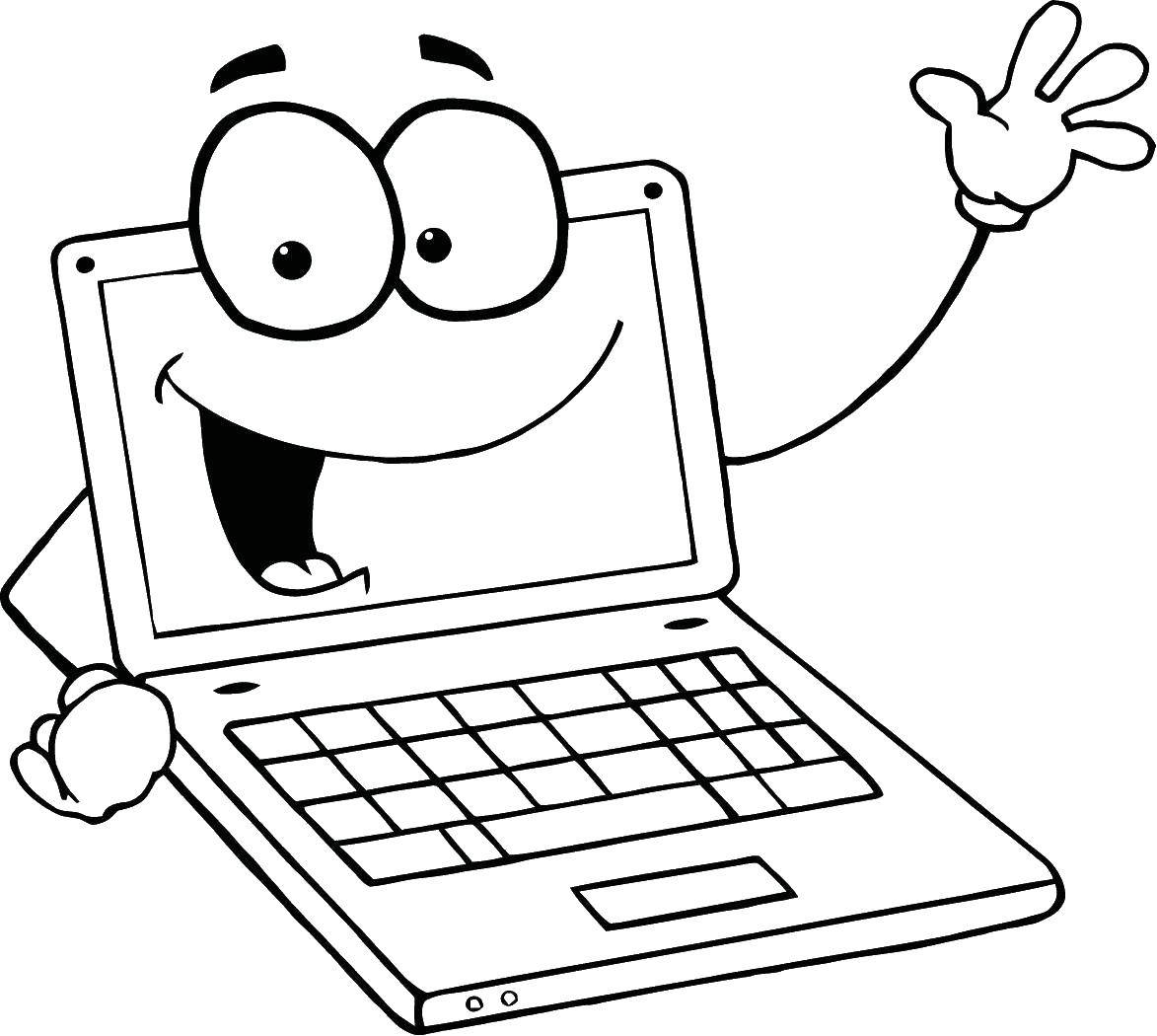
**Test Plan (v 20.12.23)**



**1.** **Introduction (Введение)**

**1.1** **Purpose**

Данный документ представляет собой план тестирования ИС “Финист”.

**1.2 Scope (Область применения)**

Данный документ предназначен для разработчиков системы “Финист” и описывает план тестирования системы перед релизом. Он также может служить отчётом перед стейкхолдерами в рамках приёмочного тестирования.

