимодействия с внешними сервисами (например, базы данных продуктов).

# 9. Разработка рекомендаций по кодированию

Для создания качественного кода на любом языке программирования, обладающего таким свойствами, как удобочитаемость (readability) и понятность (understandability), необходимо следовать хорошо определённым стандартам и руководствам. Особенно это актуально при коллективной разработке программ. Любой стандарт кодирования призван определить набор правил, которые способствуют разработке более единообразного кода и минимизации числа общераспространенных ошибок в нем, не ущемляя при этом права разработчика на творчество.

Программная система будет реализована на языке программирования Python .  
PEP 8 — документ, описывающий соглашение о том, как писать код на языке Python. PEP 8 создан на основе рекомендаций создателя языка Гвидо ван Россума. Ключевая идея Гвидо такова: код читается намного больше раз, чем пишется.

Собственно, рекомендации о стиле написания кода направлены на то, чтобы улучшить читаемость кода и сделать его согласованным между большим числом проектов. В идеале, если весь код будет написан в едином стиле, то любой сможет легко его прочесть.

**Основные правила PEP 8:**

**Форматирование**

Использовать четыре пробела для отступов. Не делать отступов в два пробела. Wing Ide помогает правильно расставлять пробелы. По умолчанию в Wing Ide клавиша Tab ставит четыре пробела.

Писать import каждого модуля в отдельной строке

Располагать все import'ы в верхней части кода перед любыми глобальными объявлениями.

Отделять блок import'ов от кода пустой строкой.

Не использовать конструкцию from … import \*}

Скобки не отделяются пробелами с внутренней стороны. Между функцией и ее аргументами пробел не ставится.

Пример:

spam(ham[1], {eggs: 2}) # Правильно

spam( ham[ 1 ], { eggs: 2 } ) # Неверно

Перед запятой, двоеточием пробел не ставится, после -— ставится.

Пример:

if x == 4:

print(x, y)

x, y = y, x # Правильно

if x == 4 :

print(x , y)

x , y = y , x # Неверно

Всегда необходимо окружать следующие бинарные операторы ровно одним символом пробела с каждой стороны:

присваивания (=, +=, -= и т. д.),

сравнения (==, <, >, !=, <>, <=, >=, in, not in, is, is not),

логические (and, or, not),

арифметические (+, -, \*, /, //, \%, \*\*).

Не располагайте несколько инструкций в одной строке – разносить по разным строкам.

Пример:

x = 3 # Правильно

func(10)

x = 3; func(10) # Неверно

Не располагать блок из нескольких инструкций на той же строке сразу после двоеточия (после if, while и т. д.)

Комментарии

Комментарии, противоречащие коду, хуже, чем их отсутствие.

Необходимо располагать однострочные комментарии после кода в той же строке и отделяйте их от кода не менее чем двумя пробелами. Комментарии должны начинаться с # и одного пробела.

**Имена**

Не использовать символы `l', `O', и `I' как имена переменных. В некоторых шрифтах они могут быть очень похожи на цифры.

Имена переменных и функций должны содержать только маленькие буквы. Слова разделяются символами подчёркивания.

Примеры:

name, name\_with\_several\_words\_in\_it

Имена констант должны содержать только заглавные буквы. Слова разделяются символами подчёркивания.

Примеры:

NAME, NAME\_WITH\_SEVERAL\_WORDS\_IN\_IT

Необходимо давать переменным говорящие английские имена - не использовать транслит.

num\_letters = int(input()) # Правильно

kolvo\_bukv = int(input()) # Неверно

**Функции**

Необходимо разделять определения функций двумя пустыми строками, а также отделять определения функций от основного кода двумя пустыми строками до и после.

Тело функции разделять на логические части одной пустой строкой.

# 10. Разработка плана тестирования проекта

**Тесты для подсистемы «Пользовательский интерфейс»**

* **Тест TEST\_UI\_001:**
  1. **Тестируемые требования:** REQ\_UI\_001.
  2. **Описание теста:** Запуск сайта, проверка наличия шапки и кнопок навигации.
  3. **Ожидаемый результат:** Шапка сайта содержит логотип, кнопки навигации, а также кнопку «Вход».
* **Тест TEST\_UI\_003:**
  1. **Тестируемые требования:** REQ\_UI\_003.
  2. **Описание теста:** Проверка адаптивности интерфейса на мобильных устройствах.
  3. **Ожидаемый результат:** Интерфейс должен корректно адаптироваться к различным размерам экранов мобильных устройств.
  4. **Видимый результат**: Полностью совпадает с ожидаемым.

