  
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 MOCK СЕРВЕРА**

КУРСОВАЯ РАБОТА  
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  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Спирин Д.А.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Исаков Т.А.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сапогов И.М.  
Руководитель:  
старший преподаватель департамента ПИиИИ  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю.С.

г. Владивосток  
2024

Оглавление

[Введение 3](#_Toc124874819)

[1 Разработка плана проекта 4](#_Toc124874820)

[2 Разработка регламента проведения инспекции 5](#_Toc124874821)

[3 Разработка модели состояний задач 8](#_Toc124874822)

[4 Разработка презентации проекта 10](#_Toc124874823)

[5 Разработка требований к проекту 13](#_Toc124874824)

[6 Разработка архитектуры проекта 17](#_Toc124874825)

[7 Разработка измерений проекта 19](#_Toc124874826)

[8 Разработка перечня задач проекта 21](#_Toc124874827)

[9 Разработка рекомендаций по кодированию 23](#_Toc124874828)

[10 Разработка плана тестирования проекта 25](#_Toc124874829)

[11 Тестирование проекта 27](#_Toc124874830)

[Заключение 31](#_Toc124874831)

[Список литературы 32](#_Toc124874832)

# Введение

С развитием веб-приложений становится все более ясной важность тестирования, особенно в контексте фронтенд-разработки. Однако, часто возникает проблема тестирования фронтенда, когда реальный бэкенд еще не готов или когда требуется отдельное, контролируемое окружение для проведения тестов. В таких ситуациях важно иметь возможность эмулировать поведение реального сервера, и здесь на помощь приходит технология мок-серверов.

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 «Moniken» и составление технической документации к данному средству.

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;
* Протестировать проект.

# 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 — Спирин Д.А.
* BackEnder — Спирин Д.А.
* FrontEnder — Исаков Т.А.
* Technical Writer — Сапогов И.М.

Был разработан перечень задач для выполнения и примерные сроки их реализации [Рисунок 1].

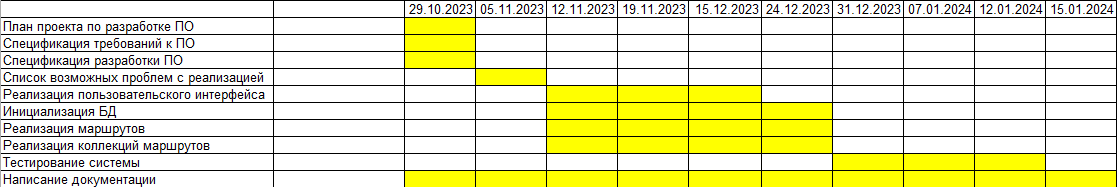


Рисунок 1 — План проекта

# Разработка регламента проведения инспекции

Проверка рабочих продуктов — это ключевой этап в обеспечении их качества. Современная практика разработки программного обеспечения включает в себя специальные стандарты, методы и инструменты для проведения так называемых инспекций (peer reviews).

Инспекция представляет собой событие, направленное на контроль качества рабочих продуктов в рамках проектов по разработке ПО и других областей, и проводится разработчиками, часто с участием представителей заказчика. Основные цели инспекции заключаются в том, чтобы:

* Выявлять ошибки в функциональности, логике, содержании или реализации продуктов на ранних этапах разработки и предотвращать их дальнейшее распространение;
* Ясно и осознанно представлять задумку или реализацию продукта всем заинтересованным сторонам, вовлекая их в процесс;
* Оптимизировать, оценить или улучшить рабочий продукт, добавляя ценности с самого начала его создания.

## Критерии формальности инспекции

Неформальная инспекция проводится:

* В случае изменения участка документа, содержащего не более 5 строк, для текстовых документов;
* В случае изменения не более 5 элементов для документов дизайна.

Формальная инспекция проводится в случае невозможности проведения неформальной инспекции.

## Участники инспекции

Участники могут иметь следующие роли:

1. Автор — участник, внёсший изменения в рабочий продукт. Загружает изменения в систему контроля версий и инициирует инспекцию.
2. Председатель — тимлид (Team Leader), контролирующий процесс инспекции. Назначает инспектора и одобряет внесение ключевых изменений в основную версию продукта.
3. Инспектор — участник, проводящий проверку внесённых изменений. Оставляет замечания и выносит вердикт о внесении изменений в продукт.