**1.3** **Intended Audience (Предполагаемая аудитория)**

Может использоваться заказчиком и будет использоваться командой разработки для контроля качества разрабатываемой системы.

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

TDD (test-driven development) - техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест.

**1.5** **References (Ссылки)**

* [*Vision*](https://docs.google.com/document/d/1Et3FG_X17h4TG_VhagQ7rDRJbUPPGXa0/edit)
* [*SDP*](https://docs.google.com/document/d/1ORaBCZ9OZ__4-H_0NbBh14hKTVwRDkQvLb25Uwyzr64/edit)
* [*RiskList*](https://docs.google.com/document/d/1a00LoZT9BMtYX72rNu5aCoQsbu_n5IoJ/edit)
* [*Glossary*](https://docs.google.com/document/d/1reBBvZCW8-QsYr5F5Ixum-bbKSkMxmGXP-yf2m_T3RU/edit)
* [*UseCase*](https://docs.google.com/document/d/1ydDMM3psIyYvH4MX7Mlkt2e8n_XKm07K/edit)
* [*SRS*](https://docs.google.com/document/d/1KZPhcczRg4XHN3is2uTxyx07Gj8I-aY-/edit)
* [*SDP*](https://docs.google.com/document/u/0/d/1ORaBCZ9OZ__4-H_0NbBh14hKTVwRDkQvLb25Uwyzr64/edit)
* [*SAD*](https://docs.google.com/document/d/1HiD5ntAlroIWEwKhZmxedCMvdzvv0j_RwxdwoSpTfmE/edit)

**1.6** **Document Structure (Структура документа)**

* Цель и мотивы тестирования - описывает, с какой целью проводится тестирование проекта и какие источники информации для этого будут применены.
* Целевые объекты тестирования - перечисляет, что именно будет проверяться с помощью тестов.
* План тестов - перечисляет все виды тестов, которые будут включены в процесс тестирования.
* Подход к тестированию - представляет рекомендованные стратегии для разработки и (не все виды тестов обязательно должны быть реализованы).
* Критерии старта и окончания - указывает условия, при которых процесс тестирования начинается, считается оконченным и приостанавливается.
* Ожидаемые результаты тестирования - перечисляет артефакты, которые будут созданы в процессе тестирования.
* Необходимое окружение для проведения тестирования - содержит описание ресурсов, необходимых для выполнения плана тестирования.
* Обязанности сотрудников - описываются необходимые навыки и знания людей, осуществляющих процесс тестирования.
* Управление - содержит описание различных мероприятий по управлению процессом тестирования.

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

**2.1** **Background (Справочная информация)**

Данная информация содержится в документе [Vision](https://docs.google.com/document/d/1Et3FG_X17h4TG_VhagQ7rDRJbUPPGXa0/edit), разделах 2,3 и 4.

**2.2** **Evaluation Mission (Цели тестирования)**

Тестирование направлено на улучшение качества создаваемого программного продукта. Планируется, что проведение тестов поможет выявить максимальное количество ошибок до завершения окончательной разработки и выпуска продукта. В дополнение, автоматизированные тесты обеспечат более безопасные возможности рефакторинга кода и внедрения новых функциональных возможностей в будущем.

Также, результаты тестирования могут служить подтверждением работоспособности системы для стейкхолдеров. Тем не менее, они вправе провести независимый аудит.

В план тестирования также включено нагрузочное тестирования, проведение которого преследует следующие цели:

* выявить максимальную производительность тестируемой системы
* подтвердить стабильность работы тестируемой системы под нагрузкой на уровне не ниже требований по производительности
* подтвердить отказоустойчивость тестируемой системы при возникновении нештатных ситуаций

**2.3** **Test Motivators (Мотивы тестирования)**

Источниками информации для тестирования будут служить функциональные и нефункциональные требования, описанные в разделах 6, 7 и 9 документа [Vision](https://docs.google.com/document/d/1Et3FG_X17h4TG_VhagQ7rDRJbUPPGXa0/edit); пользовательские сценарии использования ([UseCase](https://docs.google.com/document/d/1ydDMM3psIyYvH4MX7Mlkt2e8n_XKm07K/edit) документ); контролируемые технические риски из документа [RiskList](https://docs.google.com/document/d/1a00LoZT9BMtYX72rNu5aCoQsbu_n5IoJ/edit).

