  
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 ПРОГРАММНОЙ СИСТЕМЫ ПОДБОРА РАСТЕНИЙ ДЛЯ ДОМА**

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  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кучапина С. С.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Орлов Г. А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Петров Ф. А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Романова И. С.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чащин Т. А.  
Руководитель:  
ассистент департамента ПИиИИ  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю. С.

г. Владивосток  
2023

Оглавление

[Введение 3](#_Toc153576942)

[1. План проекта 4](#_Toc153576943)

[2. Регламент проведения инспекции 6](#_Toc153576944)

[3. Модель состояний задач 12](#_Toc153576945)

[4. Презентация проекта 16](#_Toc153576946)

[5. Требования к проекту 17](#_Toc153576947)

[6. Архитектура проекта 20](#_Toc153576948)

[7. Метрики проекта 21](#_Toc153576949)

[8. Перечень задач проекта 22](#_Toc153576950)

[9. Стандарты кода 23](#_Toc153576951)

[10. План тестирования проекта 24](#_Toc153576952)

[11. Тестирование проекта 25](#_Toc153576953)

[Заключение 26](#_Toc153576954)

[Список литературы 27](#_Toc153576955)

*Весь текст, который я как-либо изменил, я залил синим цветом*

*Вот таким красивым*

*Я не буду выделять синим текст в главах, начиная с 4й, т.к. их не было в прошлой версии отчёта*

*То, что я выделил некоторое заголовки синим – это я просто изменил автонумерацию на текст*

# Введение

Промышленная разработка информационных систем включает в себя множество этапов, начиная от разработки плана проекта, заканчивая его тестированием, для чего необходимо множество специалистов различных профилей, а также унифицированные методы коммуникации между ними, с помощью которых можно разделить обязанности членов команды по их специализации.

Исходя из описанного выше, необходимо использовать определенные технологии коллективной разработки для повышения эффективности работы в группе и соответствия конечного продукта заявленным требованиям.

В данной курсовой работе рассматривается задача коллективной разработки программного средства «Программная система подбора растений для дома» и составление технической документации к данному средству.

Таким образом, целью курсовой работы является разработка программного средства с использованием подходов коллективной промышленной разработки.

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.

На рисунке 1 представлены роли каждого из участников, этапы проекта, и задачи, реализуемые на каждом из этапов.

