  
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«Дальневосточный федеральный университет»**

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ  
Департамент программной инженерии и искусственного интеллекта**

**РАЗРАБОТКА СИСТЕМЫ ДЛЯ РАЦИОНАЛЬНОГО РАСПОРЯЖЕНИЯ ФИНАНСАМИ**

КУРСОВОЙ ПРОЕКТ  
по дисциплине «Технологии коллективной промышленной разработки информационных систем» по образовательной программе подготовки бакалавров по направлению 09.03.04 «Программная инженерия»

Выполнили:  
студенты гр. Б9120-09.03.04прогин  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Пестряк Н.Е.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сидоров Н.В.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Гилко М.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Бройко О.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сампилов М.Б.   
Руководитель:  
Ассистент департамента ПИиИИ  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю.С.

г. Владивосток  
2024

**Оглавление**

[Введение 3](#_Toc155960486)

[1. План проекта 4](#_Toc155960487)

[2. Регламент проведения инспекции 5](#_Toc155960488)

[3. Модель состояний задач 8](#_Toc155960489)

[4 Презентация проекта 10](#_Toc155960490)

[5. Требований к проекту 13](#_Toc155960491)

[6. Разработка архитектуры проекта 15](#_Toc155960492)

[7. Измерения проекта 18](#_Toc155960493)

[8. Перечень задач проекта 20](#_Toc155960494)

[9. Правила по кодированию 22](#_Toc155960495)

[10. Разработка плана тестирования проекта 26](#_Toc155960496)

[Заключение 29](#_Toc155960497)

[Список литературы 30](#_Toc155960498)

# Введение

Промышленная разработка информационных систем охватывает разнообразные этапы, начиная от разработки проектного плана и заканчивая тестированием, требуя участия специалистов различных профилей. Для обеспечения эффективной коммуникации и четкого распределения обязанностей важно использовать стандартизированные методы.

В рамках данной курсовой работы рассматривается задача коллективной разработки программного средства под названием "Трекер расходов". Главная цель проекта заключается не только в создании самого инструмента, но и в разработке соответствующей технической документации. Таким образом, основная задача курсовой работы заключается в успешной реализации программного средства с использованием принципов коллективной промышленной разработки, обеспечивая тем самым эффективную работу команды и соответствие конечного продукта заявленным требованиям.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать план проекта.
2. Разработать регламент проведения инспекции.
3. Разработать модель состояний задач.
4. Разработать презентацию проекта.
5. Разработать требования к проекту.
6. Разработать архитектуру проекта.
7. Разработать измерения проекта.
8. Разработать перечень задач проекта.
9. Разработать рекомендации по кодированию.
10. Разработать план тестирования проекта.
11. Протестировать проект.

# 1. План 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 – Пестряк Н.Е.;
* Coder 1 – Сидоров Н.В.;
* QA engineer – Сампилов М.Б.
* Architect – Бройко О.С.;
* Technical Writer – Гилко М.С..

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

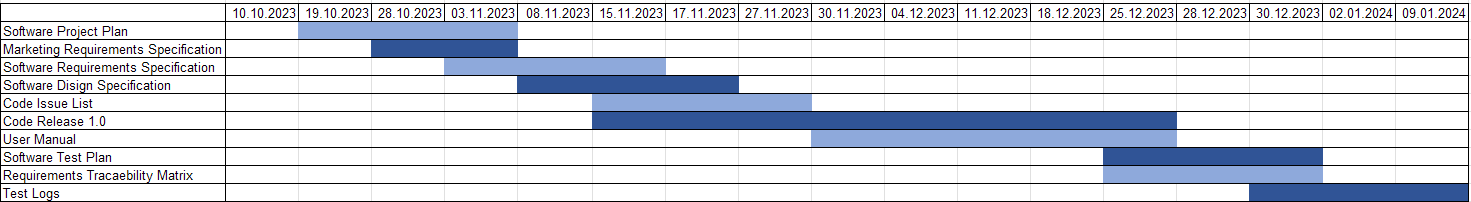


Рисунок 1 – План проекта

# 2. Регламент проведения инспекции

Верификация рабочих продуктов является неотъемлемой частью процесса обеспечения их качества. Современные технологии программирования предлагают специальные стандарты, подходы и механизмы для проведения верификации рабочих продуктов, включая инспекции.

* Инспекция — это мероприятие, направленное на обеспечение качества рабочих продуктов проектов по разработке программного обеспечения и других деятельностей, которое проводится разработчиками с возможным участием представителей заказчика.
* Концептуально инспекция имеет следующие цели:
* Выявление ошибок в функциях, логике, содержании или реализации рабочих продуктов на ранних этапах разработки и предотвращение их наследования;
* Эффективное донесение концепции или реализации продукта до всех заинтересованных сторон (через их участие);
* оптимизация, оценка или улучшение рабочего продукта.

## Критерии отнесения к формальной/неформальной инспекции для различных типов рабочих продуктов

|  |  |  |
| --- | --- | --- |
|  | **Формальная инспекция в случае CRUD-операций над** | **Неформальная инспекция в случае CRUD-операций над** |
| **Требования** | >1 требованием | 1 требованием |
| **Документы дизайна** | >1 объектом дизайна внешнего вида готового продукта | 1 объектом дизайна внешнего вида готового продукта (изображение) |
| **Код** | >7 строчками | <8 строчками |
| **Тесты** | >3 тестами | < 4тестами |

## Перечень ролей участников инспекции и их обязанности

Автор (Author) – сотрудник, разработавший инспектируемый рабочий продукт, либо сделавший инспектируемые изменения в существующем рабочем продукте.