В инспекции в обязательном порядке присутствуют два участника, имеющие роли автора и инспектора соответственно. При внесении изменений в дизайн рабочего продукта, необходимо присутствие председателя, который может также выполнять роль инспектора.

## Этапы инспекции

1. Инициация — создание автором запроса на внесение изменений в продукт и назначение инспектора.
2. Подготовка и проведение — анализ изменений и внесение замечаний инспектором.
3. Завершение — вынесение вердикта о внесении изменений в продукт.

## Порядок организации инспекции

Работа над продуктом ведётся в системе контроля версий GIT. Автор изменений оформляет Pull Request (Merge Request) и назначает инспектора, отправляет в рабочий чат в Telegram сообщение с упоминанием инспектора.

Столкнувшись с изменением дизайна проекта, инспектор обращается к председателю инспекции в рабочем чате в Telegram. По окончании своей работы инспектор одобряет Pull Request или отправляет на доработку автору, уведомляя его в рабочем чате в Telegram.

## Порядок подготовки и проведения инспекции

Инспекция должна быть проведена в течение 7 дней с момента её инициации.

После анализа изменений инспектор оставляет в системе контроля версий замечания, обозначая степень их важности. При наличии замечаний, требующих исправлений, работа передаётся автору на доработку. При отсутствии подобных замечаний инспекция считается завершённой, и изменения вступают в силу.

## Перечень статусов и степени важности замечаний

1. Комментарий — рекомендация по улучшению продукта, не требующая обязательных изменений.
2. Ошибка — замечание, сообщающее о необходимости исправления.
3. Отклонение – запрос на внесение изменений не является актуальной или в нем нет необходимости.
4. Одобрение – запрос на внесение изменений одобрено и вступает в силу.

## Порядок верификации учёта замечаний

После внесения повторных изменений инспектор просматривает замечания и проверяет соответствующие изменения. По окончании верификации выносится вердикт о внесении изменений в продукт или выносятся новые замечания.

## Метрики, характеризующие эффективность инспекций

Inspection Rate(IR): IR = Размер продукта / Общее время инспектирования

Стратегическая цель метрики — повысить качество разрабатываемого ПО.

Изучаемый объект метрики — проведение инспекции, измеряемый атрибут — производительность инспектирования.

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

# 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 возможных состояний задач и их интерпретация

* New — новая подзадача.
* Analysis — подзадача находится в процессе анализирования, то есть, как только участник команды начнет анализ подзадачи, подзадача перейдет в статус анализирования.
* Progress — это состояние задачи подразумевает переход в стадию «Дальнейшая разработка». Подзадача может перейти в это состояние только после анализирования и назначения определенного участника команды, как ответственного.
* 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.
* 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.

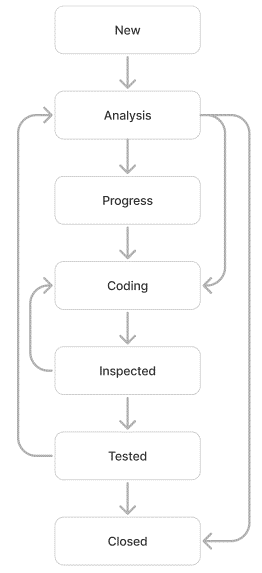
## Правила создания новой задачи

Любой участник команды разработки может в любое время создавать задачи в рамках назначенной ему части проекта.