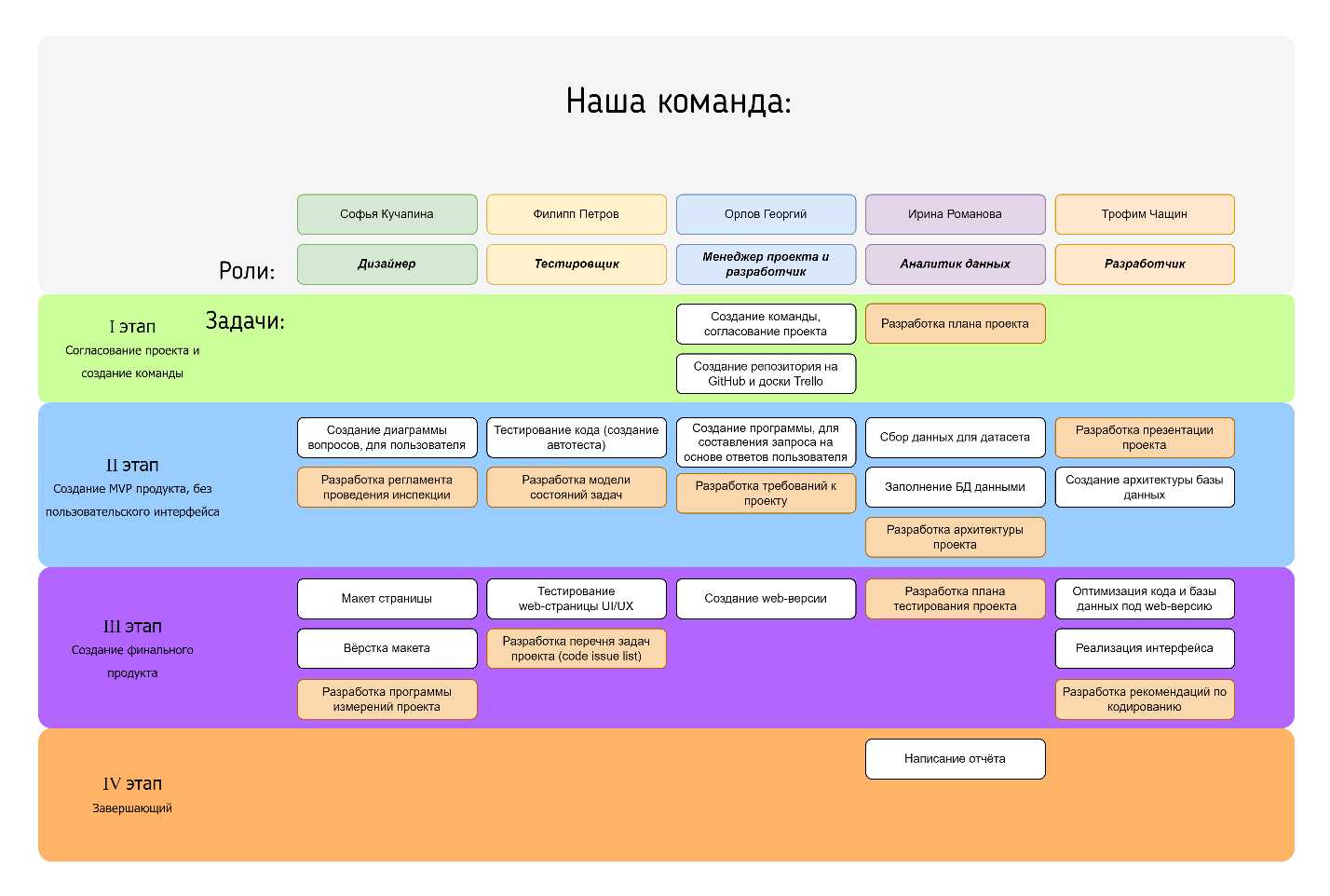


Рисунок 1 – Роли и задачи

Исполнителями данного проекта являются следующие лица:

* Team Leader – Орлов Георгий Александрович;
* Coder 1 – Кучапина Софья Сергеевна;
* Coder 2 – Орлов Георгий Александрович;
* Coder 3 – Петров Филипп Александрович;
* Coder 4 – Романова Ирина Сергеевна;
* Coder 5 – Чащин Трофим Александрович;
* Build Engineer – Орлов Георгий Александрович;
* Technical Writer – Романова Ирина Сергеевна.

На Рисунке 2 представлены перечень задач для выполнения и примерные сроки их реализации.