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

* Frontend: на основе пользовательских сценариев.
* Backend: на основе предъявленных ранее в других документах требований к производительности и стабильности.
* База данных, оборудование, сеть: стандартные проверки на отказоустойчивость при взаимодействии с backend. За поддержку работоспособности отвечает высококлассный сетевой инженер Афанасьев Дмитрий Борисович, поэтому проблем быть не должно.

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

* Функциональное тестирование - проверка пользовательских сценариев на выполнимость. *При выполнении фазы тестирования, приоритет будет отдан именно этому тестированию.*
* Нагрузочное тестирование - выявить максимальную производительность системы; подтвердить стабильность работы систёмы под нагрузкой на уровне не ниже требований по производительности; подтвердить отказоустойчивость системы при возникновении внештатных ситуаций. Снизить риски эксплуатации системы в промышленной среде.
* Unit-тестирование - тестирование отдельных классов или функций на корректное выполнение ими своих обязанностей.
* Интеграционное тестирование - тестирование API, взаимодействия фронтенда с бекендом, и бекенда с базой данных.
* Тестирование пользовательского интерфейса.
* Тестирование отказоустойчивости
* Тестирование безопасности
* Volume-тестирование

Стоит учитывать, что решение, создавать автотесты или выполнять ручное тестирование, будет принято позднее, основываясь на том, успевает ли команда разработки в срок.

**5.** **Test Approach (Подход к тестированию)**

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

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

| Technique Objective:  (Цель) | Выявить дефекты функциональности или подтвердить их отсутствие. |
| --- | --- |
| Technique:  (Описание процесса) | 0. Подготовка входных данных для тестируемой системы;  1. Проверка ограничений целостности:   * Создать тестовые данные, включающие дублирующиеся значения в столбце с уникальным ограничением: * дублирование первичного ключа: инициализировать все таблицы и попытаться добавить в каждую запись с первичным ключом, уже содержащемся в таблице. Ожидаемый результат: ошибка добавления записи - нарушение уникальности. * Создать записи в одной таблице, имеющие ссылки на несуществующие записи в связанной таблице.   + заполнить таблицы permission и t\_role тестовыми значениями с первичными ключами [a;b] и [c;d]. соответственно, где a, b,c,d **∈ N**   + Попытаться добавить записи в таблицу role\_to\_permission со полей permission\_id и role\_id не входящими в отрезки [a;b] и [c;d] соответственно. * Вставить данные, нарушающие условия ограничений CHECK или других правил, и проверить, будет ли это принято базой данных.   2. Проверка связей между таблицами:   * Испытать внешние ключи, добавляя, обновляя и удаляя записи в связанных таблицах, и убедиться, что изменения корректно отражаются в других таблицах:   + заполнить таблицы users и mariages тестовыми данными, соблюдая ограничения целостности, создав связи между таблицами с помощью внешних ключей bride\_id и groom\_id в таблице marriages.   + Произвести каскадное удаление записей в таблице users.   + Убедиться, что связанные записи в таблице marriages также были удалены.   3. Проверка правильности данных:   * Вставить данные неправильного типа в столбец и убедиться, что база данных отклоняет такие записи:   + для полей типа integer использовать строковые значения;   + для полей типа varchar(n) использовать значения, длина которых превышает n;   + для полей типа date использовать поля типа varchar, boolean, int, причем поля типа varchar не должны представлять текстовую запись даты. * Вставить значения, выходящие за пределы заданного диапазона, и убедиться, что такие данные не принимаются базой данных.   4. Тестирование транзакций и согласованности данных:   * Провести операции чтения и записи одновременно несколькими пользователями, чтобы проверить соблюдение ACID-свойств базой данных при параллельных операциях.   + подготовить скрипт на любом языке программирования в котором: создаются несколько подключений к базе данных различными пользователями;   + в параллельных потоках производится одновременное чтение и запись разными пользователями из/в одну таблицу;   + написать скрипт, в котором до завершения транзакции записи в таблицу будет начата другая транзакция чтения из той же таблицы.   5. Проверка корректных запросов в БД:   * Добавление (убедиться, что данные были добавлены) * Обновление (убедиться, что данные были обновлены) * Удаление (убедиться, что данные были удалены)   7. Формирование отчета по итоговым результатам |
| Oracles:  (Источники) | SAD, SRS |
| Required Tools:  (Инструменты) | PostgreSQL, IDEA с расширением DataGrip, SQL-скрипты |
| Success Criteria:  (Критерий успеха) | * Данные записываются в базу; * Отсутствие ошибок во время проверки ограничений целостности. * Соответствие данных заданным стандартам и ожиданиям. |