## Правила перехода задачи из состояния в состояние

Состояния задач всегда идут последовательно друг за другом, в

некоторых случаях пункты могут опускаться или повторяться:



# Разработка презентации проекта

Была разработана презентация проекта, состоящая из шести слайдов:

1. Титульный слайд [Рисунок 2]
2. Команда разработки [Рисунок 3]
3. Проблема пользователей [Рисунок 4, Рисунок 5]
4. Решение проблемы пользователей [Рисунок 6, Рисунок 7]
5. Интерфейс программы [Рисунок 8]
6. Основные возможности программы [Рисунок 9]

Изображение выглядит как текст, снимок экрана, Шрифт, белый

Автоматически созданное описание

Рисунок 2 — Титульный слайд

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Рисунок 3 — Команда разработки

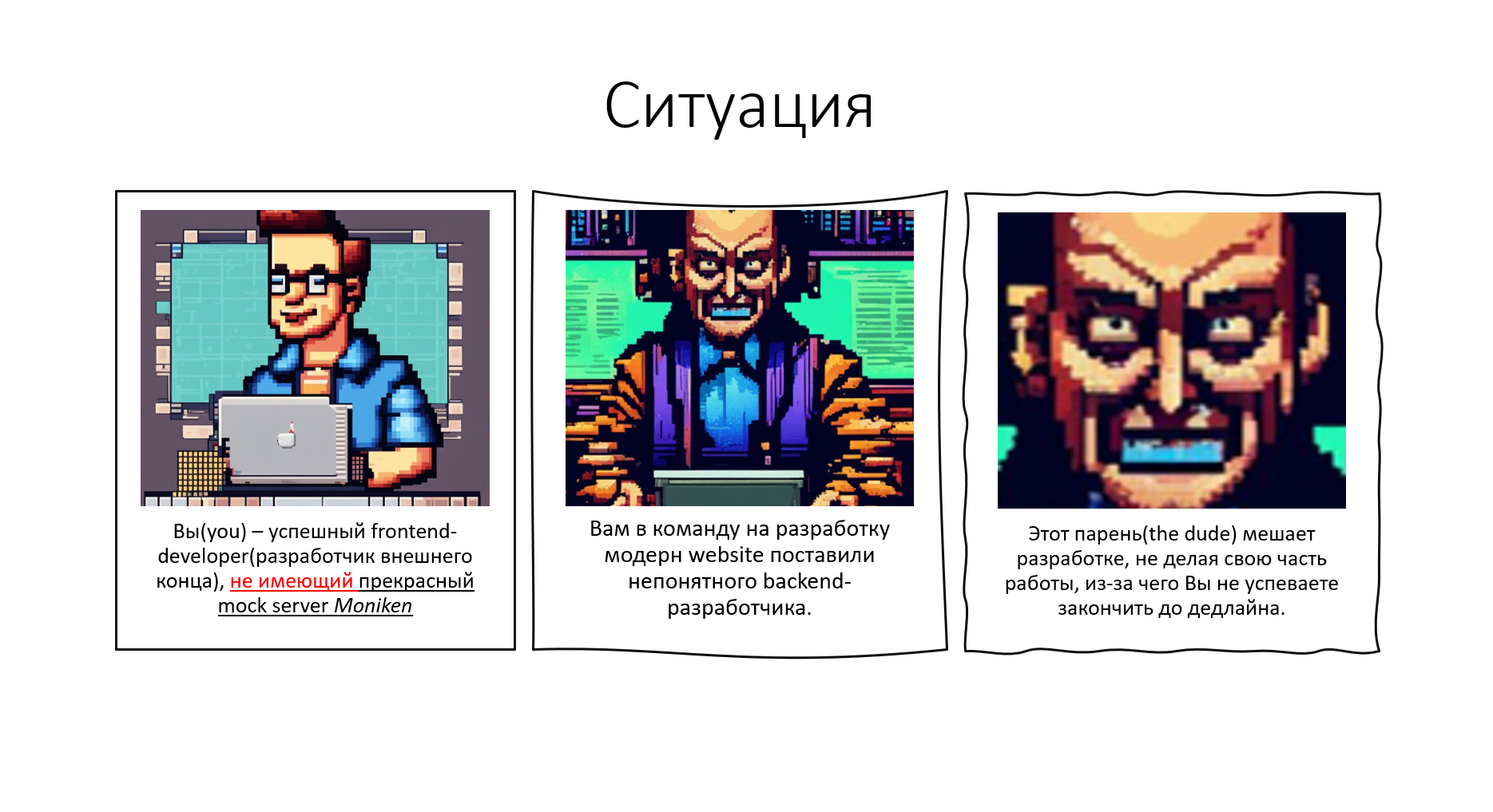


Рисунок 4 —Проблема пользователей

Изображение выглядит как текст, снимок экрана, мультфильм

Автоматически созданное описание

Рисунок 5 — Проблема пользователей

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

Автоматически созданное описание

Рисунок 6 — Решение проблемы пользователей

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7 — Решение проблемы пользователей

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 8 — Интерфейс программы

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9 — Основные возможности программы

# Разработка требований к проекту

Программный продукт: «Moniken» — простой mock сервер, разработанный с помощью Spring Boot, который может использоваться для тестирования и разработки фронтенда без зависимости от реального сервера.

