  
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:  
Ассистент департамента ПИиИИ  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю.С.

г. Владивосток  
2024

Оглавление

[Введение 3](#_Toc156641398)

[1 Разработка плана проекта 4](#_Toc156641399)

[2 Разработка регламента проведения инспекции 5](#_Toc156641400)

[3 Разработка модели состояний задач 8](#_Toc156641401)

[4 Разработка презентации проекта 9](#_Toc156641402)

[5 Разработка требований к проекту 12](#_Toc156641403)

[6 Разработка архитектуры проекта 14](#_Toc156641404)

[7 Разработка измерений проекта 15](#_Toc156641405)

[8 Разработка перечня задач проекта 17](#_Toc156641406)

[9 Разработка рекомендаций по кодированию 19](#_Toc156641407)

[10 Разработка плана тестирования проекта 21](#_Toc156641408)

[11 Тестирование проекта 23](#_Toc156641409)

[Заключение 25](#_Toc156641410)

**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.

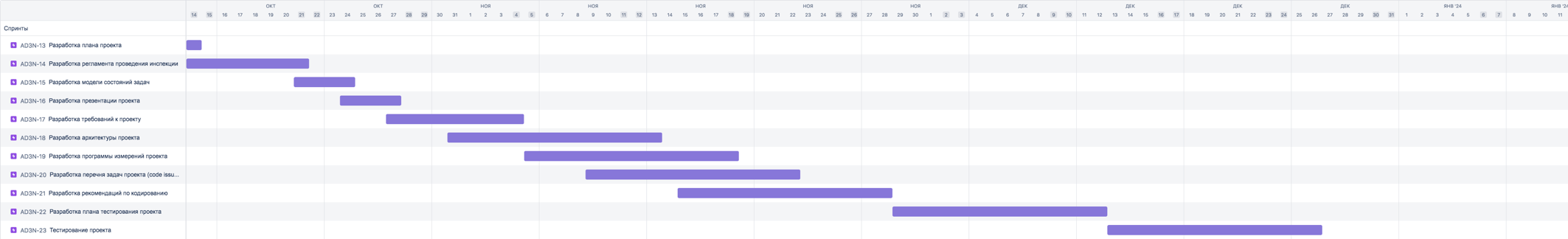
# **Разработка плана проекта**

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

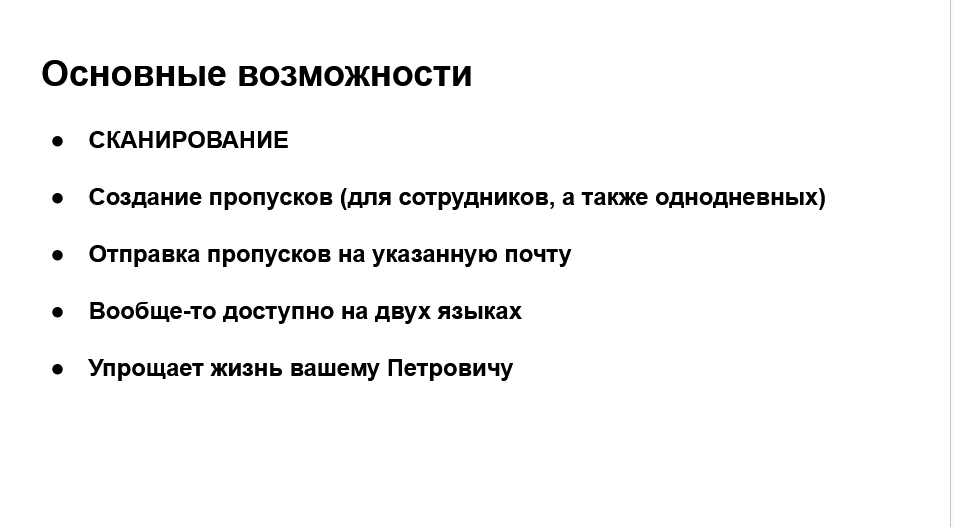


Рисунок 4 — Основные возможности



Рисунок 5 — Проект интерфейса

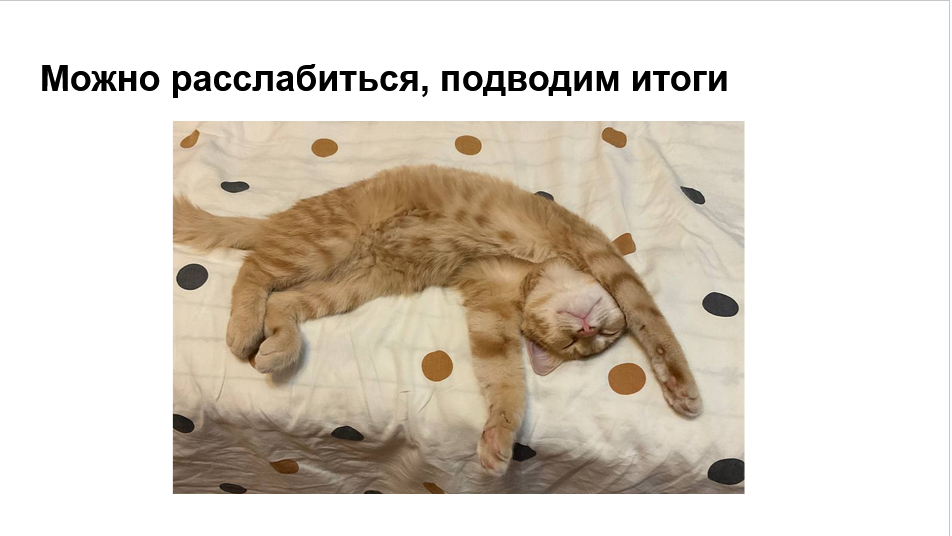


Рисунок 6 — Подводим итоги

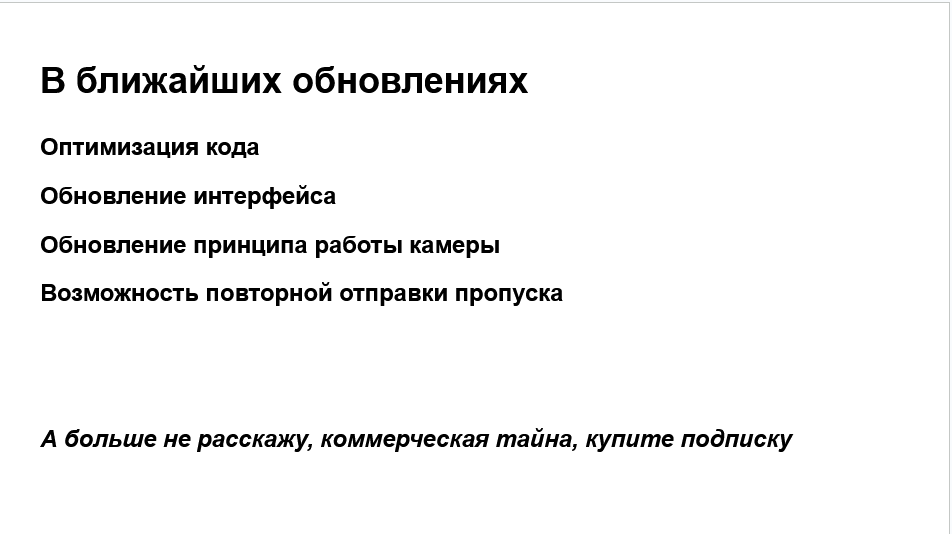


Рисунок 7 — В ближайших релизах

# **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 `3.10`.

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 «Регистрация»**

