

# Test Plan

## (План тестирования)

### 1. Introduction (Введение)

*[Введение представляет собой обзор на весь документ в целом и включает в себя следующие разделы - назначение, область применения, определения и аббревиатуры, ссылки и обзор.]*

#### 1.1 Purpose

*[Укажите назначение данного документа.]*

Этот документ описывает стратегию тестирования для приложения ИСУВИ. Целью тестирования является обеспечение корректной работы всех микросервисов, пользовательского интерфейса (UI), проверка производительности, нагрузочного и стрессового тестирования, а также обеспечение безопасности и устойчивости системы.

#### 1.2 Scope (Область применения)

*[Приведите краткое описание области применения данного документа, к какому(им) проекту(ам) он относится, кем будет использоваться и т.д.]*

Тестирование охватывает следующие аспекты:

1. Функциональность: Проверка выполнения ключевых сценариев (создание катаклизмов, назначение заданий, мониторинг нагрузки).
2. Нагрузочные тесты: Работа системы под высокой нагрузкой.
3. Безопасность: Тестирование прав доступа и шифрования данных.
4. Интеграция: Проверка взаимодействия микросервисов.

#### 1.3 Intended Audience (Предполагаемая аудитория)

*[Укажите, для кого написан данный документ и в каких целях он будет использоваться.]*

- Команда разработчиков
- Менеджеры проекта
- Тестировщики





#### 1.4 Document Terminology and Acronyms (Терминология документа)

*[Укажите значение терминов и аббревиатур, которые употребляются в данном документе. Возможно указание ссылки на Глоссарий проекта.]*

Описаны в  Gloss .

#### 1.5 References (Ссылки)

*[Перечислите списком названия документов, на которые ссылаетесь в данном, укажите их источники.]*

-  T3
-  SRS
-  SAD.docx
-  UC

#### 1.6 Document Structure (Структура документа)

*[Приведите краткое описание остальных разделов документа.]*

Секция	Описание
2	Цель и мотивы тестирования.
3	Целевые объекты тестирования.
4	План тестов.
5	Подход к тестированию.
6	Критерии старта и окончания.
7	Ожидаемые результаты тестирования.
8	Требования к окружению.
9	Роли и обязанности.
10	Управление процессом тестирования.

## 2. Evaluation Mission and Test Motivation (Цель и мотивы тестирования)

### 2.1 Background (Справочная информация)

*[В данном разделе кратко опишите проект, какие цели он преследует, как будет использоваться, какова его архитектура. Уместны ссылки на другие документы.]*

ИСУВИ — это распределённая система, построенная на микросервисной архитектуре. Она управляет временными изменениями в мультивселенной, позволяя создавать катаклизмы, назначать задания и мониторить состояние станка.

### 2.2 Evaluation Mission (Цели тестирования)

*[В данном разделе укажите, с какой целью проводится тестирование проекта. Например, удовлетворить заказчика, найти как можно больше ошибок до окончательного завершения разработки, выявить самые главные проблемы системы и т.д.]*

1. Убедиться, что функциональность системы соответствует требованиям.
2. Проверить надежность работы всех микросервисов.
3. Гарантировать доступность системы под высокой нагрузкой.
4. Убедиться в безопасности данных пользователей.
5. Проверить работу интерфейса для разных ролей.

### 2.3 Test Motivators (Мотивы (?) тестирования)

*[Укажите, какие элементы будут служить источником информации для тестирования - функциональные и нефункциональные требования, ограничения системы, риски и т.д.]*

#### Функциональные требования

##### 1. Создание катаклизмов

- Вариант может создавать катаклизмы с указанием описания, приоритета и времени.
- Проверка:

- Валидация входных данных.
  - Успешное сохранение катаклизма в базе данных.
  - Отображение созданных катаклизмов в интерфейсе.
2. **Назначение заданий**
- Менеджер должен иметь возможность назначать задания агентам и помощникам.
  - Проверка:
    - Корректное отображение списка доступных заданий.
    - Отправка уведомлений назначенным агентам.
    - Изменение статуса заданий в реальном времени.
3. **Мониторинг нагрузки станка**
- Управляющий станком должен получать актуальные данные о текущей нагрузке.
  - Проверка:
    - Быстрая загрузка графика нагрузки (<2 секунд).
    - Отображение предупреждений о перегрузке.
4. **Управление пользователями**
- Администратор должен создавать, удалять и обновлять учетные записи пользователей.
  - Проверка:
    - Валидация уникальности имени пользователя.
    - Ограничение доступа в зависимости от ролей.
5. **Интеграция микросервисов**
- Все микросервисы должны корректно взаимодействовать через API.
  - Проверка:
    - Ответы API содержат данные в правильном формате (JSON).
    - Обработка ошибок на уровне API.

## **Нефункциональные требования**

1. **Производительность**
- Время отклика API должно быть  $\leq 3$  секунд.
  - Система должна обрабатывать до 100 одновременных пользователей без деградации производительности.
2. **Масштабируемость**
- Возможность добавления новых микросервисов без нарушения существующей архитектуры.
3. **Безопасность**
- Роль-ориентированный доступ к функциям системы.
  - Защита от SQL-инъекций и XSS-атак.