***5.2.2 Business Case Testing (Тестирование бизнес-кейсов)***

| Technique Objective:  (Цель) | Проверка тому, что система позволяет выполнить все пользовательские сценарии в полном объёме |
| --- | --- |
| Technique:  (Описание процесса) | 1. Изучение функциональных требований, участвующих в сценарии  2. Разработать набор тест-кейсов, которые будут тестироваться  Пример подхода по тестированию:   * Выполнение определенных шагов сценария (в конце запрос к серверной части системы) * Проверка результата, полученного от серверной части   3. Формирование отчета по итоговым результатам  Сценарий основного BC:   1. Авторизоваться в системе с ролью “Сваха” 2. Перейти на страницу “Комната Свахи” 3. Нажать кнопку “Создать состязание” 4. Задать город, название и 5 участников: жених, невеста (3 шт), супостатъ, помощникъ) 5. Подробно описать испытания для жениха и невест 6. Запустить состязание |
| Oracles:  (Источники) | UseCase |
| Required Tools:  (Инструменты) | Selenium |
| Success Criteria:  (Критерий успеха) | Все пользовательские сценарии можно выполнить; неправильные данные не приводят к крашу ИС; система соответствует всем функциональным требованиям; |

***5.2.3 User Interface Testing (Тестирование интерфейса)***

| Technique Objective:  (Цель) | Проверка того, что интерфейс не ломается, а интерактивные элементы работают в соответствии с сценариями использования. |
| --- | --- |
| Technique:  (Описание процесса) | 1. Проверка интерфейса на соответствие [гайдлайнам](https://www.figma.com/file/VDgCvItanpcz2LAeEGY4mb/UI-KIT-(Community)?type=design&mode=design&t=ahOxDss60W4UHHcn-1) из макета (цвета, шрифты, элементы интерфейса). |
| Oracles:  (Источники) | UseCase |
| Required Tools:  (Инструменты) | Selenium, WebKit debugging tools |
| Success Criteria:  (Критерий успеха) | При использовании актуальной версии Google Chrome сайт отображается корректно: разметка не сломана, кнопки нажимаются, поля заполняются, I/O c API работает. |

### *5.2.4 Performance Profiling (Тестирование производительности)*

| Technique Objective:  (Цель) | Оценка производительности, устойчивости и способности системы справляться с заданной нагрузкой. |
| --- | --- |
| Technique:  (Описание процесса) | 1. Формирование профиля нагрузки на основе наиболее часто используемых сценариев (процент вхождения > 90%) 2. Подача нагрузки на систему с помощью JMeter для создания нагрузки разного уровня 3. Сбор метрик времени отклика и ответа сервера |
| Oracles:  (Источники) | SRS  Use Case Specification |
| Required Tools:  (Инструменты) | * Postgres * JMeter * JVM * telegraf, prometheus, grafana |
| Success Criteria:  (Критерий успеха) | Среднее время отклика системы соответствует проектной документации и уменьшается при уменьшении нагрузки |

### 

### *5.2.5 Load Testing (Нагрузочное тестирование)*

| Technique Objective:  (Цель) | Оценка производительности, устойчивости и способности системы справляться с нагрузкой, превышающей запланированную |
| --- | --- |
| Technique:  (Описание процесса) | 1. Запуск БД Postgres 2. Запуск серверного приложения 3. Сформировать профиль нагрузки (использовать сформированный в рамках тестирования производительности) 4. Нагрузить систему с помощью JMeter на уровне нагрузки 150% от бизнес-требований 5. Собрать статистику работы системы после 1 часа работы 6. Измерить среднее время отклика системы |
| Oracles:  (Источники) | SRS  Use Case Specification |
| Required Tools:  (Инструменты) | Postgres  JMeter  telegraf, prometheus, grafana |
| Success Criteria:  (Критерий успеха) | Система продолжает функционировать при высоких нагрузках. Не происходит отказов приложения. Среднее время ответа на запрос составляет менее 1 минуты |

