  
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 КОНТРОЛЯ УЧЕТА ПОСЕЩЕНИЙ ДЛЯ ПРЕДПРИЯТИЯ**

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  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Алексейцева Н.Д.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Дондоков Д.С.

Руководитель:  
Старший преподаватель ПИиИИ  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю.С.

г. Владивосток  
2024

Оглавление

[Введение 3](#_Toc160564456)

[1 Разработка плана проекта 4](#_Toc160564457)

[2 Разработка регламента проведения инспекции 5](#_Toc160564458)

[3 Разработка модели состояний задач 8](#_Toc160564459)

[4 Разработка презентации проекта 10](#_Toc160564460)

[5 Разработка требований к проекту 13](#_Toc160564461)

[6 Разработка архитектуры проекта 15](#_Toc160564462)

[7 Разработка измерений проекта 16](#_Toc160564463)

[8 Разработка перечня задач проекта 18](#_Toc160564464)

[9 Разработка рекомендаций по кодированию 19](#_Toc160564465)

[10 Разработка плана тестирования проекта 21](#_Toc160564466)

[11 Тестирование проекта 24](#_Toc160564467)

[Заключение 29](#_Toc160564468)

**Введение**

Промышленная разработка информационных систем включает в себя множество этапов, начиная от разработки плана проекта, заканчивая тестированием проекта для чего, очевидно 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 «Система контроля учета посещений для предприятия» и составление технической документации к данному средству.

Таким образом, целью курсовой работы является разработка программного средства «Система контроля учета посещений для предприятия» с использованием подходов коллективной промышленной разработки.

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 и 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 текущем проекте исполнителями являются следующие лица:

* Full-stack developer – Алексейцева Н.Д.
* Technical Writer – Дондоков Д.С.

Был разработан перечень задач для выполнения и примерные сроки их реализации (*Рисунок 1*).

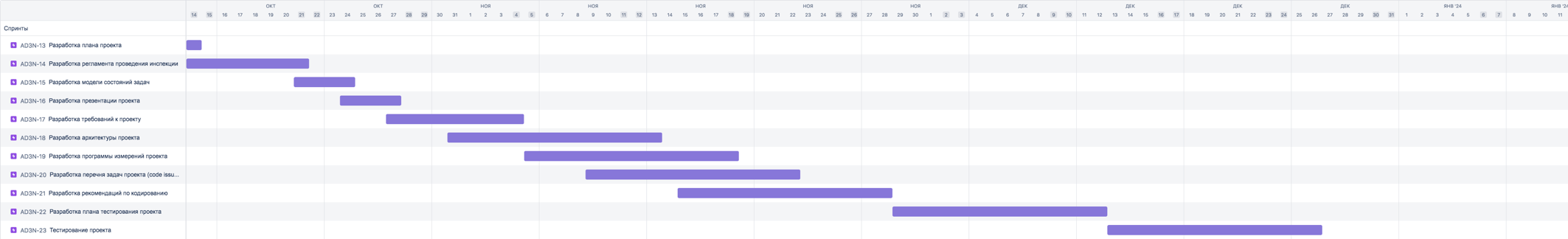


Рисунок 1 — План проекта

# **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 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 – предварительное ознакомление с продуктом.
3. Проведение – анализ изменений и внесение замечаний инспектором.
4. 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 ВК сообщение с упоминанием инспектора.

Столкнувшись с изменением дизайна проекта, инспектор обращается к председателю инспекции в рабочем чате в ВК. 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. Ошибка – замечание, сообщающее о необходимости исправления.
2. Комментарий – рекомендация по улучшению продукта, не требующая обязательных изменений.
3. Замечание для исследования — проблема, природа которой не может быть определена на собрании и требует доп. исследований.

## **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**

1. New – новые задачи.
2. In progress –задачи в процессе выполнения.
3. Testing – задачи в состоянии проверки работоспособности изменений.
4. To review — задачи в состоянии проверки изменений на соответствие требованиям проекта.
5. Done – выполненные задачи.

## **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**

В качестве системы отслеживания задач используется Jira.

Созданная задача имеет состояние «New». Team Leader может назначать на задачу исполнителя и инспектора, также участник команды может взять роль исполнителя задачи на себя.