## **Риски**

1. **Интеграция микросервисов**
- Потенциальные проблемы взаимодействия между микросервисами из-за различий в форматах данных.
  - Возможные таймауты запросов между сервисами.
2. **Нагрузочные проблемы**

- Замедление обработки запросов из-за высокой активности пользователей.
- 3. **Ошибка в распределении ролей**
  - Неверно назначенные права пользователей могут привести к утечке данных или сбоям в выполнении задач.
- 4. **Сбой в хранилище данных**
  - Возможные ошибки при записи данных в базу (например, дублирование записей).
  - Отказ базы данных при большом количестве запросов.
- 5. **UI проблемы**
  - Ошибки в работе динамических элементов UI (формы).
- 6. **Производительность**
  - Снижение скорости работы системы при большом объеме данных (например, сотни активных катаклизмов).

### 3. Target Test Items (Целевые объекты тестирования)

*[Перечислите объекты тестирования, т.е. что именно будет проверяться с помощью тестов.]*

Целевыми объектами тестирования для системы ИСУВИ являются ключевые функциональные модули, интерфейсы и взаимодействия между ними.

#### 3.1 Модули микросервисной архитектуры

##### Общие требования к тестированию микросервисов

Для всех сервисов, независимо от их функциональности, проверяются:

1. **Корректная обработка ошибок**
  - Обработка исключений, связанных с недоступностью других сервисов.
  - Обработка неверных входных данных.
2. **Производительность**
  - Минимальное время отклика API (не более 3 секунд).
  - Устойчивость под нагрузкой (до 100 одновременных пользователей).
3. **Целостность данных**
  - Корректная запись, обновление и удаление данных в базе.
4. **Интеграция с другими сервисами**
  - Корректность передачи данных между сервисами через API Gateway.
  - Совместимость форматов данных (JSON).
5. **Безопасность**
  - Проверка прав доступа к каждому запросу.
  - Защита от SQL-инъекций и XSS-атак.

##### Сервисы

1. **Auth-Service (Сервис авторизации)**

- **Функциональность**  
Обеспечивает аутентификацию пользователей через токены (JWT).
  - **Проверяется**
    - Генерация токенов при успешной авторизации.
    - Проверка валидности токенов.
    - Обработка истекших или недействительных токенов.
    - Безопасность хранения и передачи токенов.
- 2. User-Service (Сервис управления пользователями)**
- **Функциональность**  
Управление учетными записями пользователей и их ролями.
  - **Проверяется**
    - Создание, изменение и удаление учетных записей.
    - Назначение и обновление ролей.
    - Проверка безопасности данных пользователей.
    - Проверка доступа в зависимости от роли пользователя.
- 3. Cataclysm-Service (Сервис катаклизмов)**
- **Функциональность**  
Управляет созданием, обновлением и удалением катаклизмов.
  - **Проверяется**
    - Валидация входных данных (тип катаклизма, описание, время).
    - Корректная запись и обновление данных в базе.
    - Обработка ошибок (например, дублирование ID катаклизма).
    - Уведомление других сервисов о создании нового катаклизма.
- 4. Task-Service (Сервис заданий)**
- **Функциональность**  
Управляет созданием, назначением и завершением заданий.
  - **Проверяется**
    - Распределение задач между агентами.
    - Корректное отображение статусов задач в реальном времени.
    - Обработка ошибок (например, назначение задания на несуществующего пользователя).
- 5. Machine-Service (Сервис мониторинга станка)**
- **Функциональность**  
Обеспечивает данные о текущей нагрузке станка и его состоянии.
  - **Проверяется**
    - Время отклика API при запросах данных.
    - Точность данных о текущей нагрузке.
    - Уведомления о перегрузке.
- 6. Timeline-Service (Сервис временных линий)**
- **Функциональность**  
Управляет созданием, уничтожением и слиянием временных линий.
  - **Проверяется**
    - Создание и уничтожение временных линий.
    - Сбор ресурсов с временных линий.
    - Корректность слияния временных линий.
    - Сохранение истории операций над временными линиями.
- 7. API Gateway (Шлюз API)**

- **Функциональность:**  
Управляет маршрутизацией запросов между сервисами.
- **Проверяется**
  - Корректная маршрутизация запросов к нужным сервисам.
  - Обработка таймаутов и ошибок от микросервисов.
  - Производительность при большом количестве запросов.

## 3.2 Взаимодействие модулей

### 1. API-интеграции

- Проверяется:
  - Совместимость форматов данных (JSON).
  - Обработка ошибок при взаимодействии между сервисами.
  - Производительность при интеграции.

### 2. База данных (PostgreSQL)

- Проверяется:
  - Целостность данных.
  - Производительность при большом объеме данных.
  - Аудит операций (например, логирование изменений).

## 3.3 Пользовательский интерфейс (UI)

### 1. Основные экраны:

- Экран управления катаклизмами.
- Экран назначения заданий.
- График нагрузки станка.
- Управление учетными записями.

### 2. Проверяется:

- Корректное отображение данных.
- Динамические элементы (формы, уведомления).
- Доступность экранов.

## 3.4 Производительность и нагрузка

### 1. Производительность системы

- Проверяется:
  - Время отклика API.
  - Время загрузки интерфейсов.

### 2. Нагрузочное тестирование

- Максимальная нагрузка: 100 пользователей одновременно.
- Проверяется:
  - Корректность обработки запросов при пиковых нагрузках.
  - Производительность базы данных.

### 3. Стрессовое тестирование

- Проверяется:
  - Поведение системы при превышении допустимой нагрузки.
  - Устойчивость работы и восстановление после сбоя.