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.

## **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].

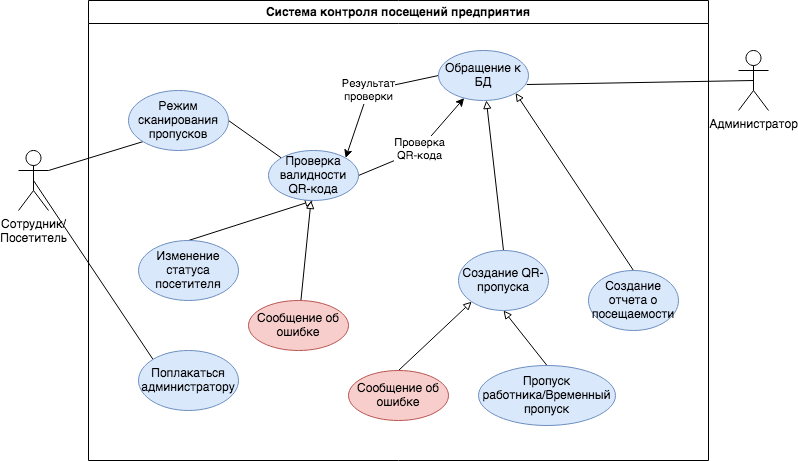


Рисунок 9 — Архитектурно-контекстная диаграмма данных

# **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.

2. 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**

1. 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**

[UF-1] Придумать как разнести по файликам модули

Создать модули для подсистем.

[UF-2] Подготовить пайплайн для тестов

Подготовить файлы с кодом на языке Java, скрипты для автоматического запуска тестов программной системы на них.

## **Задачи для подсистемы «Cканирование»**

[UI-1] Добавить основные элементы интерфейса

Необходимо добавить на сайт поля блоки с кодом, инструменты загрузки файла и кнопку начала трансляции (см. UI\_001).

[UI-2] Вывод однородных ошибок

Необходимо собрать ошибки всех модулей и выводить их на экран пользователя в одном формате в случае их возникновения (см. UI\_002). Сообщение об ошибке должно содержать в себе её примерное местоположение (см. UI\_003).

[UI-3] Загрузка файла для трансляции

Код на Java должен быть закружен в систему в виде файла, в интерфейсе нужен соответствующий функционал (см. UI\_004).

## **Задачи для подсистемы «Регистрация»**

[LX-1] Разработать начальную версию

Вход: любой код на языке Java.

Выход: JSON-файл с токенами (LX\_001, LX\_002, LX\_003).

[LX-2] Обеспечить вывод человекопонятных ошибок

Реализация требования LX\_004.