Как только участник команды разработки готов приступить к задаче, на которую он назначен, он переводит её в состояние «In progress». На этом этапе исполнитель выполняет задачу.

Когда задача выполнена, она переходит в состояние «Testing». На этом этапе исполнитель проверяет работоспособность изменений на наборе тестов.

По завершении тестирования, задача переходит в состояние «To review». На этом этапе проходит инспекция изменений. В случае необходимости внесения изменений, состояния задачи меняется на «In progress», иначе на «Done».

# **Разработка презентации проекта**

Была разработана презентация проекта, состоящая из шести слайдов:

1. Титульный слайд (Рисунок 2);
2. Почему именно мы (Рисунок 3);
3. Основные возможности (Рисунок 4);
4. Проект интерфейса (Рисунок 5);
5. Подводим итоги (Рисунок 6);
6. В ближайших релизах (Рисунок 7).



Рисунок 2 — Титульный слайд



Рисунок 3 — Почему именно мы

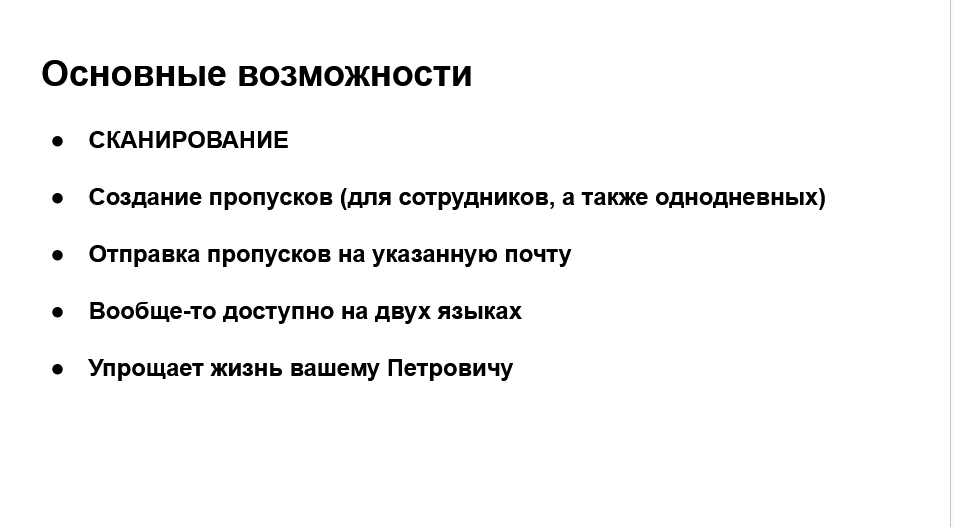


Рисунок 4 — Основные возможности



Рисунок 5 — Проект интерфейса

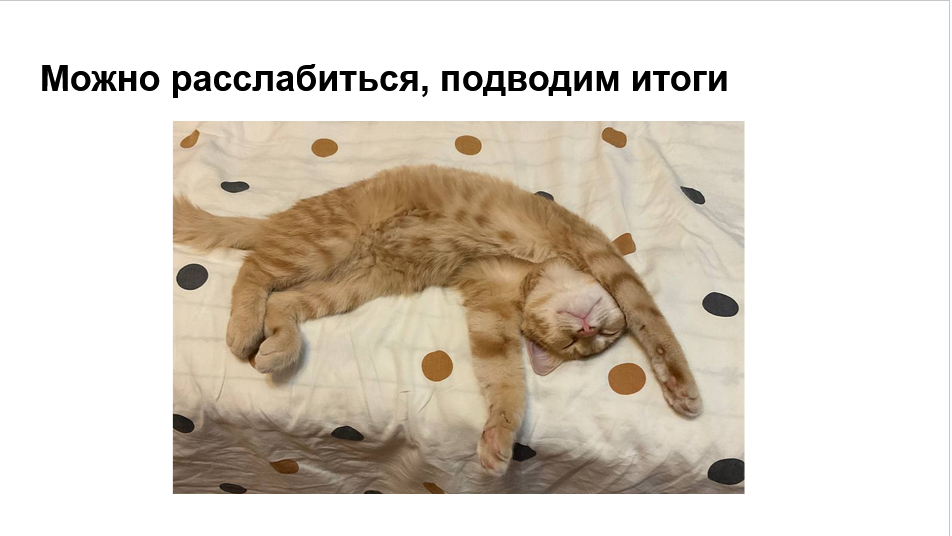


Рисунок 6 — Подводим итоги

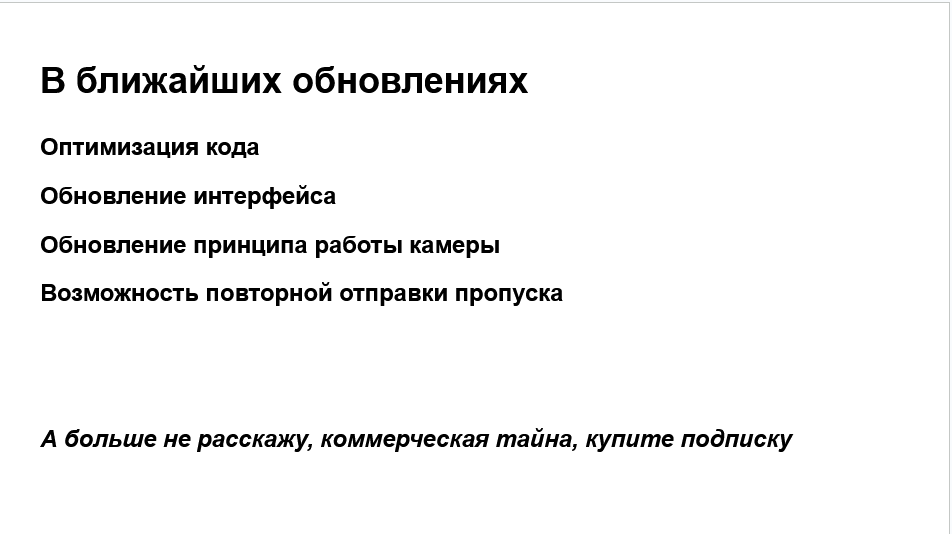


Рисунок 7 — В ближайших релизах

# **Разработка требований к проекту**

Программный продукт: Система контроля учета посещений для предприятия.

Программный продукт состоит из следующих подсистем:

1. Пользовательский интерфейс;
2. Модуль сканирования;
3. Модуль регистрации;
4. Модуль статуса персонала.

## **Общие требования**

### **Функциональные требования**

Требование FU\_001 Продукт должен позволять создавать пропуски на предприятия.