## 4. Outline of Planned Tests (План тестов)

*[В данном разделе перечислите все виды тестов, которые будут включены в процесс тестирования. Кратко опишите, что они будут проверять.]*

В данном разделе перечислены виды тестов, которые будут включены в процесс тестирования системы ИСУВИ. Для каждого типа тестирования указаны основные цели и проверяемые аспекты.

### 4.1 Unit Testing (Модульное тестирование)

- **Цель:** Проверка корректности работы отдельных модулей (методов, классов) каждого сервиса.
- **Область тестирования:**
  - Логика работы микросервисов (валидация данных, обработка ошибок).
- **Примеры тестов:**
  - Проверка валидации входных данных в Cataclysm-Service.
  - Тестирование работы Auth-Service, хранения и обработки данных

### 4.2 Integration Testing (Интеграционное тестирование)

- **Цель:** Проверка работы микро сервиса используя контроллеры
- **Область тестирования:**
  - Запущенный микро сервиса, который разворачивает для тестирования
  - Обработка ошибок при интеграции (например, недоступность одного из сервисов).
- **Примеры тестов:**
  - Создание катаклизма в Cataclysm-Service и вызов передачи данных в Task-Service.
  - Проверка корректности запросов в API Gateway.

### 4.3 Functional Testing (Функциональное тестирование)

- **Цель:** Проверка выполнения бизнес-логики системы согласно требованиям.
- **Область тестирования:**
  - Функционал всех ролей (Варианты, Менеджеры, Агенты).
  - Выполнение сценариев использования (создание катаклизмов, назначение заданий, мониторинг станка).
- **Примеры тестов:**
  - Создание временной линии и успешный сбор ресурсов в Timeline-Service.
  - Назначение задания и обновление статуса в Task-Service.

### 4.4 User Interface Testing (Тестирование пользовательского интерфейса)

- **Цель:** Проверка работоспособности интерфейса и корректного отображения данных.
- **Область тестирования:**
  - Динамические элементы интерфейса (формы, графики).
  - Адаптивность интерфейса на различных устройствах и браузерах.

- **Примеры тестов:**
  - Проверка корректности отображения графиков нагрузки станка в Machine-Service.
  - Проверка ввода данных в форме создания катаклизма.

#### 4.5 Performance Testing (Тестирование производительности)

- **Цель:** Проверка быстродействия системы и её компонентов.
- **Область тестирования:**
  - Время отклика API (не более 3 секунд).
  - Быстродействие базы данных при большом количестве запросов.
- **Примеры тестов:**
  - Время выполнения массовых операций (например, создание 100 заданий).
  - Производительность API Gateway при 100 одновременных запросах.

#### 4.6 Load Testing (Нагрузочное тестирование)

- **Цель:** Проверка работы системы под высокой нагрузкой.
- **Область тестирования:**
  - Корректность обработки 100 одновременных пользователей.
  - Работа базы данных и микросервисов под нагрузкой.
- **Примеры тестов:**
  - Создание 1000 катаклизмов в Cataclysm-Service за короткий промежуток времени.
  - Обработка запросов от всех ролей одновременно.

#### 4.7 Stress Testing (Стрессовое тестирование)

- **Цель:** Оценка устойчивости системы при экстремальных нагрузках.
- **Область тестирования:**
  - Поведение системы при превышении допустимой нагрузки.
  - Восстановление после сбоев.
- **Примеры тестов:**
  - Создание 10 000 заданий за короткий промежуток времени.
  - Искусственное отключение одного из микросервисов (например, Machine-Service).

#### 4.8 Security Testing (Тестирование безопасности)

- **Цель:** Обеспечение безопасности данных пользователей и защиты от атак.
- **Область тестирования:**
  - Проверка прав доступа для всех ролей.
  - Устойчивость к SQL-инъекциям и XSS-атакам.
- **Примеры тестов:**
  - Попытка доступа к данным катаклизмов с неправильной ролью.
  - Ввод некорректных данных в поля API.

#### 4.9 Failover and Recovery Testing (Тестирование отказоустойчивости)



- **Цель:** Проверка способности системы восстанавливаться после сбоев.
- **Область тестирования:**
  - Перезапуск системы после аварийного завершения работы.
  - Сохранность данных при сбоях.
- **Примеры тестов:**
  - Прерывание работы одного из микросервисов и проверка его восстановления.

## 5. Test Approach (Подход к тестированию)

*[Данный раздел представляет рекомендованные стратегии для разработки и выполнения обязательных тестов. Не все виды тестов обязательно должны быть реализованы.]*

### 5.1 Testing Techniques and Types (Техники тестирования)

*[Для описания каждой из используемых техник тестирования рекомендуется воспользоваться следующей таблицей:]*

Technique Objective: (Цель)	<i>[В чем состоит цель данного типа тестов, что он проверяет]</i>
Technique: (Описание процесса)	<i>[Пошаговое подробное описание процесса выполнения тестов]</i>
Oracles: (Источники)	<i>[На какой документ/элемент системы опираются тесты для проверки результата выполнения.]</i>
Required Tools: (Инструменты)	<i>[Инструменты, необходимые для проведения теста - сторонние программы, необходимое окружение пользователя и т.д.]</i>
Success Criteria: (Критерий успеха)	<i>[Опишите условия, при которых данный тип тестов считается пройденным.]</i>

5.2.1 *Data and Database Integrity Testing (Тестирование базы данных)*

5.2.2 *Function Testing (Функциональное тестирование)*

5.2.3 *Business Cycle Testing (Тестирование бизнес-цикла)*

5.2.4 *User Interface Testing (Тестирование интерфейса)*

5.2.5 *Performance Profiling (Тестирование производительности)*

5.2.6 *Load Testing (Нагрузочное тестирование)*

5.2.7 *Stress Testing (Стрессовое тестирование)*

5.2.8 *Volume Testing (Объемное тестирование)*

5.2.9 *Security and Access Control Testing (Тестирование безопасности и прав доступа)*

5.2.10 *Failover and Recovery Testing (Тестирование на отказ и восстановление)*

5.2.11 *Configuration Testing (Тестирование конфигурации)*

5.2.12 *Installation Testing (Тестирование процесса установки)*

В данном разделе описан процесс выполнения тестов для системы ИСУВИ, включая используемые техники, инструменты и критерии успеха.