[LX-3] Написать и проверить тесты для лексера

Подготовить начальный набор тестов для модуля.

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

[SY-1] Разработать начальную версию

Вход: JSON-файл с токенами (SY\_001).

Выход: дерево разбора.

[SY-2] Обеспечить вывод человекопонятных ошибок

Реализация требования SY\_002.

[SY-3] Написать и проверить тесты для синтаксера

Подготовить начальный набор тестов для модуля.

# **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**

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

your\_name = my\_var

get\_name()

init.py

my\_app.py

1. SCREAMING\_SNAKE\_CASE для констант.

PI = 3.1415

MY\_PI = 3.14

1. UpperCamelCase для имён классов.

class MyClass

class DataFrame

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

name

words

best\_sum

1. Имена функций и методов должны начинаться с глагола.

get\_name()

set\_name()

sort()

is\_number()

1. Функции отделяются друг от друга двумя пустыми строками.

def get\_name():

pass

def set\_name():

pass

1. Методы отделяются друг от друга одной пустой строкой.

class MyClass:

def get\_name():

pass

def set\_name():

pass

1. 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 подсистемы «Сканирование»**

**Тест TEST\_UI\_001**

Тестируемые требования: UI\_001

**Тест TEST\_UI\_002**

Тестируемые требования: UI\_002, UI\_003

Сообщение об ошибке при вводе некорректного файла.

**Тест TEST\_UI\_003**

Тестируемые требования: UI\_004

Сайт должен принимать на вход файл только с расширением .java

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

**Тест TEST\_LX\_001**

Тестируемые требования: LX\_001, LX\_002

Проверка разбиения кода Java на токены, содержащие всю описанную в требованиях информацию, и их соответствие формату JSON. На вход модулю подаётся строку, содержащую код на языке java. Производится попытка разбиения на токены и сохранение их в формате JSON.

**Тест TEST\_LX\_002**

Тестируемые требования: LX\_003, LX\_004

Проверка работоспособности в случае нахождения лексемы, не попадающей в список выделенных, токен помечается ошибкой. При этом модуль не прекращает работу и формирует сообщение с описанием ошибки и доставляет его до интерфейса пользователя.

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

**Тест TEST\_SY\_001**

Тестируемые требования: SY\_001

Проверка работоспособности подсистемы на работу с основными арифметическими операциями. На вход подаётся файл с кодом, в котором создаются две переменные типа int: а = 5 и b = 2. Далее они используются в формуле (3 + (a - 1)) \* b / 2.

**Тест TEST\_SY\_002**

Тестируемые требования: SY\_001

Проверка работоспособности подсистемы на работу с несколькими пользовательскими методами. На вход подаётся файл с кодом, в котором объявлены два метода: до метода main и после него.

**Тест TEST\_SY\_003**

Тестируемые требования: SY\_002

Проверка корректности сообщения при ошибке с недостающей точкой с запятой. На вход подаётся файл, в котором не хватает точки с запятой на 6 строке.

**Тест TEST\_SY\_004**

Тестируемые требования: SY\_002

Проверка корректности сообщения при ошибке с отсутствующей в грамматике конструкцией языка. На вход подаётся файл, в котором не хватает скобок в блоке условного оператора в 4 строке.

## **Матрица покрытия тестами требований**

Была построена матрица покрытия тестами требований [Рисунок 10].

Рисунок 10 — Матрица покрытия тестами требований

# **Тестирование проекта**

## **Тесты для подсистемы «Скранирование»**

**Тест TEST\_UI\_001**

Тестируемые требования: UI\_001

На сайте должны быть следующие элементы интерфейса:

* кнопка добавления файла с исходным кодом на Java,
* поле для отображения исходного кода,
* поле для отображения транслированного кода на C++,
* кнопка отправки запроса на перевод.

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Все элементы интерфейса есть на сайте

Видимый результат: Все элементы интерфейса есть на сайте

Резюме: Тест пройден

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

**Тест TEST\_LX\_001**

Тестируемые требования: LX\_001, LX\_002

Проверка разбиения кода Java на токены, содержащие всю описанную в требованиях информацию, и их соответствие формату JSON. На вход модулю подаётся строку, содержащую код на языке java. Производится попытка разбиения на токены и сохранение их в формате JSON.

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Файл с массивом токенов формата JSON

Видимый результат: Файл с массивом токенов формата JSON

Резюме: Тест пройден

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

**Тест TEST\_SY\_001**

Тестируемые требования: SY\_001

Проверка работоспособности подсистемы на работу с основными арифметическими операциями. На вход подаётся файл с кодом, в котором создаются две переменные типа int: а = 5 и b = 2. Далее они используются в формуле (3 + (a - 1)) \* b / 2.

Тестируемая версия продукта: 1.0.1

Ожидаемый результат: Дерево разбора

Видимый результат: Дерево разбора

Резюме: Тест пройден

# **Заключение**

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

1. Количество дней на обработку задачи (PRR) составило 14.533 дней.
2. Эффективность обнаружения дефектов (FS) составила 82.352%
3. Плотность неполадок (BTF) составила 0.0026 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.