Требование FU\_002 Продукт должен формировать единственный QR-код для одного работника.

Требование FU\_003 Продукт должен формировать единственный QR-код для одного посетителя в день.

Требование FU\_004 Продукт должен позволять производить сканирование одного QR-кода единовременно.

Требование FU\_005 Продукт должен производить валидацию сканируемых QR-кодов.

### **Нефункциональные требования**

Требование UF\_001 Продукт должен быть написан на языке `Python` версии 3.12.

Требование UF\_003 Продукт должен иметь графический интерфейс.

Требование UF\_004 Контроль версий продукта должен вестись в GitHub.

### **Системные требования**

Требование SR\_001 Продукт должен запускаться на компьютере с операционными системами Windows, macOS.

Требование SR\_002 Для полноценной работы продукта необходимо наличие сканера или камеры.

## **Требования к подсистеме «Сканирование»**

Требование SC\_001 Высокая точность распознавания.

Требование SC\_002 Валидация QR-кодов.

## **Требования к подсистеме «Создание пропуска»**

Требование RG\_001 Вывод ошибок при незаполненности полей.

Требование RG\_002 Отправка QR-кода на указанную почту.

Требование RG\_003 Невозможность повторной регистрации по одной и той же почте.

Требование RG\_004 Возможность регистрации разных типов пользователей.

## **Требования к подсистеме «Статус на объекте»**

Требование SP\_001 Должно быть разделение на сотрудников и посетителей.

# **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.

Была разработана Архитектурно-контекстная диаграмма данных между подсистемами разрабатываемого транслятора (Рисунок 9).