Инспектор (Inspector) – сотрудник, ответственный за эффективную проверку инспектируемого рабочего продукта.

## Этапы инспекции

## Планирование инспекции.

## При планировании инспекции коллективно выбирается дата, время, формат (очный или заочный) и платформа (при заочной инспекции) проведения инспекции.

## Подготовка к инспекции.

## Инспектор самостоятельно изучает предоставленный для инспекции рабочий продукт, используя накопленный опыт и стандарты.

## Собрание по инспекции.

## На собрании происходит обсуждение замечаний и рекомендаций инспектора по рабочему продукту. На собрании по инспекции обязательно присутствует инспектор и автор рабочего продукта, требующего инспекции. Присутствие остальных участников команды разработки по желанию.

## Завершение инспекции.

## Если рабочий продукт требует доработки, то автор фиксирует все замечания и рекомендации инспектора, разрабатывает план предстоящих работ и далее согласовывает его с инспектором. Если рабочий продукт не требует доработки, то инспектор подтверждает слияние рабочей ветки в «master» ветку.

## Перечень статусов и степени важности замечаний

1. Ошибка – проблема, которая найдена на той же фазе, на которой внесена. Допустимые значения степени серьёзности ошибки:

* Критическая (Critical)
* Средняя (Moderate)
* Мелкая, незначительная (Minor)
* Другие (Other)

1. Комментарий – это наблюдение, предложение, рекомендация или улучшение, предложенное для будущего выпуска рабочего продукта или вопрос, требующий разъяснения.

## Метрики, характеризующие эффективность инспекций

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

В качестве метрики, характеризующей эффективность инспекции, была выбрана Inspection Preparation Rate (IPR):

IPR = (Количество инспекторов \* Размер продукта) / Общее время подготовки

Изучаемый объект метрики – подготовка к инспекции, измеряемый атрибут – производительность подготовки к инспекции.

Единица измерения – <страница, требование, LOC, тест> / час

# 3. Модель 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:

## New – новая подзадача.

## Analysis – в процессе анализа. В это состояние подзадачу переводит сотрудник после того, как начнёт её анализ.

## Forward – в данном случае имеет значение «переданный на дальнейшую разработку». В это состояние задача переводится CCB после анализа при назначении задачи на разработку конкретному сотруднику.

## Coding – кодирование. В это состояние задача переводится сотрудникомразработчиком, при начале работы по кодированию, связанному с задачей.

## Inspected – проинспектировано. В это состояние задача переводится сотрудником-разработчиком после завершения кодирования и инспектирования изменений рабочего продукта.

## Integrated – заинтегрировано. Переводится сотрудником, осуществляющим интеграцию изменений в основную ветку рабочего продукта после успешной интеграции этих изменений.

## Tested – протестировано. Переводится сотрудником, осуществляющим тестирование изменений в рабочий продукт (tester).

## Closed – закрыто. В это состояние задача переводится CCB по результатам отчёта о тестировании сделанных изменений.

## Правила создания новой задачи

Созданием новых задач могут заниматься все участники команды. Происходит это на начальном этапе, когда необходимо организовать структурированную работу над проектом и в процессе работы над проектом в случае присутствия в плане слишком объемных задач, требующих упрощения. Также, когда необходимо организовать баг-фиксинг или разработать новый функционал.

## Правила перехода задачи из состояния в состояние

Состояния задач всегда идут последовательно друг за другом, в некоторых случаях пункты могут опускаться или повторяться. Схема перехода из состояния в состояние показана на рисунке 2.

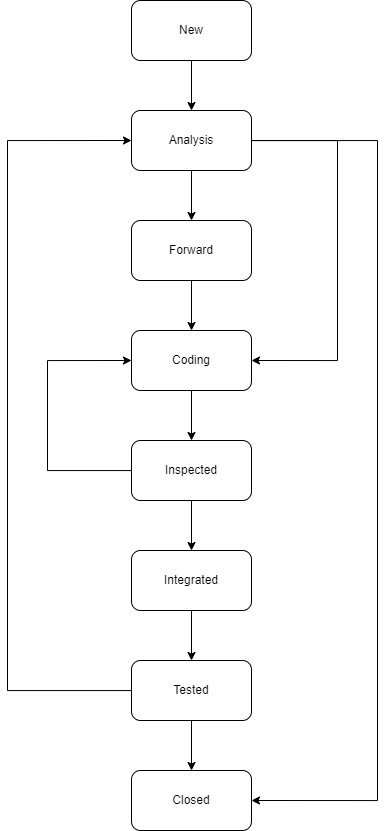


Рисунок 2 – Состояния задачи

# 4 Презентация проекта

На рисунке 3 представлена титульная страница презентации.