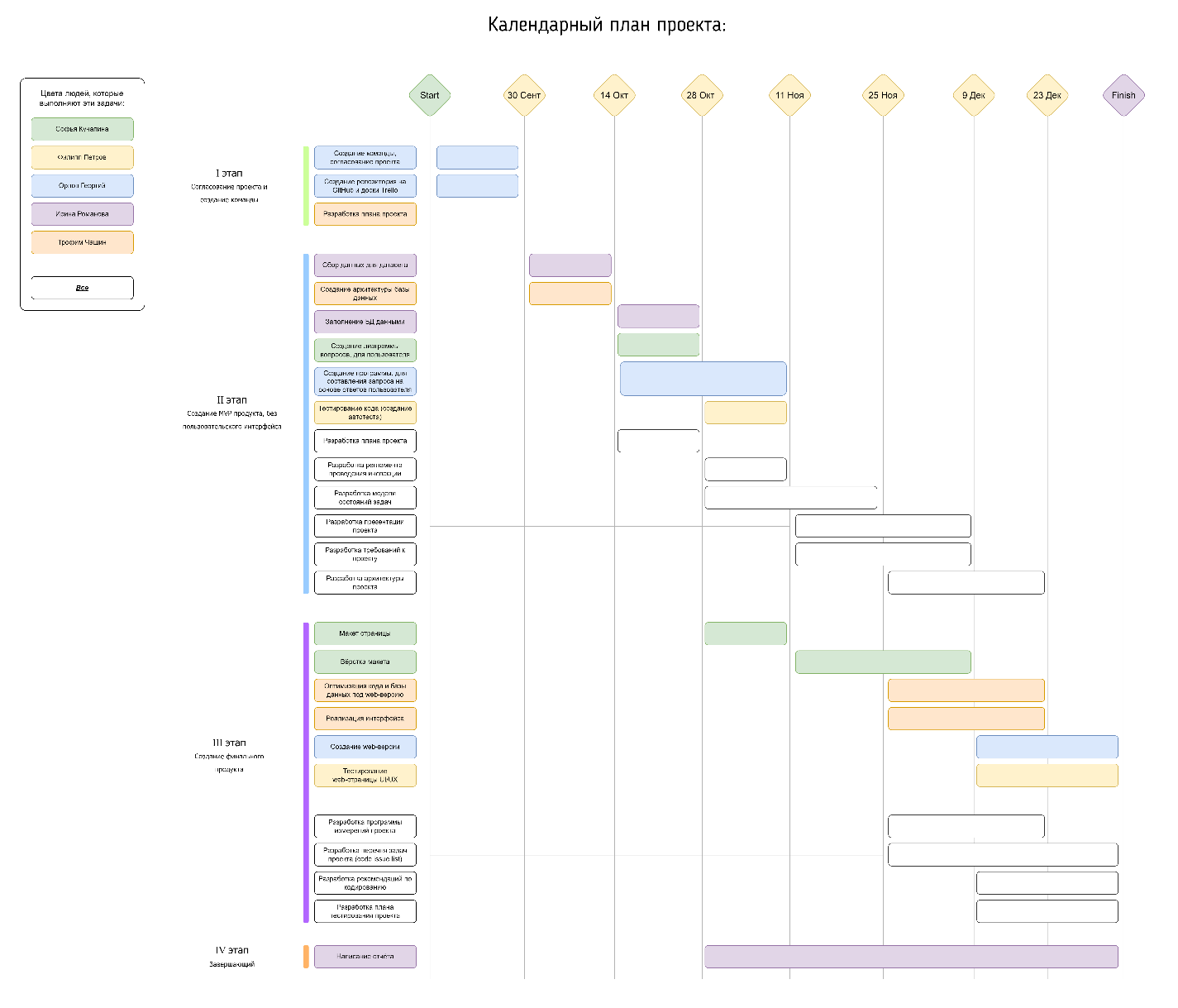


Рисунок 2 – Календарный план проекта

# 2. Регламент 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.

**2.1 Виды инспекций**

Существует несколько разновидностей инспекций. В самом общем случае их подразделяют на формальные (проводимые по специальному регламенту) и неформальные (проводимые в рабочем порядке) инспекции. Критерием в пользу проведения того или иного типа инспекции служит размер инспектируемого рабочего продукта. Данный критерий по отношению к формальным инспекциям может быть, например, следующим:

|  |  |
| --- | --- |
| **Рабочий продукт** | **Размер продукта или изменения в нем, не менее** |
| Требования | 5 страниц или 10% рабочего продукта |
| Документы по дизайну | 5 страниц или 10% рабочего продукта |
| Не комментированный код | 50 NCLOC |
| Тесты (Test Cases) | 5 страниц или 10% рабочего продукта |

**2.2 Участники инспекции**

Участники инспекции могут иметь следующие роли:

1. Автор (Author) - сотрудник, разработавший инспектируемый рабочий продукт, либо сделавший инспектируемые изменения в существующем рабочем продукте;
2. Председатель (Moderator) - ответственный сотрудник, выполняющий роль председателя инспекции;
3. Секретарь (Recorder) - сотрудник, ответственный за создание и распространение документации по инспекции;
4. Ведущий (Presenter) - сотрудник, представляющий рабочий продукт инспекторам;
5. Инспектор (Inspector) - сотрудник, ответственный за эффективную проверку инспектируемого рабочего продукта.

Один человек может играть сразу несколько ролей. В случае совмещения ролей рекомендуется, чтобы один человек играл не более двух ролей одновременно. В инспекции должны участвовать не менее 3 человек. При назначении одного человека на несколько ролей следует руководствоваться следующим правилом: не допустимо совмещение ролей Председатель и Автор, Автор и Ведущий, Автор и Секретарь, Автор и Инспектор. Все остальные сочетания ролей допускаются.

**2.3 Этапы инспекции**

**2.3.1 Планирование инспекции**

Все инспекции на конкретном проекте проводятся согласно принятому на предприятии стандартному процессу, планируются на фазе планирования проекта и фиксируются в документе SQAP (Software Quality Assurance Plan).

Кроме того, при изменении любого продукта должна быть проведена инспекция этих изменений в том случае, если объем этих изменений превышает определенные значения, служащие критериями проведения формальных инспекций.

При планировании инспекции коллективно выбирается дата, время, формат (очный или заочный) и платформа (при заочной инспекции) проведения инспекции. Эти параметры определяются с учетом удобства и доступности участников инспекции, обеспечивая эффективное проведение процесса и обмена необходимой информацией.

Секретарю, ведущему и инспектору необходимо принять или отклонить приглашение на формальную инспекцию с обязательным указанием причин.