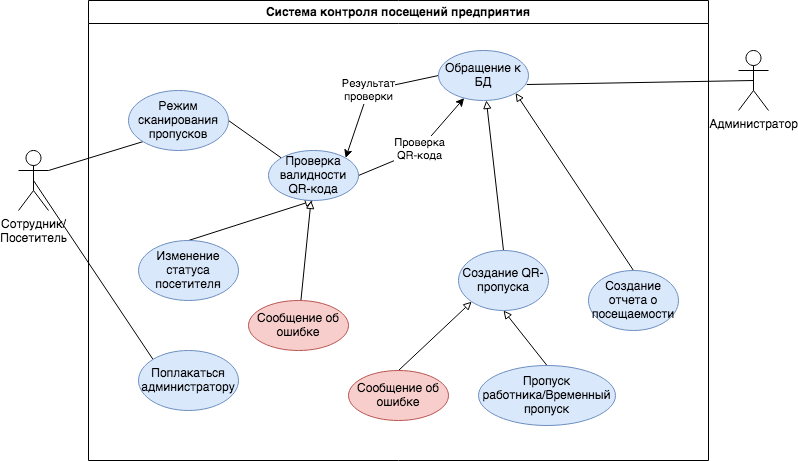


Рисунок 9 — Архитектурно-контекстная диаграмма данных

# **Разработка измерений проекта**

Контроль за 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 компании, называется программой измерений компании. Программа измерений выполняется как в рамках отдельных проектов, осуществляемых компанией, так и в рамках определённых видов деятельности компании. На крупных предприятиях программа измерений осуществляется специальным отделом по обеспечению качества (SQA — Software Quality Assurance team).

## **Метрики эффективности процесса разработки**

1. Problem Resolution Rate (PRR)

PRR = Количество дней на обработку задачи

Стратегическая цель метрики – сократить сроки разработки модулей ПО.

Изучаемый объект метрики – задача.

Измеряемый атрибут – время обработки.

Единица измерения – день.

1. Faults Screening (FS)

FS = (Общее количество ошибок − Число ошибок, найденных до бета-тестирования) × 100% / Общее количество ошибок

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – проект.

Измеряемый атрибут – эффективность обнаружения дефектов.

Единица измерения – %.

## **Метрики качества программного продукта**

Beta Testing Faults (BTF)

BTF = Число обнаруженных на этапе бета-тестирования ошибок / LOC

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – продукт.

Измеряемый атрибут – плотность неполадок.

Единица измерения – неполадка / LOC.

# **Разработка перечня задач проекта**

## **Общие задачи**

[FU-1] Разработать модуль создания QR-кодов.

[FU-2] Разработать модуль сканирования QR-кодов.

[FU-3] Разработать модуль валидации сканируемых QR-кодов.

## **Задачи для подсистемы «Пользовательский интерфейс»**

[UI-1] Добавить основные элементы интерфейса.

## **Задачи для подсистемы «Сканирование»**

[SC-1] Обеспечить высокую точность распознавания.

[SC-2] Обеспечить вывод ошибок при распознавании некорректного QR-кода.

## **Задачи для подсистемы «Создание пропуска»**

[RG-1] Обеспечить вывод ошибок при не заполненных полей.

[RG-2] Обеспечить невозможность повторной регистрации по одной и той же почте.

[RG-3] Обеспечить хранение QR-кодов на локальном сервере.

## **Задачи для подсистемы «Статус на объекте»**

[SP-1] Обеспечить разделение посетителей на типы: сотрудники и посетители.

# **Разработка рекомендаций по кодированию**

В основе рекомендаций – руководство по оформлению кода на Python PEP-8. Все правила PEP-8 должны соблюдаться в проекте. Ниже перечислены самые важные рекомендации и требования по оформлению кода.

## **Рекомендации и требования к оформлению кода**

* snake\_case для всех имён, кроме констант и классов.

your\_name = my\_var

get\_name()

init.py

my\_app.py

* SCREAMING\_SNAKE\_CASE для констант.

PI = 3.1415

MY\_PI = 3.14

* UpperCamelCase для имён классов.

class MyClass

class DataFrame

* Во всех именах ключевое слово должно быть существительным, за исключением имён функций и методов.

name

words

best\_sum

* Имена функций и методов должны начинаться с глагола.

get\_name()

set\_name()

sort()

is\_number()

* Функции отделяются друг от друга двумя пустыми строками.

def get\_name():

pass

def set\_name():

pass

* Методы отделяются друг от друга одной пустой строкой.

class MyClass:

def get\_name():

pass

def set\_name():

pass

* В выражениях не должны участвовать неименованные константы (magic numbers).

side\_1 = 5

side\_2 = 20

square = side\_1 \* side\_2

## **Чек-лист для проверки корректности кода**

snake\_case для всех имён, кроме констант и классов.