### 5.2.1 Data and Database Integrity Testing (Тестирование базы данных)

Параметр	Описание
Цель	Проверка целостности данных в базе
Описание процесса	<ol style="list-style-type: none"><li>1. Определите основные сценарии для тестирования данных:<ol style="list-style-type: none"><li>1.1. Вставка корректных и некорректных данных.</li><li>1.2. Проверка работы ограничений (например, уникальность, связи FK).</li><li>1.3. Удаление данных с зависимостями.</li></ol></li><li>2. Напишите SQL-скрипты для каждого сценария. Примеры:<ol style="list-style-type: none"><li>2.1. Вставка записи с уже существующим уникальным идентификатором.</li><li>2.2. Удаление записи, от которой зависят другие данные.</li></ol></li><li>3. Запустите скрипты через pgAdmin, терминал или возможности IDEA.</li><li>4. Проверьте:<ol style="list-style-type: none"><li>4.1. Сообщения об ошибках в случае некорректных данных.</li><li>4.2. Сохранение корректных данных.</li></ol></li><li>5. Зафиксируйте результаты выполнения каждого скрипта в формате .txt.</li></ol>
Источники	SRS.
Инструменты	SQL-скрипты, pgAdmin.
Критерий успеха	Все данные успешно сохраняются, ограничения и связи в базе данных соблюдены.

### 5.2.2 Function Testing (Функциональное тестирование)

Параметр	Описание
Цель	Проверка корректности выполнения функциональных требований.
Описание процесса	<ol style="list-style-type: none"> <li>1. Определите основные функции микросервисов (например, создание пользователя, обновление данных).</li> <li>2. Используйте JUnit для создания автоматизированных тестов: <ol style="list-style-type: none"> <li>а. Напишите тесты для каждого эндпоинта API с использованием библиотеки RestAssured.</li> <li>б. Покройте позитивные и негативные сценарии (например, некорректные входные данные).</li> </ol> </li> <li>3. Запустите тесты через IntelliJ IDEA и проверьте результаты.</li> <li>4. Сформируйте отчёт о прохождении тестов в формате HTML с помощью jacoco</li> </ol>
Источники	SRS
Инструменты	JUnit
Критерий успеха	Все функциональные тесты успешно пройдены.

### 5.2.3 Business Cycle Testing (Тестирование бизнес-цикла)

Параметр	Описание
Цель	Проверка выполнения пользовательских сценариев
Описание процесса	<ol style="list-style-type: none"> <li>1. Определите основные бизнес-циклы (например, создание катаклизма, назначение заданий, завершение временной линии).</li> <li>2. Используйте <b>Selenium WebDriver</b>: <ol style="list-style-type: none"> <li>а. Напишите скрипты для автоматизации пользовательских действий в браузере (например, заполнение форм,</li> </ol> </li> </ol>

	отправка запросов). 3. Запустите тесты в <b>Google Chrome</b> или <b>Safari</b> . 4. Зафиксируйте результаты тестов в отчёте <b>Google Docs</b> .
Источники	Use Case, SRS.
Инструменты	Selenium
Критерий успеха	Все сценарии бизнес-процессов выполняются корректно.

#### 5.2.4 User Interface Testing (Тестирование интерфейса)

Параметр	Описание
Цель	Проверка корректности работы интерфейса и корректное отображение элементов.
Описание процесса	<ol style="list-style-type: none"> <li>Разработайте чек-лист для ручного тестирования UI. Пример: <ol style="list-style-type: none"> <li>Корректная навигация между страницами.</li> <li>Соответствие элементов интерфейса дизайн-макету.</li> <li>Соответствие дизайн-системе Material 3</li> </ol> </li> <li>Откройте приложение в <b>Google Chrome</b>.</li> <li>Последовательно выполняйте тесты из чек-листа.</li> <li>Зафиксируйте результаты в <a href="#">Google Sheets</a>.</li> </ol>
Источники	UX/UI макеты, SRS.
Инструменты	Ручное тестирование
Критерий успеха	Интерфейс корректно работает и реализует описанные в пункте 3.6.1.1 документа SRS требования.

#### 5.2.5 Performance Profiling (Тестирование производительности)

Параметр	Описание
Цель	Проверка быстродействия системы и её компонентов.