**2.3.2 Назначение инспекции**

Автор должен оповестить руководителя проекта (или руководителя группы) о готовности рабочего продукта к формальной инспекции и зафиксировать промежуточную стабильную версию рабочего продукта. С момента предоставления рабочего продукта на формальную инспекцию до собрания включительно автор не имеет права вносить изменения. Также автор должен запросить руководителя проекта об имеющихся ресурсах на роль председателя инспекции. В случае необходимости, провести обзорное собрание, целью которого является ввести участников формальной инспекции в курс дела.

Председатель должен убедиться, что рабочий продукт удовлетворяет критерию готовности к формальной инспекции. Также председателю нужно определить необходимость проведения обзорного собрания и проверить, что все обязательные участники формальной инспекции приняли приглашение либо прислали отказ с обязательным указанием причин и/или предложением своих сроков проведения инспекции. В случае получения хотя бы одного отказа, председатель должен проанализировать сложившуюся ситуацию на предмет того, может ли быть проведена инспекция без отказавшегося участника.

Кроме того, председатель или автор должны обеспечить следующее: 1) определить материально – техническую сторону проведения инспекции, 2) подобрать команду участников и распределить роли, 3) оповестить всех участников формальной инспекции.

**2.3.3 Подготовка к инспекции**

Инспектор самостоятельно независимо от других участников изучает предоставленный для инспекции рабочий продукт, используя накопленный опыт и стандарты. Также ему требуется заполнить необходимые поля протокола подготовки к формальной инспекции и отправить его председателю формальной инспекции и автору.

На основе полученных от инспекторов протоколов подготовки к формальной инспекции, председателю необходимо принять одно из следующих решений: провести, перенести или отменить инспекцию. После принятия решения председатель должен оповестить всех участников формальной инспекции об изменениях.

До собрания автору необходимо ознакомиться с содержанием полученных от инспекторов протоколов подготовки к формальной инспекции, проанализировать изложенные в них замечания.

Секретарь должен проанализировать замечания, зафиксированные инспекторами в протоколах подготовки к формальной инспекции, выявить повторяющиеся.

**2.3.4 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.

**2.3.5 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.

**2.4 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.
2. Дефект – проблема, которая найдена на фазе, отличной от той, на которой внесена.
3. Замечание для исследования – проблема, природа которой не может быть определена на собрании и требует дополнительного исследования. В результате дополнительного исследования такая проблема должна получить одно из вышеупомянутых значений статусов.

**2.5 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**

1. New – новая подзадача.

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

3. Analysis completed – анализ завершён. Переводится сотрудником после завершения анализа задачи.

4. Terminated – прерванный. В это состояние задача может быть переведена CCB после рассмотрения и принятия решения о необходимости отмены, прекращения работы над задачей. В это состояние задача может быть переведена, например, сразу после создания новой или по результатам её анализа.

5. 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.

6. Coding – кодирование. В это состояние задача переводится сотрудником разработчиком, при начале работы по кодированию, связанному с задачей.

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

8. Resolved – проверено. Переводится после проверки изменений рабочего продукта по результатам инспектирования специалистом-экспертом (code expert).

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

10. Tested - протестировано. Переводится сотрудником, осуществляющим тестирование изменений в рабочий продукт (tester).

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

**Правила создания новой задачи**

* + - 1. *Описание задачи*. Задача должна быть ясно и чётко описана, чтобы каждый член команды мог вникнуть в суть. В описание необходимо включить цели и ожидаемые результаты;
      2. *Требования и критерии завершённости*. Необходимо указать требования к задаче и определить критерии завершённости для чёткой оценки выполнения;
      3. *Приоритет и сроки*. Для эффективной разработки проекта, необходимо указать приоритет новой задачи относительно других задач и установить реалистичные сроки выполнения;
      4. *Ресурсы*. Требуется определить необходимые ресурсы и обозначить зависимости от других задач или компонентов проекта;
      5. *Тестирование*. Нужно определить требования к тестированию и проверке новой задачи и указать возможные тестовые сценарии;
      6. *Документация*. Завершить задачу с необходимой документацией, чтобы облегчить понимание и поддержание кода.

Так как информирование о создании новой задачи осуществляется для всей команды, то приступить к разработке может приступить любой член команды. Решением новой задачи будут заниматься участник/участники, у которых меньше всего работы на тот момент времени, когда задача появилась. Это сделано для того, чтобы работа распределялась равномерно.