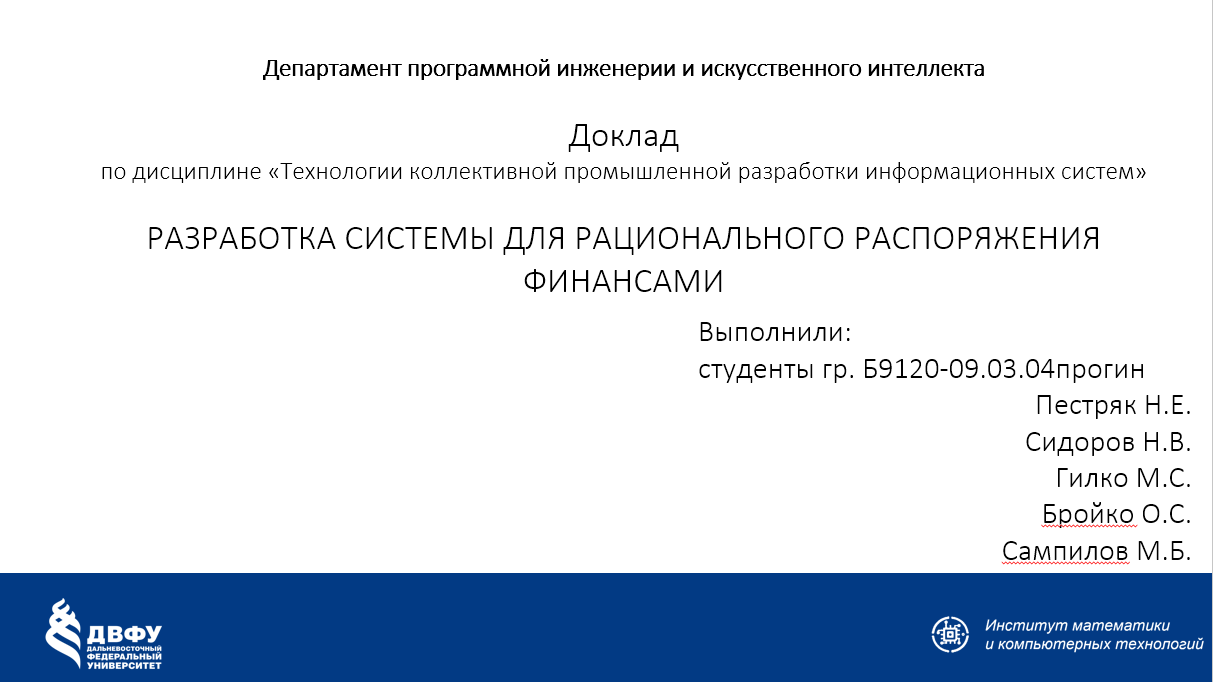


Рисунок 3 – Титульная страница

Проблемы, возникающие в предметной области разрабатываемого средства, представлены на рисунке 4.

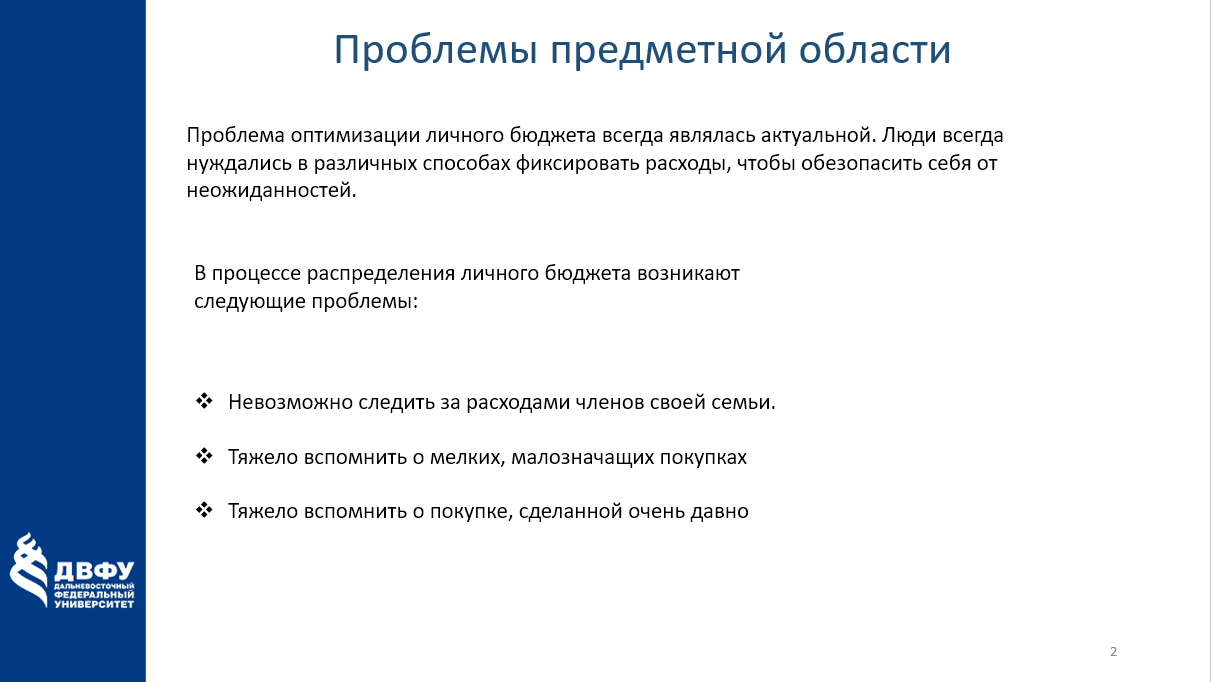


Рисунок 4 – Проблемы

На рисунке 5 демонстрируется, что можно получить и как, используя программное средство.