Программный продукт предназначен для:

1. Создания, управления и обрабатывания запросов от встроенного мок-сервера, что позволяет разрабатывать фронтенд приложения без зависимости от реального сервера.
2. Разработки на ранних этапах, особенно в случаях, когда развитие бэкэнда и фронтэнда проводится независимо друг от друга.
3. Создания точной спецификации взаимодействия между серверной и клиентской частями приложения.

Программный продукт состоит из следующих подсистем:

1. FE-1 — Пользовательский интерфейс.
2. FE-2 — Модуль управления коллекциями маршрутов.
3. FE-3 — Модуль управления маршрутами.

Матрица требований к подсистемам

|  |  |
| --- | --- |
| **Матрица требований** | **Идентификаторы** |
| Пользовательский интерфейс | FE-1 UI |
| Операции всех подсистем в веб-интерфейсе | REQ-UI-1 |
| Операции с модулем управления коллекциями маршрутов | FE-2 RCMM |
| Просмотр существующих коллекций | REQ-RCMM-1 |
| Добавление коллекции | REQ-RCMM-2 |
| Удаление коллекции | REQ-RCMM-3 |
| Обновление коллекции | REQ-RCMM-4 |
| Операции с модулем управления маршрутов | FE-3 RMM |
| Просмотр существующих маршрутов | REQ-RMM-1 |
| Добавление маршрута | REQ-RMM-2 |
| Удаление маршрута | REQ-RMM-3 |
| Обновление маршрута | REQ-RMM-4 |

Расшифровка идентификаторов

* RCMM – Rote Collection Management Module (Модуль управления коллекциями маршрутов)
* RMM – Rote Management Module (Модуль управления маршрутами)

# Разработка архитектуры проекта

Архитектура программного обеспечения относится к основополагающим структурам программной системы и методам разработки этих структур. Каждая из этих структур включает в себя элементы программного обеспечения, их взаимосвязи, а также характеристики как самих элементов, так и их взаимосвязей. Архитектура программной системы аналогична архитектуре здания: она представляет собой план для системы и процесса её разработки, в котором разъясняются задачи, стоящие перед командами разработчиков.

Для представления архитектура проекта были спроектированы ресурсы API [Рисунок 10], АКД-диаграмма [Рисунок 11] и диаграмма взаимодействия [Рисунок 12].



Рисунок 10 — Проектирование ресурсов API



Рисунок 11 — АКД

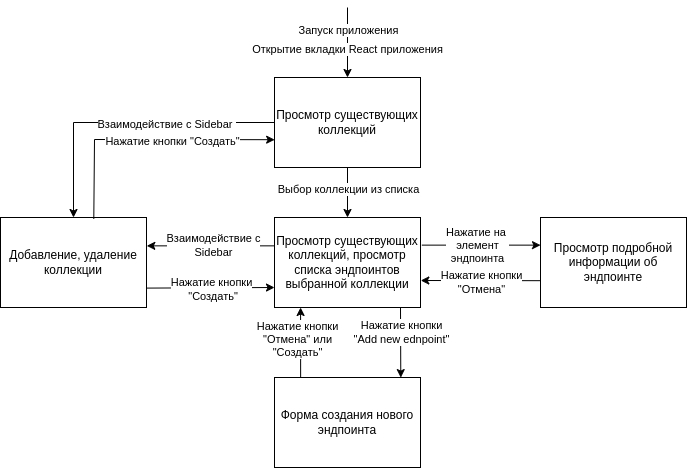


Рисунок 12 — Диаграмма взаимодействия

# Разработка измерений проекта

Управление производственным процессом и контроль за его результатами играют решающую роль на любом современном предприятии, занимающемся заказной разработкой программного обеспечения. Учитывая уникальные особенности программного продукта, особое внимание уделяется оценке эффективности процесса и качества конечного продукта с применением специальных подходов. Совокупность мероприятий, направленных на количественную оценку работы, называется программой измерений.