**Правила перехода задач из состояния в состояние**

В основном, состояния задач идут последовательно друг за другом(по стрелкам). В некоторых случаях состояния могут возвращаться на несколько назад или пропускать некоторые состояния. Схема перехода из состояния в состояние показана на Рисунке 3.

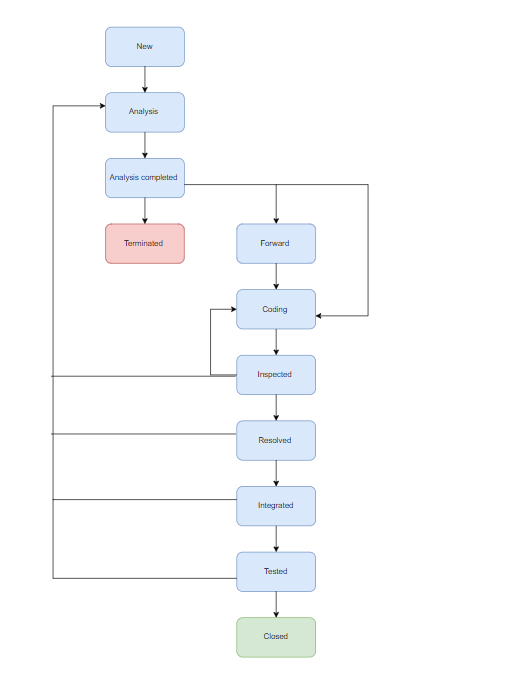
****

Рисунок 3 – Схема перехода из состояния в состояние

Примечания:

* Задача переходит в состояние *Terminated* только в самых критических ситуациях: когда не хватает ресурсов или при анализе были сделаны выводы о нецелесообразности работы над данной задачей;
* После состояния Analysis completed задача переходит в состояние

Forward. На данном этапе Team Leader может назначать на задачу исполнителя и инспектора. Также из состояния Analysis completed можно перейти в Coding, если на выполнение задачи не переназначали исполнителя/исполнителей;

* Из состояния Inspected возможно вернуться в Coding и Analysis, если в ходе инспектирования нашлись ошибки или возникли вопросы к решению задач;
* Из состояний Resolved, Integrated и Tested тоже есть возможность вернуться в состояние Analysis, так как появляются ошибки и новые идеи по преобразованию решения задачи;
* В состояние Closed можно перейти только в случае полной уверенности в выполнении задачи.

# 4. Презентация проекта

*Я так понимаю, это уже финальная презентация проекта, а не то, что у нас было. Поэтому тут пока что ничего нет*

# 5. Требования к проекту

Программный продукт: Сервис подбора растений для дома

Данный онлайн-сервис предназначен для подбора растений, по заданным характеристикам

Онлайн сервис состоит из следующих подсистем:

1. База данных растений
2. Алгоритм подбора растений, по введённым характеристикам
3. Модуль запросов к БД
4. Модуль хостинга онлайн-сервиса
5. Пользовательский интерфейс

*Тут какие-то идентификаторы, нам про это пока что не рассказывали. Так что напишу пока что требования, без них*

* Продукт должен иметь пользовательский интерфейс.
* Продукт должен запускаться и корректно отображаться на любом устройстве, поддерживающим выход в интернет, и просмотр web-страниц. В том числе:
  + Настольные компьютеры.
  + Планшеты.
  + Телефоны.
* Продукт должен подобрать одно или несколько растений (в зависимости от введённых характеристик).
  + В случае, когда для перечня введённых характеристик нет ни одного растения в Базе данных, продукт должен уведомить пользователя о том, что ему не удалось подобрать ни одного растения.
* В Базе данных (далее - БД) должны содержаться записи о растениях, покрывающие большее количество возможных запросов, с указанием характеристик (> 90% всех запросов должны возвращать результат), однако, не учитывая характеристики, связанные с цветом растения.
* Модуль хостинга онлайн-сервиса должен быть всегда доступен.
* Модуль запросов к БД должен быть защищён от SQL-инъекций.
* Контроль версий, на всех стадиях разработки продукта, должен вестись на платформе GitHub.
* В структуре онлайн-сервиса, должны быть следующие страницы, с данным наполнением:

1. Главная страница – подбора растения по вводимым характеристикам.
   * Онлайн сервис должен по умолчанию открываться на этой странице.
2. Страница информации об авторах.
   * Ссылка на данную страницу должна располагаться в футере, на всех других страницах.
   * На данной странице должна быть кнопка перехода на главную страницу.
3. Страница просмотра всего каталога растений
   * На данной странице должны отображаться все растения, имеющиеся в БД, с их фотографиями.
   * Изначально, на странице открывается небольшое число записей о растениях (10-15), и записи должны автоматически загружаться, когда пользователь пролистывает страницу вниз.
   * Внизу и вверху страницы должна быть кнопка перехода на главную страницу.