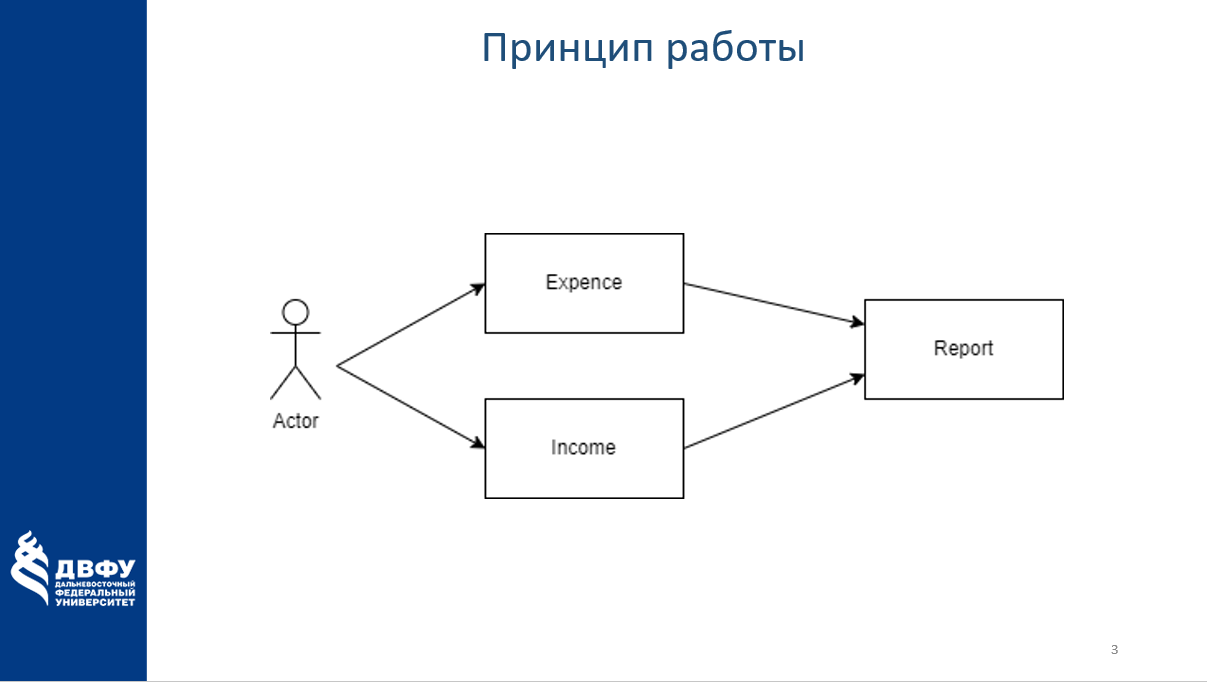


Рисунок 5 – Принцип работы

Внешний вид разрабатываемого средства представлен на рисунке 6.