<p>Описание процесса</p>	<ol style="list-style-type: none"> <li>1. Откройте Apache JMeter и создайте новый тестовый план.</li> <li>2. Thread Group: <ol style="list-style-type: none"> <li>a. Потоки: 100.</li> <li>b. Ramp-Up Period: 10 секунд.</li> <li>c. Loop Count: бесконечный или нужное количество циклов.</li> </ol> </li> <li>3. HTTP Header Manager: <ol style="list-style-type: none"> <li>a. Добавьте заголовки, необходимые для запросов (например, Content-Type: application/json или Authorization: Bearer &lt;token&gt;).</li> </ol> </li> <li>4. HTTP Request: <ol style="list-style-type: none"> <li>a. Укажите URL API-эндпоинта, который необходимо протестировать.</li> <li>b. Настройте метод запроса (GET, POST и т.д.).</li> <li>c. Если нужен запрос с телом (например, POST), добавьте JSON или другой формат в поле Body Data.</li> </ol> </li> <li>5. Constant Throughput Timer: <ol style="list-style-type: none"> <li>a. Укажите желаемую скорость запросов: 100 RPS.</li> </ol> </li> <li>6. View Results Tree: <ol style="list-style-type: none"> <li>a. Позволяет отследить каждый запрос, его ответ и статус (успех или ошибка).</li> </ol> </li> <li>7. Aggregate Graph: <ol style="list-style-type: none"> <li>a. Для визуального отображения метрик производительности (время отклика, пропускная способность).</li> </ol> </li> <li>8. Aggregate Report: <ol style="list-style-type: none"> <li>a. Для сбора подробных данных о времени отклика, количестве успешных и неуспешных запросов.</li> </ol> </li> <li>9. Duration Assertion: <ol style="list-style-type: none"> <li>a. Установите ограничение времени отклика (например, <math>\leq 3</math> секунд).</li> </ol> </li> <li>10. Запустите тест.</li> <li>11. Соберите данные из Aggregate Report и View Results Tree для</li> </ol>
--------------------------	--

	анализа. 12. Сохраните результаты в формате .html
Источники	T3, SRS
Инструменты	Apache JMeter.
Критерий успеха	Время отклика при запросах с указанной в SRS нагрузкой соответствует заявленным требованиям в пункте 3.4.1.

### 5.2.6 Load Testing (Нагрузочное тестирование)

Параметр	Описание
Цель	Проверка работы системы под высокой нагрузкой.
Описание процесса	<ol style="list-style-type: none"> <li>1. Откройте Apache JMeter и создайте новый тестовый план.</li> <li>2. Thread Group: <ol style="list-style-type: none"> <li>a. Потоки: 200.</li> <li>b. Ramp-Up Period: 20 секунд.</li> <li>c. Loop Count: бесконечный или нужное количество циклов.</li> </ol> </li> <li>3. HTTP Header Manager: <ol style="list-style-type: none"> <li>a. Добавьте заголовки, необходимые для запросов (например, Content-Type: application/json или Authorization: Bearer &lt;token&gt;).</li> </ol> </li> <li>4. HTTP Request: <ol style="list-style-type: none"> <li>a. Укажите URL API-эндпоинта, который необходимо протестировать.</li> <li>b. Настройте метод запроса (GET, POST и т.д.).</li> <li>c. Если нужен запрос с телом (например, POST), добавьте JSON или другой формат в поле Body Data.</li> </ol> </li> <li>5. Constant Throughput Timer: <ol style="list-style-type: none"> <li>a. Укажите желаемую скорость запросов: 200 RPS.</li> </ol> </li> <li>6. View Results Tree: <ol style="list-style-type: none"> <li>a. Позволяет отследить</li> </ol> </li> </ol>

	<p>каждый запрос, его ответ и статус (успех или ошибка).</p> <ol style="list-style-type: none"> <li>7. Aggregate Graph: <ol style="list-style-type: none"> <li>а. Для визуального отображения метрик производительности (время отклика, пропускная способность).</li> </ol> </li> <li>8. Aggregate Report: <ol style="list-style-type: none"> <li>а. Для сбора подробных данных о времени отклика, количестве успешных и неуспешных запросов.</li> </ol> </li> <li>9. Duration Assertion: <ol style="list-style-type: none"> <li>а. Установите ограничение времени отклика (например, <math>\leq 3</math> секунд).</li> </ol> </li> <li>10. Запустите тест.</li> <li>11. Соберите данные из Aggregate Report и View Results Tree для анализа.</li> <li>12. Сохраните результаты в формате .html</li> </ol>
Источники	SLA, SRS.
Инструменты	Apache JMeter.
Критерий успеха	Система остается стабильной при нагрузках в 2 раза выше, чем были согласованы с заказчиком в SRS (3.4.2)

### 5.2.7 Stress Testing (Стрессовое тестирование)

Параметр	Описание
Цель	Оценка поведения системы при экстремальных нагрузках и её способности к восстановлению.
Описание процесса	<ol style="list-style-type: none"> <li>1. Откройте Apache JMeter и создайте новый тестовый план.</li> <li>2. Thread Group: <ol style="list-style-type: none"> <li>а. Потоки: 500.</li> <li>б. Ramp-Up Period: 50 секунд.</li> <li>в. Loop Count: бесконечный или нужное количество циклов.</li> </ol> </li> <li>3. HTTP Header Manager: <ol style="list-style-type: none"> <li>а. Добавьте заголовки, необходимые для запросов (например,</li> </ol> </li> </ol>