## Метрики эффективности процесса разработки

1. Inspection Fault Density (IFD)

IFD = (количество обнаруженных ошибок) / (объем инспектированного кода)

Стратегическая цель метрики — повысить качество разрабатываемого ПО.

Изучаемый объект метрики — инспекции.

Измеряемый атрибут — плотность, найденных в ходе инспекций ошибок.

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

2. Faults Screening (FS)

FS = (количество обнаруженных дефектов в процессе скрининга) / (общее количество обнаруженных дефектов)\*100%

Стратегическая цель метрики — повысить качество разрабатываемого ПО.

Изучаемый объект метрики — проект.

Измеряемый атрибут — эффективность обнаружения дефектов.

Единица измерения — %.

3. In Process Faults (IPF)

IPF = (количество выявленных дефектов в конкретной фазе) / (общее количество выявленных дефектов)

Стратегическая цель метрики — повысить качество разрабатываемого ПО.

Изучаемый объект метрики — проект.

Измеряемый атрибут — плотность неполадок.

Единица измерения — количество выявленных дефектов в конкретной фазе / общее количество выявленных дефектов.

# Разработка перечня задач проекта

## Подсистема FE-1

TASK-1 UI

Пользовательский интерфейс

До 15.12.2023 реализовать пользовательский интерфейс для работы с системой. Интерфейс должен включать в себя элементы всех подсистем системы (модуль управления коллекциями маршрутов, модуль управления маршрутами) – REQ-UI-1.

## Подсистема FE-2

TASK-1 RCMM

Модуль управления коллекциями маршрутов

До 24.12.2023 реализовать коллекции маршрутов с добавлением, удалением, обновлением и их просмотром – REQ-RCMM-1, REQ-RCMM-2, REQ-RCMM-3, REQ-RCMM-4.

## Подсистема FE-3

TASK-1 RMM

Модуль управления маршрутами

До 24.12.2023 реализовать маршруты с добавлением, удалением, обновлением и их просмотром – REQ-RMM-1, REQ-RMM-2, REQ-RMM-3, REQ-RMM-4.

# 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.

В основе рекомендаций — конвенция кода Java от 1997. Написанный код на языке Java должен всецело придерживаться данной конвенции.

Помимо того, необходимо придерживаться следующих правил:

* Переопределение методов equals и hashCode, при этом должны удовлетворяться условия контрактов equal и hashCode, как по отдельности, так и их общий контракт.
* Переопределение методов toString в каждом data-классе.
* Все обьекты не относящиеся к data-обьектам должны создаваться либо при помощи фабрик, либо быть bean-компонентами.
* Многострочные комментарии кратко, но полно обьясняют алгоритм функции, однострочные комментарии поясняют отдельную строку алгоритма.
* Обьекты-модели (Domain-обьекты) называются без суффиксов Model, все различные преобразования обьекта-модели должны содержать суффикс, определяющий их принадлежность к логике, которая была применена к обьекту-модели. Например, если обьект модели Route, то производные обьекты, обозначающие конвертацию для клиентского представления, должны иметь суффикс DTO.

# Разработка плана тестирования проекта

* **TEST\_UI**

Тестируемые требования: REQ-UI-1, REQ-RCMM-1, REQ-RMM-1

Описание теста: выбор коллекции, нажав на одну из кнопок, представляющие их, в боковом меню пользовательского интерфейса

Ожидаемый результат: обновление основного окна, после которого в основном окне будут отображаться маршруты выбранной коллекции

* **TEST\_UI\_RCMM\_1**

Тестируемые требования: REQ-UI-1, REQ-RCMM-2

Описание теста: Нажать на кнопку добавлении коллекции

Ожидаемый результат: добавление нового элемента, представляющего добавляемую коллекцию, в боковом меню пользовательского интерфейса и добавление новой коллекции