* В пользовательском интерфейсе должны быть следующие элементы:

1. Кнопка старта.
   * По нажатию на кнопку старта должны последовательно появляться вопросы, на которые пользователь может дать ответ.
2. Кнопка открытия меню.
   * Кнопка закрытия меню.
   * Кнопка перехода на страницу об авторах сервиса.
   * Кнопка перехода на страницу с каталогом растений.
3. Если в вопросе есть поле ввода, оно должно проверять введённые данные на корректность, и выводить соответствующее сообщение о некорректно введённых данных.
4. При изменении одного из предыдущих ответов, все последующие ответы должны сбрасываться, и скрываться
5. После ответа на все вопросы, должна отобразиться кнопка «Показать результаты».
   * После нажатия на эту кнопку, должны отобразиться результаты, либо сообщение о том, что сервис не смог подобрать растения, по заданным характеристикам.

* Алгоритм подбора растений, должен получать набор характеристик. Введённых пользователем, преобразовывать их в SQL-запрос, и передавать модулю запросов к БД

# 6. Архитектура 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.

*… Вот сюда АКД вставлять?*

*Я сделал вот такую АКД. Я не знаю, правильная ли она*

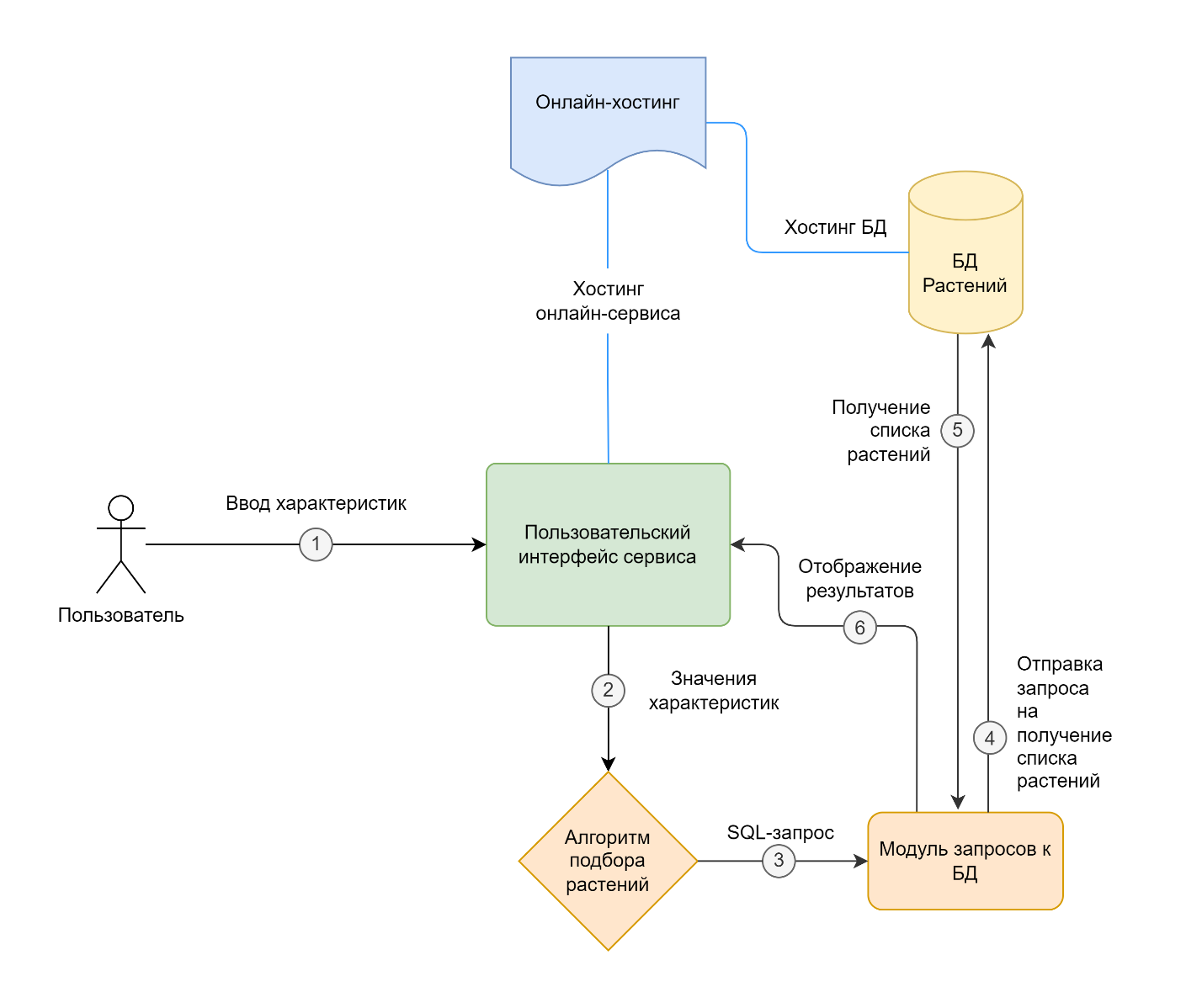
**

Рисунок 4 – Архитектурно-контекстная диаграмма

# 7. Метрики проекта

Метрики проекта — это количественные показатели, которые используются для измерения и оценки эффективности и успеха проекта. Они предоставляют объективную информацию о производительности проекта и помогают команде принимать обоснованные решения на основе данных.

Метрики проекта важны по следующим причинам:

**Оценка производительности:** Метрики проекта помогают оценить, насколько хорошо проект выполняется по отношению к заранее определенным целям и стандартам.

**Управление рисками:** Метрики проекта могут выявить потенциальные проблемы и риски, позволяя команде принять меры для их устранения или смягчения.

**Принятие решений:** Метрики проекта предоставляют данные, которые могут использоваться для принятия обоснованных решений о том, как управлять и улучшать проект.

**Отчетность:** Метрики проекта обеспечивают прозрачность и отчетность перед стейкхолдерами, позволяя им видеть, как проект продвигается и достигает своих целей.

Таблица 2 - Описания измерений