	<p>Content-Type: application/json или Authorization: Bearer &lt;token&gt;).</p> <ol style="list-style-type: none"> <li>4. HTTP Request:             <ol style="list-style-type: none"> <li>a. Укажите URL API-эндпоинта, который необходимо протестировать.</li> <li>b. Настройте метод запроса (GET, POST и т.д.).</li> <li>c. Если нужен запрос с телом (например, POST), добавьте JSON или другой формат в поле Body Data.</li> </ol> </li> <li>5. Constant Throughput Timer:             <ol style="list-style-type: none"> <li>a. Укажите желаемую скорость запросов: 500 RPS.</li> </ol> </li> <li>6. View Results Tree:             <ol style="list-style-type: none"> <li>a. Позволяет отследить каждый запрос, его ответ и статус (успех или ошибка).</li> </ol> </li> <li>7. Aggregate Graph:             <ol style="list-style-type: none"> <li>a. Для визуального отображения метрик производительности (время отклика, пропускная способность).</li> </ol> </li> <li>8. Aggregate Report:             <ol style="list-style-type: none"> <li>a. Для сбора подробных данных о времени отклика, количестве успешных и неуспешных запросов.</li> </ol> </li> <li>9. Duration Assertion:             <ol style="list-style-type: none"> <li>a. убираем ограничение, чтобы явно видеть запросы которые перегружают приложения, в них мы будем получать 500</li> </ol> </li> <li>10. Запустите тест.</li> <li>11. Соберите данные из Aggregate Report и View Results Tree для анализа.</li> <li>12. Сохраните результаты в формате .html</li> </ol>
Источники	ТЗ
Инструменты	Apache JMeter.
Критерий успеха	Система корректно восстанавливается после превышения допустимой нагрузки.



	Интерфейс доступен и может функционировать при большом количестве данных на бэкенд стороне.
--	---

#### 5.2.8 Security and Access Control Testing (Тестирование безопасности и прав доступа)

Параметр	Описание
Цель	Проверка защиты системы микросервисов от уязвимостей и правильной реализации прав доступа.
Описание процесса	<ol style="list-style-type: none"> <li>1. Используйте Postman для тестирования API.</li> <li>2. Проверьте: <ol style="list-style-type: none"> <li>а. Попытку выполнения административных операций (например, создание пользователя) под разными ролями.</li> <li>б. Проверьте, что только администратор может успешно создать пользователя.</li> </ol> </li> <li>3. Проверьте JWT-токены: <ol style="list-style-type: none"> <li>а. Проверка срока действия.</li> <li>б. Использование токена для доступа к запрещённым ресурсам.</li> </ol> </li> <li>4. Документируйте все обнаруженные уязвимости и нарушения прав доступа.</li> </ol>
Источники	SRS
Инструменты	Postman для ручных проверок API.
Критерий успеха	<ol style="list-style-type: none"> <li>1. Уязвимости на уровне безопасности отсутствуют (например, SQL-инъекции или XSS).</li> <li>2. Доступ к защищённым эндпоинтам возможен только для авторизованных пользователей с соответствующей ролью.</li> <li>3. Аутентификация и авторизация работают корректно, а токены защищены от несанкционированного использования.</li> </ol>

#### Критерии успеха для всех видов тестирования

- Успешное выполнение всех ключевых сценариев использования.

- Отсутствие критических ошибок в функциональности.
- Производительность системы соответствует SLA (время отклика  $\leq 3$  секунд, обработка нагрузки до 100 пользователей).
- Безопасность данных подтверждена инструментами тестирования.

## 6. Entry and Exit Criteria (Критерии старта и окончания)

### 6.1. Test Plan Entry Criteria (Критерий старта)

Тестирование может начаться только при выполнении следующих условий:

Критерий	Описание
Доступность тестовой среды	Подготовлено окружение для тестирования: настроены серверы, базы данных, микросервисы и API Gateway.
Готовность функционала	Завершена разработка и интеграция всех компонентов, подлежащих тестированию.
Документация	Доступны актуальные версии: SRS, SAD, Use Case Specification.
Автоматизация тестов	Написаны и готовы к запуску тесты для модульного и интеграционного тестирования.
Инструменты для тестирования	Установлены и настроены необходимые инструменты: Selenium, JMeter, Postman
Подготовка тестовых данных	Тестовые данные с различными сценариями загружены в базу данных. Созданы пользователи с различными ролями.
Согласование тест-плана	Тест-план утвержден всеми заинтересованными сторонами.

### 6.2 Test Plan Exit Criteria (Критерий окончания)

*[Укажите условие, при котором процесс тестирования считается окончанным.]*

Тестирование считается завершенным при выполнении следующих условий:

Критерий	Описание
Покрытие тестов	Все ключевые сценарии (один сценарий) и функциональные требования (одного сервиса) протестированы.
Прохождение тестов	Все критические баги устранены, и система проходит $\geq 95\%$ тестов без ошибок.
Производительность	Система соответствует требованиям SLA: время отклика API $\leq 3$ секунд, поддержка до

	100 пользователей. (для критической ручки создания катаклизма)
Документирование ошибок	Все найденные баги зафиксированы в журнале инцидентов и либо устранены, либо классифицированы как некритичные.
Отчеты о тестировании	Сформированы отчеты о выполнении тестов, включая покрытие, производительность и качество системы.
Утверждение QA-командой	QA-команда подтверждает, что система готова к следующему этапу (релизу или пользовательскому тестированию).

### 6.3 Suspension and Resumption Criteria (Критерий паузы и возобновления)

*[Укажите условие, при котором необходимо приостановить процесс тестирования и при котором продолжить.]*

**Тестирование может быть приостановлено или возобновлено при следующих условиях:**

Критерий приостановки	Описание
Критические ошибки	Обнаружены критические баги, которые блокируют дальнейшее тестирование (например, отказ API Gateway).
Недоступность тестовой среды	Тестовая среда или серверы временно недоступны.
Несогласованность функционала	Открыты или изменены требования, что делает тесты не соответствующими текущей версии системы.