SCREAMING\_SNAKE\_CASE для констант.

UpperCamelCase для имён классов.

Во всех именах ключевое слово – существительное, кроме функций и методов.

Имена функций и методов начинаются с глагола.

Функции отделены друг от друга двумя пустыми строками.

Методы отделены друг от друга одной пустой строкой.

# **Разработка плана тестирования проекта**

## **Тесты для общих задач**

Тест TEST\_FU\_001

Тестируемые требования: FU-1

В модуле создания QR-кодов данные должны быть получены из графической формы «Создание пропуска». Полученные данные должны быть проверены на отсутствие нарушения уникальности и создании соответствующего пропуска. В случае нарушения выводится сообщение об ошибке.

Тест TEST\_FU\_002

Тестируемые требования: FU-2

В модуле сканирования QR-кодов данные должны быть получены из сканирования пропуска. Полученные данные передаются в модуль валидации сканируемых QR-кодов.

Тест TEST\_FU\_003

Тестируемые требования: FU-3

В модуле валидации QR-кодов данные должны быть получены из модуля сканирования QR-кодов. Полученные данные должны быть проверены на совпадение в базе данных и принятия решения о допуске на объект, в случае не валидности пропуска выводится сообщение об ошибке.

## **Тесты для подсистемы «Сканирование»**

Тест TEST\_SC\_001

Тестируемые требования: SC-1

Модуль сканирования программного средства должен обеспечить высокую точность распознавания пропусков.

Тест TEST\_SC\_002

Тестируемые требования: SC-2

Модуль сканирования должен выводить сообщение об ошибке при сканировании некорректного QR-кода.

## **Тесты для подсистемы «Создание пропуска»**

Тест TEST\_RG\_001

Тестируемые требования: RG-1

Модуль регистрации должен выводить сообщение об ошибке при наличии не заполненных полей для ввода.

Тест TEST\_RG\_002

Тестируемые требования: RG-2

Модуль регистрации должен выводить сообщение об ошибке при попытке повторной регистрации по одной и той же почте.

Тест TEST\_RG\_003

Тестируемые требования: RG-3

Модуль регистрации должен сохранять локально копии созданных QR-кодов.

## **Тесты для подсистемы «Статус на объекте»**

Тест TEST\_SP\_001

Тестируемые требования: SP\_001

Модуль «Статус на объекте» должен позволять однозначно интерпретировать тип посетителя.

## **Матрица покрытия тестами требований**

Была построена матрица покрытия тестами требований (Рисунок 10).

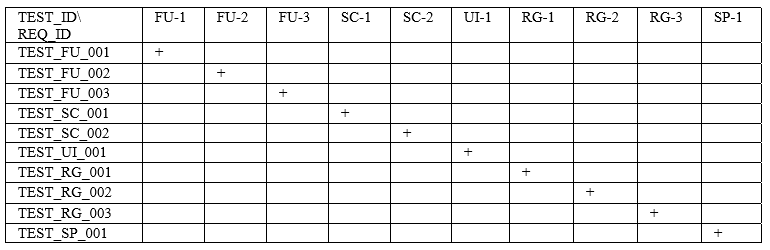


Рисунок 10 — Матрица покрытия тестами требований

# **Тестирование проекта**

## **Тесты для общих задач**

Тест TEST\_FU\_001\_1

Тестируемые требования: FU\_001, UI\_001

На главном экране приложения должны быть следующие элементы интерфейса:

* Логотип приложения.
* Кнопки меню «Сканирование», «Статус на объекте», «Создание».

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Все элементы интерфейса присутствуют.

Видимый результат: Все элементы интерфейса присутствуют.

Резюме: Тест пройден

Тест TEST\_FU\_001\_2

Тестируемые требования: FU\_001

На экране «Сканирование» должны быть следующие элементы интерфейса:

* Значок сканирования.
* Функциональные кнопки «Сканировать» и «Назад».

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Все элементы интерфейса присутствуют.

Видимый результат: Все элементы интерфейса присутствуют.