**Тесты для подсистемы «Подсчет калорий»**

* **Тест TEST\_CC\_001:**
  1. **Тестируемые требования:** REQ\_CC\_001.
  2. **Описание теста:** Ввод данных в калькулятор, расчет калорий.
  3. **Ожидаемый результат:** Правильный расчет калорий и отображение результата.
* **Тест TEST\_CC\_002:**
  1. **Тестируемые требования:** REQ\_CC\_002.
  2. **Описание теста:** Проверка корректности расчета калорий при вводе разных значений (вес, активность, возраст).
  3. **Ожидаемый результат:** Программа должна корректно вычислять необходимое количество калорий в зависимости от введенных данных.
  4. **Видимый результат**: Полностью совпадает с ожидаемым.
* **Тест TEST\_CC\_003:**
  1. **Тестируемые требования:** REQ\_CC\_003.
  2. **Описание теста:** Проверка калькулятора на ошибки при вводе неверных или пустых данных.
  3. **Ожидаемый результат:** Программа должна сообщить об ошибке или запросить корректный ввод данных.
  4. **Видимый результат**: Полностью совпадает с ожидаемым

**Тесты для подсистемы «Управление пользователями»**

* **Тест TEST\_UM\_001:**
  1. **Тестируемые требования:** REQ\_UM\_001.
  2. **Описание теста:** Регистрация и вход в личный кабинет.
  3. **Ожидаемый результат:** Успешная регистрация и доступ к профилю пользователя.
* **Тест TEST\_UM\_002:**
  1. **Тестируемые требования:** REQ\_UM\_002.
  2. **Описание теста:** Проверка восстановления пароля через email.
  3. **Ожидаемый результат:** Пользователь получает ссылку для восстановления пароля на указанный email.
  4. **Видимый результат**: Полностью совпадает с ожидаемым.
* **Тест TEST\_UM\_003:**
  1. **Тестируемые требования:** REQ\_UM\_003.
  2. **Описание теста:** Проверка работы функционала выхода из аккаунта.
  3. **Ожидаемый результат:** Пользователь успешно выходит из системы, и сессия завершается.
  4. **Видимый результат**: Полностью совпадает с ожидаемым.

**Тесты для подсистемы «Подбор меню»**

* **Тест TEST\_MM\_001:**
  1. **Тестируемые требования:** REQ\_MM\_001.
  2. **Описание теста:** Выбор параметров для подбора меню.
  3. **Ожидаемый результат:** Генерация меню на основе предпочтений пользователя.
* **Тест TEST\_MM\_002:**
  1. **Тестируемые требования:** REQ\_MM\_002.
  2. **Описание теста:** Проверка корректности работы фильтра подбора меню по категориям (например, без глютена, веганское и т.д.).
  3. **Ожидаемый результат:** Меню отображает только блюда, соответствующие выбранным фильтрам.
  4. **Видимый результат**: Полностью совпадает с ожидаемым.
* **Тест TEST\_MM\_003:**
  1. **Тестируемые требования:** REQ\_MM\_003.
  2. **Описание теста:** Проверка алгоритма подбора меню с учетом предпочтений пользователя.
  3. **Ожидаемый результат:** Меню генерируется в зависимости от предпочтений пользователя (калорийность, тип пищи и т.д.).
  4. **Видимый результат**: Полностью совпадает с ожидаемым.

**Тесты для подсистемы «Интеграция и API»**

* **Тест TEST\_IA\_001:**
  1. **Тестируемые требования:** REQ\_IA\_001.
  2. **Описание теста:** Отправка запроса к API.
  3. **Ожидаемый результат:** Корректный ответ API.
* **Тест TEST\_IA\_002:**
  1. **Тестируемые требования:** REQ\_IA\_002.
  2. **Описание теста:** Проверка корректности обработки ответа от внешнего API при отправке данных (например, для рецептов).
  3. **Ожидаемый результат:** Ответ от API должен быть корректным и содержать ожидаемые данные.
  4. **Видимый результат**: Полностью совпадает с ожидаемым.
* **Тест TEST\_IA\_003:**
  1. **Тестируемые требования:** REQ\_IA\_003.
  2. **Описание теста** Проверка корректности обработки ошибки при недоступности API.
  3. **Ожидаемый результат:** : При недоступности API система должна корректно обрабатывать ошибку и информировать пользователя.
  4. **Видимый результат**: Полностью совпадает с ожидаемым.

# Заключение

В рамках курсовой работы был проведён анализ и подготовка к разработке веб-сервиса «FitFuel». Для достижения поставленной цели были решены следующие задачи:

* разработан план проекта;
* создан регламент проведения инспекции;
* построена модель состояний задач;
* подготовлена презентация проекта;
* сформулированы требования к проекту;
* разработана архитектура проекта;
* определены измерения проекта;
* составлен перечень задач
* даны рекомендации по кодированию;
* составлен план тестирования проекта.

Таким образом, несмотря на то, что сам веб-сервис не был реализован в рамках данной работы, была проведена тщательная подготовка и составлен отчёт, который может служить основой для дальнейшей разработки проекта.

# Список литературы

1. Гриняк В.М. Лекции по дисциплине «Технологии коллективной промышленной разработки информационных систем», [Электронный ресурс]. – Режим доступа: [https://bb.dvfu.ru/webapps/blackboard/content/listContent.jsp?course\_id=\_5025\_1&content\_id=\_172368\_1&mode=reset](https://bb.dvfu.ru/webapps/blackboard/content/listContent.jsp?course_id=_5025_1&content_id=_172368_1&mode=reset%20) (дата обращения: 03.12.2024)