Критерий возобновления	Описание
Устранение ошибок	Критические баги устранены, протестированы и подтверждены QA-командой.
Доступность среды	Тестовая среда полностью восстановлена.
Уточнение требований	Все изменения в требованиях согласованы, документация обновлена, тестовые сценарии адаптированы.

## 7. Deliverables (Ожидаемые результаты тестирования)

*[В данном разделе перечислите артефакты, которые будут созданы в процессе тестирования.]*

В данном разделе описаны артефакты, которые будут подготовлены и представлены по результатам тестирования системы ИСУВИ.

### 7.1 Test Evaluation Summaries (Результаты выполнения тестов)

*[Опишите формат и содержание результатов выполнения тестирования]*

Название документа	Описание	Формат
Отчет о тестах БД	Логи выполненных запросов для проверки корректности операций с базой данных.	файл лога с расширением .txt Пример: <a href="#">ссылка</a>
Отчет о покрытии тестов	Статистика по количеству написанных, выполненных и успешно пройденных тестов.	Html отчет о покрытии тестами сделанный с помощью утилиты jасосо. Пример: <a href="#">ссылка</a>
UI тесты (selenium)	Краткое описание ключевых тестов, лог выполнения тестов. В логах отображены основные вызовы ui компонентов.	файл лога с расширением .txt Пример: <a href="#">ссылка</a>
UI тесты	Описание выполненных действий с описанием, как отреагировала система.	Таблица со списком тестов и результатом их прохождения. Пример: <a href="#">ссылка</a>
Анализ производительности	Отчет по времени отклика, включающий графики времени выполнения запросов, а также анализ устойчивости системы при разных условиях нагрузки.	HTML отчет сгенерированный через jmeter. Пример: <a href="#">ссылка</a>
Анализ нагрузки	Отчет по поддержке нагрузки, демонстрирующий, как система справляется с увеличением количества запросов. Включает графики времени выполнения и пропускной способности.	HTML отчет сгенерированный через jmeter. Пример: <a href="#">ссылка</a>
Отчет стресс тестов	Отчет о результатах стресс-тестирования, включая информацию о критических точках системы и анализ ее поведения при экстремальных нагрузках.	HTML отчет сгенерированный через jmeter. Пример: <a href="#">ссылка</a>
Отчет тестов безопасности	Описание выполненных действий с описанием, как отреагировала система.	Таблица со списком тестов и результатом их прохождения. Пример: <a href="#">ссылка</a>

## 7.2 Perceived Quality Reports (Оценка качества)

*[Опишите формат и содержание отчета о качестве разрабатываемой системы]*

Название отчета	Описание	Формат
Общий отчет о качестве	Оценка качества системы на основе тестов (функциональность, производительность, безопасность, UI).	.doc/.docx файл с описанием того как выполнены параметры качества описанные в SRS (3.4)
Оценка готовности к релизу	Заключение QA-команды о готовности системы к следующему этапу (релизу или пользовательскому тестированию).	Документ с кратким описанием проведенных тестов, также успешно пройдены тесты в деплое.

## 7.3 Incident Logs and Change Requests (Журналы ошибок и изменений)

*[Опишите, каким образом будут фиксироваться найденные ошибки в системе, а также изменения, сделанные с целью их исправить.]*

Название артефакта	Описание	Формат
Журнал ошибок	Список выявленных ошибок с указанием их типа, степени критичности и текущего статуса (открыто/закрыто).	Таблица excel в которой будем пометать баги с указанием их критичности. Также столбец со статусами по исправлению и столбец с именем исполнителя задачи.
Запросы на изменения	Документ с запросами на изменения, предложенными на основе результатов тестирования.	
Лог выполнения тестов	Автоматически сгенерированный лог выполнения тестов, генерируемый при их выполнении.	Лог можно проверить в gitlab в ci/cd процессе и выгрузить в .txt файл для отчетности.

## 8. Environmental Needs (Необходимое окружение для проведения тестирования)

*[Данный раздел содержит описание ресурсов (за исключением людей), необходимых для выполнения плана тестирования.]*

### 8.1 Base System Hardware (Базовое аппаратное обеспечение)

*[Опишите в таблице, приведенной ниже, конфигурацию систем(ы), на которой будут запускаться тесты]*

Для тестирования производительности, нагрузочного и стресс-тестирования будет использоваться **Helios**.

Resource (Ресурс)	Quantity (Количество)	Name and Type (Название и тип)
Процессор	16 ядер	Intel(R) Xeon(R) CPU E5-2643 0 @ 3.30GHz.
Оперативная память	128 ГБ	DDR4, рабочая частота $\geq 2400$ МГц.
Накопитель	6 дисков	559 ГБ x 2 SSD, 5.5 ТБ x 4 HDD.
Сетевой интерфейс	1	Intel(R) Ethernet Controller X710 для 10GbE SFP+.

Остальное тестирование проводится на **MacBook pro m1**

Resource (Ресурс)	Quantity (Количество)	Name and Type (Название и тип)
Процессор	8 ядер CPU + 8 ядер GPU	Apple M1, ARM-архитектура.
Оперативная память	16 ГБ	Унифицированная память, рабочая частота $\geq 4266$ МГц.
Накопитель	1 диск	SSD, объём 512 ГБ.
Сетевой интерфейс	1	Wi-Fi 6 (802.11ax), поддержка Thunderbolt/USB 4.