* **TEST\_UI\_RCMM\_2**

Тестируемые требования: REQ-UI-1, REQ-RCMM-3

Описание теста: Нажать на кнопку удаления коллекции

Ожидаемый результат: удаление элемента, представляющего удаляемую коллекцию, в боковом меню пользовательского интерфейса и удаление коллекции

* **TEST\_UI\_RCMM\_RMM\_1**

Тестируемые требования: REQ-UI-1, REQ\_RCMM\_4, REQ-RMM-2

Описание теста: Нажать на кнопку добавлении маршрута, после чего ввести корректный формат маршрута

Ожидаемый результат: добавление нового маршрута в отображаемой коллекции с последующим отображением в окне

* **TEST\_UI\_RCMM\_RMM\_2**

Тестируемые требования: REQ-UI-1, REQ\_RCMM\_4, REQ-RMM-2

Описание теста: Нажать на кнопку добавлении маршрута, после чего ввести некорректный формат маршрута

Ожидаемый результат: предупреждение о неправильном вводе

* **TEST\_UI\_RCMM\_RMM\_3**

Тестируемые требования: REQ-UI-1, REQ\_RCMM\_4, REQ-RMM-3

Описание теста: Выбрав удаляемый маршрут, нажать на кнопку удаления

Ожидаемый результат: удаление выбранного маршрута

* **TEST\_UI\_RCMM\_RMM\_4**

Тестируемые требования: REQ-UI-1, REQ\_RCMM\_4, REQ-RMM-4

Описание теста: Выбрав обновляемый маршрут, нажать на кнопку изменения и записать корректный формат маршрута

Ожидаемый результат: изменения маршрута в коллекции и его последующее отображение в основном окне

* **TEST\_UI\_RCMM\_RMM\_5**

Тестируемые требования: REQ-UI-1, REQ\_RCMM\_4, REQ-RMM-4

Описание теста: Выбрав обновляемый маршрут, нажать на кнопку изменения и записать некорректный формат маршрута

Ожидаемый результат: предупреждение о неправильном вводе

# Тестирование проекта

## Подсистема FE-1

* **TEST\_UI**

Тестируемые требования: REQ-UI-1

Описание теста: выбор коллекции, нажав на одну из кнопок, представляющие их, в боковом меню пользовательского интерфейса

Ожидаемый результат: обновление основного окна, после которого в основном окне будут отображаться маршруты выбранной коллекции

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_1**

Тестируемые требования: REQ-UI-1

Описание теста: Нажать на кнопку добавлении коллекции

Ожидаемый результат: добавление нового элемента, представляющего добавляемую коллекцию, в боковом меню пользовательского интерфейса и добавление новой коллекции

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_2**

Тестируемые требования: REQ-UI-1

Описание теста: Нажать на кнопку удаления коллекции

Ожидаемый результат: удаление элемента, представляющего удаляемую коллекцию, в боковом меню пользовательского интерфейса и удаление коллекции

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_1**

Тестируемые требования: REQ-UI-1

Описание теста: Нажать на кнопку добавлении маршрута, после чего ввести корректный формат маршрута

Ожидаемый результат: добавление нового маршрута в отображаемой коллекции с последующим отображением в окне

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_2**

Тестируемые требования: REQ-UI-1

Описание теста: Нажать на кнопку добавлении маршрута, после чего ввести некорректный формат маршрута

Ожидаемый результат: предупреждение о неправильном вводе

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_3**

Тестируемые требования: REQ-UI-1

Описание теста: Выбрав удаляемый маршрут, нажать на кнопку удаления

Ожидаемый результат: Удаление элемента, представлявшего выбранный для удаления маршрут

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_4**

Тестируемые требования: REQ-UI-1

Описание теста: Выбрав обновляемый маршрут, нажать на кнопку изменения и записать корректный формат маршрута

Ожидаемый результат: Изменение отображаемых маршрутов в основном окне

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_5**

Тестируемые требования: REQ-UI-1

Описание теста: Выбрав обновляемый маршрут, нажать на кнопку изменения и записать некорректный формат маршрута

Ожидаемый результат: предупреждение о неправильном вводе

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

## Подсистема FE-2

* **TEST\_UI**

Тестируемые требования: REQ-RCMM-1

Описание теста: выбор коллекции, нажав на одну из кнопок, представляющие их, в боковом меню пользовательского интерфейса