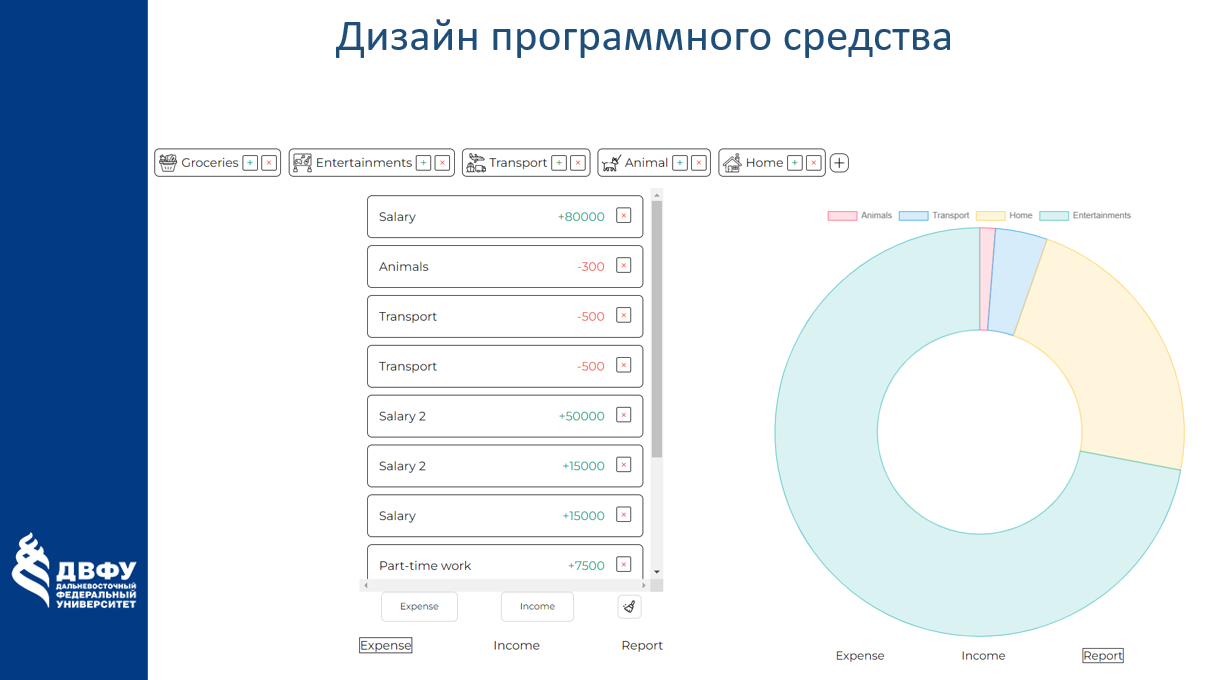


Рисунок 6 – Дизайн программного средства

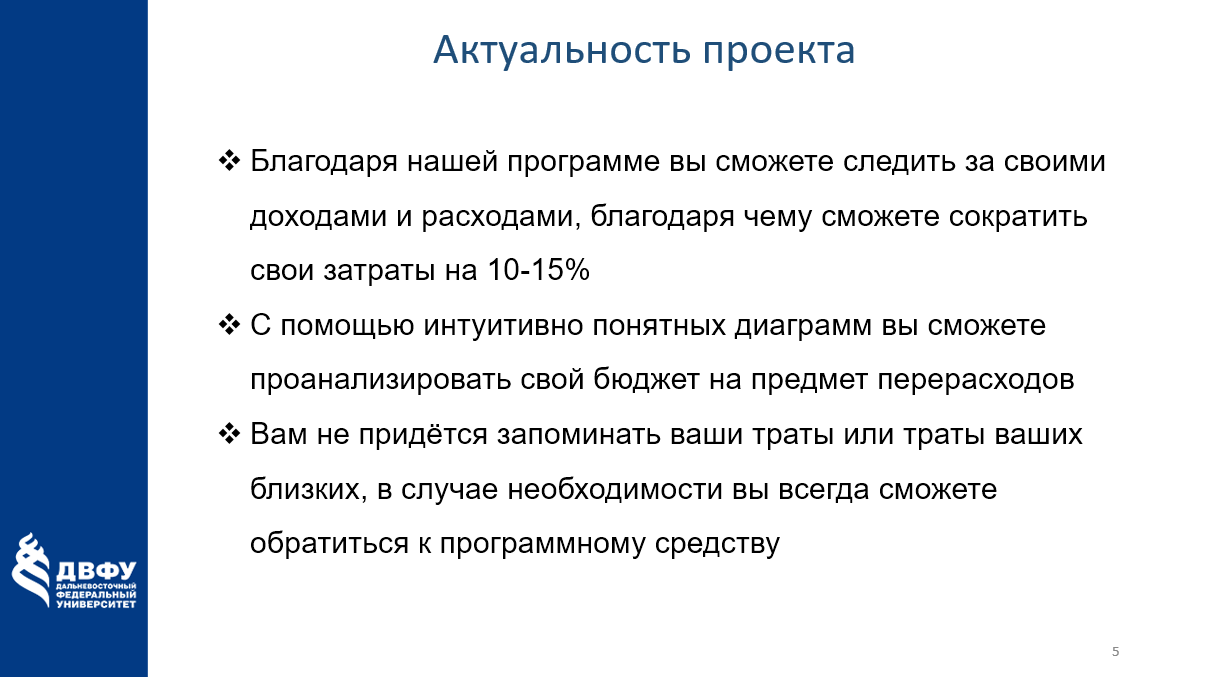
Актуальность разрабатываемого средства представлена на рисунке 7.

Рисунок 7 – Актуальность

# 5. Требований к проекту

Программный продукт «Трекер расходов» предназначен для отслеживания личных доходов и трат пользователя.

Программный продукт «Трекер расходов» состоит из следующих подсистем:

1. FE-1 – Список расходов пользователя
2. FE-2 – Список доходов пользователя
3. FE-3 – Отчёт доходов и расходов пользователя
4. FE-4 – Список категорий пользователя

## Матрица требований к подсистемам

|  |  |
| --- | --- |
| **Матрица требований** | **Идентификаторы** |
| Операции с расходами пользователя | FE-1 E |
| Просмотр | REQ-E-1 |
| Добавление | REQ-E-2 |
| Удаление | REQ-E-3 |
| Операции с доходами пользователя | FE-2 I |
| Просмотр | REQ-I-1 |
| Добавление | REQ-I-2 |
| Удаление | REQ-I-3 |
| Операции с отчётом о доходах и расходах пользователя | FE-3 R |
| Просмотр | REQ-R-1 |
| Операции со списком категорий | FE-4 CL |
| Просмотр | REQ-CL-1 |
| Добавление | REQ-CL-2 |
| Удаление | REQ-CL-3 |

## Расшифровка идентификаторов

* E – Expence (расходы пользователя)
* I – Income (доходы пользователя)
* R – Report (отчёт о доходах и расходах пользователя)
* CL – Categories List (список категорий)

## Требования к подсистеме «Список расходов пользователя»

## Требование REQ\_E\_1: Должен быть реализован просмотр списка всех трат пользователя

## Требование REQ\_E\_2: Должно быть реализовано удаление элементов списка всех трат пользователя

## Требование REQ\_E\_3: Должно быть реализовано добавление элементов списка всех трат пользователя

## Требования к подсистеме «Список доходов пользователя»

## Требование REQ\_I\_1: Должен быть реализован просмотр списка всех трат пользователя

## Требование REQ\_I\_2: Должно быть реализовано удаление элементов списка всех трат пользователя

## Требование REQ\_I\_3: Должно быть реализовано добавление элементов списка всех трат пользователя

## Требования к подсистеме «Список отчёт доходов и расходов пользователя»

## Требование REQ\_R\_1: Должен быть реализован просмотр отчёта доходов и расходов пользователя

**Требования к подсистеме «Список категорий пользователя»**

## Требование REQ-CL-1: Должен быть реализован просмотр списка всех категорий пользователя

## Требование REQ-CL-2: Должно быть реализовано добавление элементов списка категорий пользователя

## Требование REQ-CL-3: Должно быть реализовано удаление элементов списка категорий пользователя

# 6. Разработка архитектуры проекта

Архитектура программного обеспечения относится к фундаментальным структурам программной системы и дисциплине создания таких структур и систем. Каждая структура включает элементы программного обеспечения, отношения между ними, а также свойства как элементов, так и отношений.