## 8.2 Base Software Elements in the Test Environment (Базовые программы тестового окружения)

*[Опишите в таблице, приведенное ниже, какие программы должны быть установлены на тестовой(ых) системе(ах).]*

Software Element Name (Название)	Version (Версия)	Type (Тип)
OpenJDK	17	Среда выполнения для микросервисов.
Node.js	18.x	Среда выполнения для фронтенда и вспомогательных скриптов.

PostgreSQL	16.4	База данных для хранения данных системы.
Apache JMeter	5.x	Инструмент для нагрузочного и стресс-тестирования.
Selenium	Последняя стабильная	Инструмент для тестирования UI.
pgAdmin	Последняя стабильная	Инструмент для работы с базой данных.
Postman	Последняя стабильная	Инструмент для тестирования API.

**8.3 Productivity and Support Tools (Вспомогательные инструменты)**

*[Опишите в таблице, приведенное ниже, какие программы будут полезны для проведения тестирования.]*

Tool Category or Type (Тип программы)	Tool Brand Name (Название)	Vendor (Производитель)	Version (Версия)
Логирование	Встроенные средства логирования в микросервисах	-	Внутренняя реализация
Среда разработки	IntelliJ IDEA	JetBrains	2024.1.4
Система контроля версий	Git	Open Source Community	2.39.5
Веб-браузер	Google Chrome	Google	131.0.6778.110
Веб-браузер	Safari	Apple	18.0

**9. Responsibilities, Staffing, and Training Needs (Обязанности сотрудников)**

*[В данном разделе описываются необходимые навыки и знания людей, осуществляющих процесс тестирования.]*

**9.1 People and Roles (Люди и роли)**

Role (Роль)	Minimum Resources Recommended (Минимально необходимое количество людей)	Specific Responsibilities (Обязанности)
QA Lead	1	- Разработка и управление тестовым процессом.

		<ul style="list-style-type: none"> <li>- Составление тест-плана и контроль выполнения.</li> <li>- Координация команды тестировщиков.</li> </ul>
QA Engineer	1-3	<ul style="list-style-type: none"> <li>- Проведение функциональных, интеграционных и нагрузочных тестов.</li> <li>- Документирование найденных багов.</li> <li>- Подготовка отчетов о результатах тестирования.</li> </ul>
Automation QA Engineer	1-2	<ul style="list-style-type: none"> <li>- Разработка автоматизированных тестов для модульного, интеграционного и UI-тестирования.</li> <li>- Настройка инструментов автоматизации.</li> </ul>
DevOps Engineer	1	<ul style="list-style-type: none"> <li>- Поддержка тестового окружения.</li> <li>- Мониторинг серверов и управление ресурсами для тестирования.</li> </ul>
Database Specialist	1	<ul style="list-style-type: none"> <li>- Анализ базы данных на предмет целостности данных.</li> <li>- Проведение тестов SQL-скриптов.</li> </ul>

## 10. Management Process and Procedures (Управление)

*[Данный раздел содержит описание различных мероприятий по управлению процессом тестирования]*

### 10.1 Reporting on Test Coverage (Сообщение о тестовом покрытии)

*[Опишите процесс рецензирования результатов тестирования.]*

Элемент	Описание
Цель	Предоставление информации о выполненных тестах, покрытии сценариев и статусе исправления ошибок.
Методы мониторинга	<ul style="list-style-type: none"> <li>- Автоматические отчёты, сгенерированные JUnit, Selenium и Apache JMeter.</li> <li>- Ручной анализ через Postman и pgAdmin.</li> </ul>
Частота отчётности	- Итоговый отчёт по завершении тестирования.
Формат отчётов	<ul style="list-style-type: none"> <li>- HTML отчёты (генерация из инструментов тестирования).</li> <li>- Документы Excel для сводных данных в Google Sheets.</li> <li>- Логи выполнения тестов в формате .txt.</li> <li>- Итоговые отчёты в формате Google Документов (Google</li> </ul>



	Docs).
Ответственный	QA Lead совместно с Automation QA Engineers.

## 10.2 Problem Reporting, Escalation, and Issue Resolution (Выявление, избегание и решение проблем)

*[Опишите, каким образом будет вестись учет проблем, возникших во время выполнения тестов, и какие действия нужно предпринять для их решения.]*

Этап	Описание
Выявление проблемы	<ul style="list-style-type: none"> <li>- Проблемы фиксируются в Google Sheets с указанием приоритета, описания и текущего статуса (Открыто/Исправлено/Закрыто).</li> <li>- Указание типа проблемы (функциональная, производительность, UI, безопасность).</li> </ul>
Эскалация	<ul style="list-style-type: none"> <li>- Критические ошибки немедленно передаются QA Lead и разработчикам.</li> <li>- Менее критичные проблемы обрабатываются в порядке очереди.</li> </ul>
Решение	<ul style="list-style-type: none"> <li>- Разработчики исправляют баги и создают соответствующие pull-запросы.</li> <li>- QA-инженеры повторно тестируют исправленные ошибки.</li> </ul>
Ответственный	QA Lead совместно с командой разработчиков.

## 10.3 Approval and Signoff (Утверждение плана тестирования)

*[Опишите процесс утверждения данного плана тестирования, а также укажите список лиц, участвующих в нём.]*

Этап утверждения	Описание
Подготовка	QA Lead завершает документ тест-плана, включая пункты о покрытии, критериях и результатах тестирования.
Утверждение документа	Тест-план утверждается руководителем проекта или техническим директором с использованием Google

	Docs.
Формат	Финальная версия документа хранится в Google Docs.