Ожидаемый результат: обновление основного окна, после которого в основном окне будут отображаться маршруты выбранной коллекции

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_1**

Тестируемые требования: REQ-RCMM-2

Описание теста: Нажать на кнопку добавлении коллекции

Ожидаемый результат: добавление нового элемента, представляющего добавляемую коллекцию, в боковом меню пользовательского интерфейса и добавление новой коллекции

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_2**

Тестируемые требования: REQ-RCMM-3

Описание теста: Нажать на кнопку удаления коллекции

Ожидаемый результат: удаление элемента, представляющего удаляемую коллекцию, в боковом меню пользовательского интерфейса и удаление коллекции

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_1**

Тестируемые требования: REQ\_RCMM\_4

Описание теста: Нажать на кнопку добавлении маршрута, после чего ввести корректный формат маршрута

Ожидаемый результат: добавление нового маршрута в отображаемой коллекции с последующим отображением в окне

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_2**

Тестируемые требования: REQ\_RCMM\_4

Описание теста: Нажать на кнопку добавлении маршрута, после чего ввести некорректный формат маршрута

Ожидаемый результат: предупреждение о неправильном вводе

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_3**

Тестируемые требования: REQ\_RCMM\_4

Описание теста: Выбрав удаляемый маршрут, нажать на кнопку удаления

Ожидаемый результат: удаление выбранного маршрута

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_4**

Тестируемые требования: REQ\_RCMM\_4

Описание теста: Выбрав обновляемый маршрут, нажать на кнопку изменения и записать корректный формат маршрута

Ожидаемый результат: изменения маршрута в коллекции и его последующее отображение в основном окне

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_5**

Тестируемые требования: REQ\_RCMM\_4

Описание теста: Выбрав обновляемый маршрут, нажать на кнопку изменения и записать некорректный формат маршрута

Ожидаемый результат: предупреждение о неправильном вводе

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

## Подсистема FE-3

* **TEST\_UI**

Тестируемые требования: REQ-RMM-1

Описание теста: выбор коллекции, нажав на одну из кнопок, представляющие их, в боковом меню пользовательского интерфейса

Ожидаемый результат: обновление основного окна, после которого в основном окне будут отображаться маршруты выбранной коллекции

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_1**

Тестируемые требования: REQ-RMM-2

Описание теста: Нажать на кнопку добавлении маршрута, после чего ввести корректный формат маршрута

Ожидаемый результат: добавление нового маршрута в отображаемой коллекции с последующим отображением в окне

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_2**

Тестируемые требования: REQ-RMM-2

Описание теста: Нажать на кнопку добавлении маршрута, после чего ввести некорректный формат маршрута

Ожидаемый результат: предупреждение о неправильном вводе

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_3**

Тестируемые требования: REQ-RMM-3

Описание теста: Выбрав удаляемый маршрут, нажать на кнопку удаления

Ожидаемый результат: удаление выбранного маршрута

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_4**

Тестируемые требования: REQ-RMM-4

Описание теста: Выбрав обновляемый маршрут, нажать на кнопку изменения и записать корректный формат маршрута

Ожидаемый результат: изменения маршрута в коллекции и его последующее отображение в основном окне

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

* **TEST\_UI\_RCMM\_RMM\_5**

Тестируемые требования: REQ-RMM-4

Описание теста: Выбрав обновляемый маршрут, нажать на кнопку изменения и записать некорректный формат маршрута

Ожидаемый результат: предупреждение о неправильном вводе

Видимый результат: Полностью совпадает с ожидаемым

Резюме: Тест пройден

# Заключение

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

* Разработан план проекта;
* Разработан регламент проведения инспекции;
* Разработана модель состояний задач;
* Разработана презентация проекта;
* Разработаны требования к проекту;
* Разработана архитектура проекта;
* Разработать измерения проекта;
* Разработан перечь задач проекта;
* Разработаны рекомендации по кодированию;
* Разработан план тестирования проекта;
* Проект протестирован.

Таким образом, цель данного курсового проекта была достигнута.

# Список литературы

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