Архитектура программной системы – это метафора, аналогичная архитектуре здания. Он функционирует как план для системы и проекта разработки, в котором излагаются задачи, которые должны быть выполнены командами разработчиков.

## Концептуальная модель

Концептуальная модель – это совокупность взаимосвязанных понятий, лежащих в основе исследовательского дизайна, системное описание исследуемой области. Концептуальная модель наглядно описывает структуру моделируемой предметной области и связи между ее элементами.

Концептуальная модель помогает решить, какие переменные наиболее важны, какие связи наиболее значимы и, следовательно, какую информацию необходимо собирать.

Схема концептуальной модели показана на рисунке 8.

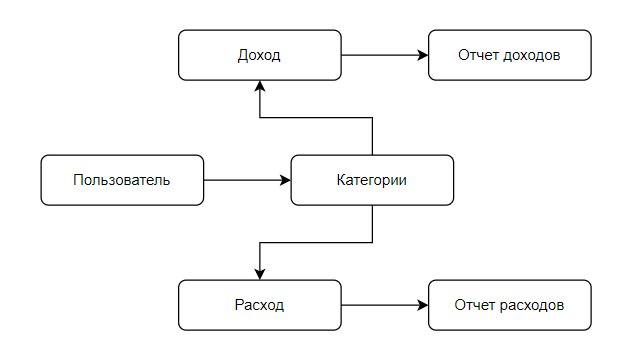


Рисунок 8 – Концептуальная модель

## Логическая модель

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области.

Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

Схема данной модели представлена на рисунке 9.



Рисунок 9 – Логическая модель

## Контекстная диаграмма потоков данных

Одним из способов представления архитектуры проекта является диаграмма потоков данных.

Диаграммы потоков данных (Data Flow Diagrams – DFD) представляют собой иерархию функциональных процессов, связанных потоками данных.

Цель такого представления – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

На рисунке 10 представлена диаграмма потоков данных нулевого уровня между подсистемами разрабатываемой системы.

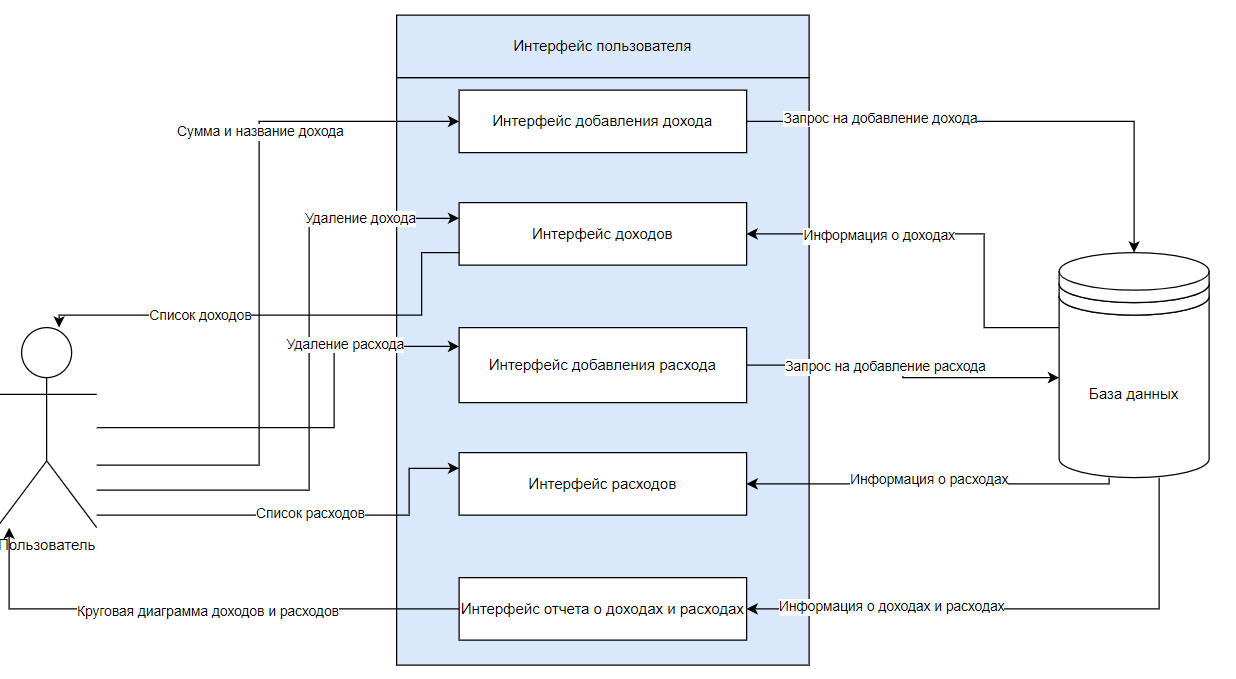


Рисунок 10 – DFD0

# 7. Измерения проекта

Контроль над производственным процессом и его результатами является ключевым видом деятельности на современном предприятии, производящем программное обеспечение на заказ. В силу специфики такого продукта, как программное обеспечение, для оценки эффективности процесса и качества конечного продукта применяются особые методы.

Комплекс мероприятий, направленных на количественную оценку эффективности работы, называется программой измерений.

## Метрика эффективности процесса производства

1. **Problem Resolution Rate (PRR)**

PRR = Количество дней на обработку задачи

Стратегическая цель метрики – сократить сроки выполнения проектов по разработке ПО.