|  |  |  |
| --- | --- | --- |
| **Номер измерения** | **Дата измерения** | **Описание состояния проекта** |
| 1 | 7.11.2023 | Согласование проекта и БД растений |
| 2 | 11.11.2023 | Написание чернового алгоритма подбора |
| 3 | 16.11.2023 | Сбор информации для БД |
| 4 | 23.11.2023 | Создание сервиса как сайта |
| 5 | 27.11.2023 | Хостинг сервиса и БД |
| 6 | \_\_.\_\_.\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**7.1 Метрика IFD**

IFD (Integrated Function Deployment) - это метрика, которая показывает, насколько хорошо проект сочетает функциональность разных модулей или компонентов в единую систему. IFD измеряется по количеству успешно интегрированных функций, а не по количеству функций в целом. Чем выше IFD, тем лучше проект работает как целое.

Таблица 3 – IFD

|  |  |  |  |
| --- | --- | --- | --- |
| **Измерение** | **Количество модулей** | **Успешно**  **интегрированные**  **модули** | **IFD** |
| 1 | 5 | 1 | 20% |
| 2 | 5 | 2 | 40% |
| 3 | 5 | 3 | 60% |
| 4 | 5 | 4 | 80% |
| 5 | 5 | 5 | 100% |

**7.2 Метрика False Scrining (FS)**

False Scrining (FS) - это метрика, которая показывает, сколько времени и ресурсов потребовалось для проверки и исправления ошибок в коде или документации проекта. FS измеряется по количеству ошибок, которые были обнаружены и устранены до сдачи продукта заказчику или пользователям.

Таблица 4 - False Scrining (FS)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Измерение** | **Количество готовых модулей** | **Модули с ошибками** | **Время на исправление**  **ошибок (часов)** | **FS** |
| 1 | 1 | 1 | 25 | 25 часов на модуль |
| 2 | 2 | 0 | 0 | 0 часов на модуль |
| 3 | 3 | 2 | 37 | 18,5 часов на модуль |
| 4 | 4 | 2 | 6 | 3 часа на модуль |
| 5 | 5 | 4 | 28 | 7 часов на модуль |

**7.3 Метрика IPF**

IPF (Integrated Product Functionality) - это метрика, которая показывает, насколько хорошо проект соответствует требованиям заказчика или пользователей по функциональности продукта. IPF измеряется по количеству функций, которые были реализованы и протестированы на практике.