Резюме: Тест пройден

Тест TEST\_FU\_001\_3

Тестируемые требования: FU\_001

На экране «Статус на объекте» должны быть следующие элементы интерфейса:

* Вкладки «Присутствуют» и «Отсутствуют» для определения статуса человека.
* Таблицы с данными о людях на объекте.
* Функциональная кнопка «Назад».

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Все элементы интерфейса присутствуют.

Видимый результат: Все элементы интерфейса присутствуют.

Резюме: Тест пройден

Тест TEST\_FU\_001\_4

Тестируемые требования: FU\_001

На экране регистрации должны быть следующие элементы интерфейса:

* Поля с соответствующими заголовками-описаниями, минимальный набор полей для сотрудника: Фамилия, Имя, Отчество, Телефон, E-mail; переключатель: Является ли посетителем.
* Функциональные кнопки «Отменить» и «Зарегистрировать».

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Все элементы интерфейса присутствуют.

Видимый результат: Все элементы интерфейса присутствуют.

Резюме: Тест пройден

Тест TEST\_FU\_002

Тестируемые требования: FU\_002

В случае сканирования валидного QR-кода должно быть отображено соответствующее сообщение.

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Сообщение об успешном сканировании.

Видимый результат: Сообщение об успешном сканировании.

Резюме: Тест пройден

Тест TEST\_FU\_003

Тестируемые требования: FU\_003

В случае сканирования не валидного QR-кода должно быть отображено соответствующее сообщение.

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Сообщение о неудачном сканировании, с описанием ошибки.

Видимый результат: Сообщение о неудачном сканировании, с описанием ошибки.

Резюме: Тест пройден.

## **Тесты для подсистемы «Сканирование»**

Тест TEST\_SC\_001

Тестируемые требования: SC\_001

Провести замер соотношения выявленных QR-кодов к невозможности определить пропуск при помощи камеры за 30 секунд.

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Определение QR-кода успешно не менее чем в 2/3 (~66%) раз от общего количества замеров.

Видимый результат: Определение QR-кода успешно в 35/46 (~76%) раз.

Резюме: Тест пройден.

Тест TEST\_SC\_002

Тестируемые требования: SC\_002

Модуль сканирования должен выводить сообщение об ошибке при сканировании некорректного QR-кода.

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Сообщение об ошибке при сканировании некорректного QR-кода.

Видимый результат: Сообщение об ошибке при сканировании некорректного QR-кода.

Резюме: Тест пройден.

## **Тесты для подсистемы «Регистрация»**

Тест TEST\_RG\_001

Тестируемые требования: RG-1

Тестируемая версия продукта: 1.0.1

Модуль регистрации должен выводить сообщение об ошибке при наличии не заполненных полей для ввода.

Ожидаемый результат:

Видимый результат:

Резюме: Тест пройден.

Тест TEST\_RG\_002

Тестируемые требования: RG-2

Тестируемая версия продукта: 1.0.1

Вывод сообщения об ошибке при попытке повторной регистрации по одной и той же почте.

Ожидаемый результат: Вывод сообщения об ошибке.

Видимый результат: Вывод сообщения об ошибке.

Резюме: Тест пройден.

Тест TEST\_RG\_003

Тестируемые требования: RG-3

Тестируемая версия продукта: 1.0.1

Сохранение локальных копий созданных QR-кодов

Ожидаемый результат: Создание локальной копии созданных QR-кодов.

Видимый результат: Создание локальной копии созданных QR-кодов.

Резюме: Тест пройден.

## **Тесты для подсистемы «Статус на объекте»**

Тест TEST\_SY\_001

Тестируемые требования: SY\_001

Корректное определения типа посетителя при сканировании пропуска.

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Корректное определения типа посетителя при сканировании пропуска.

Видимый результат: Корректное определения типа посетителя при сканировании пропуска.

Резюме: Тест пройден.

# **Заключение**

По окончанию всех работ посчитаем метрики, описанные в 7 главе:

1. Количество дней на обработку задачи (PRR) составило 7.5 дней.
2. Эффективность обнаружения дефектов (FS) составила 78.947%
3. Плотность неполадок (BTF) составила 0.00434 неполадок / LOC.

В рамках курсовой работы было разработано программное средство «Система контроля учета посещений для предприятия» 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.