Изучаемый объект метрики – задача, измеряемый атрибут – время обработки.

Единица измерения – день.

1. **Inspection Fault Density (IFD)**

IFD = (Количество найденных ошибок / Размер рабочего продукта)

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – инспекция, измеряемый атрибут – плотность найденных в ходе инспекции ошибок.

Количество найденных ошибок – ошибки, обнаруженные на этапе инспектирования.

Единица измерения – ошибка / <страница, требование**,** LOC, тест>.

## Метрика качества продукта

1. **In Process Faults (IPF)**

IPF = (Число обнаруженных ошибок до выпуска его релиза) / LOC

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – продукт, измеряемый атрибут – плотность неполадок.

Число обнаруженных ошибок – количество ошибок обнаруженных на этапе тестирования

Единица измерения – неполадка / LOC.

# 8. Перечень задач проекта

## ПОДСИСТЕМА FE-1

**TASK-1 (E-1, E-2)**

**Список расходов пользователя**

До 07.11.2023 реализовать просмотр и добавление записей расходов REQ-E-1 и REQ-E-2 (см. Software Requirement Specification).

**TASK-2 E-3**

**Удаление расходов пользователя**

До 10.11.2023 реализовать функцию удаления записи БД пользователем REQ-E-3 (см. Software Requirement Specification).

## ПОДСИСТЕМА FE-2

**TASK-1 (I-1, I-2)**

***Список доходов пользователя***

До 21.11.2023 реализовать просмотр и добавление записей доходов REQ-E-1 и REQ-E-2 (см. Software Requirement Specification).

**TASK-2 I-3**

**Удаление доходов пользователя**

До 25.11.2023 реализовать функцию удаления записи БД пользователем REQ-I-3 (см. Software Requirement Specification).

## ПОДСИСТЕМА FE-3

**TASK-1 R**

***Отчёт***

До 05.12.2023 реализовать предоставления доступа к просмотру отчёта по личным доходам и расходам пользователя REQ-R-1(см. Software Requirement Specification).

## ПОДСИСТЕМА FE-4

**TASK-1 CL**

***Просмотр и добавление списка категорий***

До 19.12.2023 реализовать просмотр и добавление, редактирования и удаление совокупности/части совокупности данных в БД, представляющей список категорий трат/доходов пользователя – REQ-CL-1, REQ-CL-2 и REQ-CL-3 (см. Software Requirement Specification).

# 9. Правила по кодированию

Для создания высококачественного кода на любом языке программирования, который отличается удобочитаемостью и понятностью, необходимо придерживаться установленных стандартов и руководств, особенно в условиях коллективной разработки программного обеспечения.

Каждый стандарт кодирования направлен на установление набора правил, способствующих созданию более структурированного кода и снижению количества распространенных ошибок, не затрагивая при этом творческую свободу разработчика.

## Рекомендации

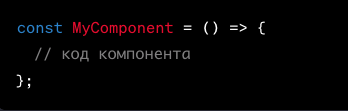
1. Комментарии должны представлять собой полноценные предложения.

Если комментарий – фраза или предложение, первое слово должно быть написано с большой буквы, если только это не имя переменной, которая начинается с маленькой буквы.

Ставьте два пробела после точки в конце предложения. Следует использовать английский язык для написания комментариев.

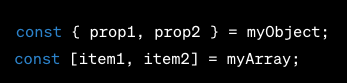
1. Использование стрелочных функций

Рекомендуется применять стрелочные функции при определении методов компонентов React, что обеспечивает сохранение контекста выполнения (this) и сделает код более лаконичным.



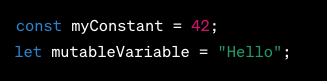
1. Деструктуризация объектов и массивов

Для повышения ясности кода используйте деструктуризацию при извлечении значений из объектов и массивов.



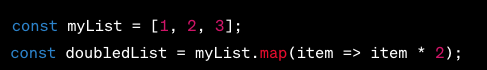
1. Использование ключевых слов let и const

Для объявления переменных, которые не будут переопределены, используйте const, а для переменных, которые могут меняться, используйте let.



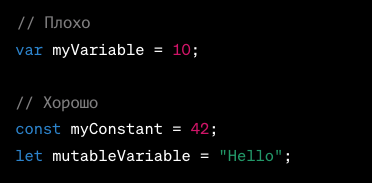
1. Использование метода map вместо циклов:

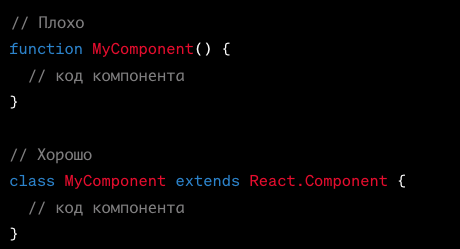
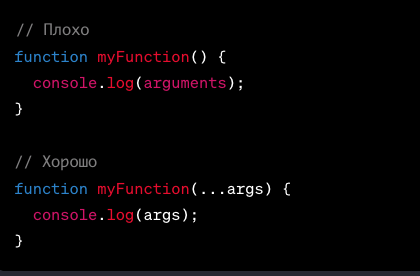
В React рекомендуется использовать метод map для обработки списков элементов вместо явных циклов, так как это способствует более декларативному и читаемому коду.