Таблица 5 – IPF

|  |  |  |  |
| --- | --- | --- | --- |
| **Измерение** | **Количество функций** | **Реализованные и**  **протестированные функции** | **IPF** |
| 1 | 12 | 1 | 8% |
| 2 | 12 | 6 | 50% |
| 3 | 12 | 7 | 58% |
| 4 | 12 | 10 | 83% |
| 5 | 12 | 11 | 92% |

**Описание функций:**

1-2: Тестовый подбор растений, из ограниченного списка

3: База данных

4-5: Подбор растений по полной БД

6: Улучшение подбора до 90%

7: Тестовая страница сервиса

8-9: Внедрение подбора характеристик

10: Хостинг сервиса и БД

11: Добавление страницы с просмотром всех растений

12: Доработка сервиса, и исправление багов

На основе данных, представленных в таблицах, можно сделать вывод, что проект развивается и улучшается с течением времени. Метрики IFD, FS и IPF показывают положительную динамику, что говорит о том, что команда проекта успешно справляется с задачами по интеграции модулей, исправлению ошибок и реализации функциональности. Это подтверждает, что проект развивается в правильном направлении.

# 8. Перечень задач проекта

*Хз что это, и как это расписывать*

# 9. Стандарты кода

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.

В основе рекомендаций — руководство по оформлению кода на JavaScript.

**9.1 Рекомендации и требования к оформлению кода**

1. В начале большого файла с кодом, должно быть краткое описание всего кода в этом файле
2. Большие блоки кода, которые отличаются по назначению, стоит разделять большими, видимыми комментариями, для того, чтобы упростить ориентирование в коде. Например:

Изображение выглядит как снимок экрана, линия

Автоматически созданное описание

Рисунок 5 – Пример большого комментария

1. Все названия переменных начинаются с маленькой буквы
2. Все названия методов и функций, начинаются с большой буквы
3. Общая часть названия метода или переменной, записывается CamelCase
4. Для описания имён переменных и методов, рекомендуется использовать большое количество слов. Так, чтобы при взгляде на название переменной, сразу становилось понятно её назначение. Например:



Рисунок 6 – Пример наименования переменной

1. Нужно оставлять достаточное количество комментариев, для пояснения различных частей кода, назначение которых может быть непонятно с первого взгляда
2. Согласно стандартам написания кода на языке JS, открывающаяся фигурная скобка { ставится на той же строке, что и объявленная конструкция
3. В конце каждой строки, желательно ставить точку с запятой ;
4. Обязательно нужно использовать ключевое слово let, для обозначения новой переменной (не использовать ключевое слово var)

# 10. План тестирования проекта

*Мы это ещё не проходили*

# 11. Тестирование проекта

*Сначала нужно сделать план тестирования)*

# Заключение

В рамках курсовой работы было разработано программное средство «Сервис подбора растений для дома» 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.
11. Протестирован проект.

Согласно метрикам проекта, все модули были успешно интегрированы в единую систему, количество ошибок в модулях уменьшилось, как и время, затраченное на их исправление. А также, были реализованы и протестированы все необходимые функции.

Таким образом, цель данной курсовой работы была достигнута.

# Список литературы

1. Гриняк В.М. Лекции по дисциплине «Технологии коллективной промышленной разработки информационных систем». Электронный вариант.
2. Жизненный цикл информационных систем [Электронный ресурс] // Режим доступа: <https://edu.tltsu.ru/sites/sites_content/site216/html/media67140/lec2_is-2.pdf>
3. Методология и технология разработки информационных систем [Электронный ресурс] // Режим доступа: <https://edu.tltsu.ru/sites/sites_content/site216/html/media67140/lec4_is-2.pdf>
4. Попов Г.Ю., Кондратьев А.А., Тищенко И.П., Фраленко В.П. Разработка информационной системы поддержки коллективной разработки проектов / Программные системы: теория и приложения : электрон. научн. журн. 2012. Т. 3, №2(11), с. 3-22. [Электронный ресурс] // Режим доступа: <https://cyberleninka.ru/article/n/razrabotka-informatsionnoy-sistemy-podderzhki-kollektivnoy-razrabotki-proektov>
5. Попов И.И. Информационные ресурсы и системы: реализация, моделирование, управление.–М.: ТПК АЛЬЯНС, 1996.– 408 с. Проектирование и дизайн пользовательского интерфейса / А.К. Гультяев., В.А. Машин.– СПб.: Коронапринт, 2000.–352 с. [Электронный ресурс] // Режим доступа: <http://inftis.narod.ru/is/learn-is-lit.htm>