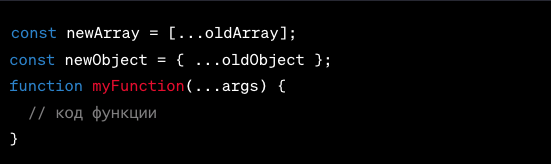
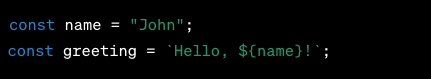
**Запрещается**

1. Запрещено использовать var



1. Запрещено использовать функции в качестве конструктора  
    
2. Запрещено использовать arguments  
    

## Требования

1. Используйте оператор spread/rest для работы с массивами и объектами  
    
2. Используйте шаблонные строки для более удобного и читаемого создания строковых выражений, особенно при вставке значениё переменных  
    
3. Используйте деструктуризацию в импортах модулей для явного указания используемых компонентов или функций  
    

# 10. Разработка плана тестирования проекта

## ПОДСИСТЕМА FE-1

* **TEST\_E\_1**

Тестируемые требования: REQ-E-1, REQ-E-2

Описание теста: Нажать на кнопку “Добавить”, выбрать нужную категорию и указав сумму, после чего вывести список расходов на экран

Ожидаемый результат: Расходы были добавлены и выведены на экран

* **TEST\_E\_2**

Тестируемые требования: REQ-E-2

Описание теста: Нажать на кнопку “Добавить”, выбрать нужную категорию и указав слишком большую сумму.

Ожидаемый результат: Поле для ввода суммы очистилось, добавление не произошло

* **TEST\_E\_2**

Тестируемые требования: REQ-E-2

Описание теста: Нажать на кнопку “Добавить”, выбрать нужную категорию и указав нечисловое значение.

Ожидаемый результат: Поле для ввода суммы очистилось, добавление не произошло

* **TEST\_E\_3**

Тестируемые требования: REQ-E-3

Описание теста: Выбрав из списка расходов, удалить нужный пункт

Ожидаемый результат: Из списка расходов удалился один выбранный пункт

## ПОДСИСТЕМА FE-2

* **TEST\_I\_1**

Тестируемые требования: REQ-E-1, REQ-E-2

Описание теста: Нажать на кнопку “Добавить”, выбрать нужную категорию и указав сумму, после чего вывести список расходов на экран

Ожидаемый результат: Доходы были добавлены и выведены на экран

* **TEST\_I\_2**

Тестируемые требования: REQ-E-2

Описание теста: Нажать на кнопку “Добавить”, выбрать нужную категорию и указав слишком большую сумму.

Ожидаемый результат: Поле для ввода суммы очистилось, добавление не произошло

* **TEST\_I\_2**

Тестируемые требования: REQ-I-2

Описание теста: Нажать на кнопку “Добавить”, выбрать нужную категорию и указав нечисловое значение.

Ожидаемый результат: Поле для ввода суммы очистилось, добавление не произошло

* **TEST\_I\_3**

Тестируемые требования: REQ-I-3

Описание теста: Выбрав из списка доходов, удалить нужный пункт

Ожидаемый результат: Из списка доходов удалился один выбранный пункт

## ПОДСИСТЕМА FE-3

* **TEST\_CL\_1**

Тестируемые требования: REQ-R-1

Описание теста: Нажать кнопку “Report’’

Ожидаемый результат: Переход на страницу формирования отчета ранее выбранных выкроек

## Матрица покрытия тестами требований

Матрица соответствия требований используется QA-инженерами для валидации покрытия требований по продукту тестами.

Цель «Traceability Matrix» состоит в том, чтобы выяснить:

* какие требования «покрыты» тестами, а какие нет;
* избыточность тестов (одно функциональное требование покрыто большим количеством тестов).

В соответствии с написанными требованиями и тестами на рисунке 12 представлена матрица покрытия тестами требований.

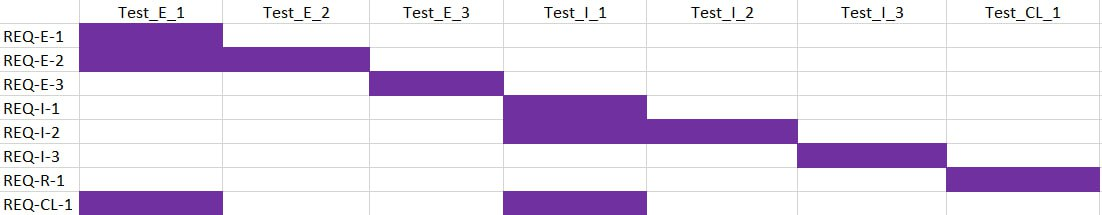


Рисунок 12 – Матрица покрытия тестами требований

# Заключение

В рамках курсовой работы было разработано программное средство «Программная система для автоматического создания выкроек» с использованием подходов коллективной промышленной разработки, для чего были решены следующие поставленные задачи:

1. Разработан план проекта.
2. Разработан регламент проведения инспекции.
3. Разработана модель состояний задач.
4. Разработана презентация проекта.
5. Разработаны требования к проекту.
6. Разработана архитектура проекта.
7. Разработаны измерения проекта.
8. Разработан перечень задач проекта.
9. Разработаны рекомендации по кодированию.
10. Разработан план тестирования проекта.

Таким образом, цель данной курсовой работы была достигнута.

По окончанию всех работ посчитаем метрики, описанные в 7 главе. Эффективность процесса разработки программного средства – 7 дней работы. Качество продукта выразилось в 15% плотности неполадок.

# Список литературы

1. Гриняк В.М. Лекции по дисциплине «Технологии коллективной промышленной разработки информационных систем». Электронный вариант.