### *5.2.6 Stress Testing (Стрессовое тестирование)*

| Technique Objective:  (Цель) | Оценка производительности, устойчивости и способности системы справляться с заданной нагрузкой. |
| --- | --- |
| Technique:  (Описание процесса) | 1. Запуск БД Postgres 2. Запуск серверного приложения 3. Сформировать профиль нагрузки (использовать сформированный в рамках тестирования производительности) 4. Запуск тестов с помощью JMeter для создания нагрузки разного уровня 5. Сбор метрик времени отклика и ответа сервера 6. Увеличение нагрузки по получения отказа сервера 7. Определение максимальных пиковых значений нагрузки |
| Oracles:  (Источники) | SRS  Use Case Specification |
| Required Tools:  (Инструменты) | * Postgres * JMeter * JVM * telegraf, prometheus, grafana |
| Success Criteria:  (Критерий успеха) | Система перестала отвечать либо невозможно увеличить нагрузку для получения отказа системы |

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

| Technique Objective:  (Цель) | Проверка тестируемого продукта с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками ПО, отказами оборудования или проблемами связи/сети. |
| --- | --- |
| Technique:  (Описание процесса) | 1. Формирование профиля нагрузки (использовать сформированный в рамках тестирования производительности) 2. Запуск БД Postgres 3. Запуск серверного приложения 4. Запуск тестов с помощью JMeter на нагрузке уровня бизнес-требований 5. Сбор метрик времени отклика и ответа сервера 6. После 20 минут стабильной работы эмулируется кратковременная недоступность сервера приложений (остановка инстанса на сервере и его перезапуск) 7. Фиксируется время потеря работоспособности, время восстановления 8. Тест завершается через 20 минут стабильной работы после эмулируемого сбоя |
| Oracles:  (Источники) | SRS  Use Case Specification |
| Required Tools:  (Инструменты) | * Postgres * JMeter * JVM * telegraf, prometheus, grafana |
| Success Criteria:  (Критерий успеха) | Система успешно восстановилась после перезагрузки сервера приложений; Подтверждена стабильность работы после перезагрузки не менее чем в течение 20 минут |

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

| Technique Objective:  (Цель) | Определение и анализ уровня защиты системы от возможных угроз безопасности, а с целью получения несанкционированного доступа к данным и функциональности. |
| --- | --- |
| Technique:  (Описание процесса) | * Проверка кода с помощью инструмента find security bugs в WebKit. * Попытки входа с помощью некорректных данных. * Ввод SQL выражений в поля ввода и через Swagger. * Попытки выполнить запрос без токена/с неправильным токеном. |
| Oracles:  (Источники) | RiskList |
| Required Tools:  (Инструменты) | WebKit debugging tools, OpenAPI |
| Success Criteria:  (Критерий успеха) | Для зарегистрированных пользователей система предоставляет права в соответствии с их ролью. Система не доступна по некорректному логину и паролю. Пароли хранятся в БД в зашифрованном виде. |

*5.2.9 Volume Testing (Объемное тестирование)*

| Technique Objective:  (Цель) | Оценить производительность приложения при увеличении объема данных в базе данных для выявления возможных узких мест, проблем производительности и оценки поведения системы при больших объемах информации. |
| --- | --- |
| Technique:  (Описание процесса) | * Определение часто используемых таблиц, выявление наиболее узких мест:   с помощью логирования запросов или мониторинга производительности PostgreSQL (pg\_stat\_statements) определить набор таблиц, которые используются в 80% всех запросов в рамках всех бизнес-кейсов.   * Подготовка SQL-скриптов для записи большого объема данных в определенные ранее таблицы.   Для определенных на предыдущем этапе таблиц подготовить скрипты для записи тестовых данных в большом объеме. Возможно использование генераторов тестовых данных.  Заполнение определенных таблиц тестовыми данными.   * Выполнение SQL-запросов, участвующих в бизнес-сценариях, анализ производительности. |
| Oracles:  (Источники) | SAD, SRS |
| Required Tools:  (Инструменты) | PostgreSQL, IDEA с расширением DataGrip, SQL-скрипты |
| Success Criteria:  (Критерий успеха) | При увеличении объема данных, время операций чтения, записи и обновления в БД не превышает установленного лимита. |

*5.2.10 Functional testing (Функциональное тестирование)*

| Technique Objective:  (Цель) | Проверка реализованной функциональности в соответствии с описанными прецедентами |
| --- | --- |
| Technique:  (Описание процесса) | В рамках тестирования реализуется проверка конкретных функций, методов или классов приложения. Включает модульное тестирование (тестирование отдельных модулей) и интеграционное тестирование (проверка взаимодействия между модулями).  С помощью автотестов проверка системы через публичный API:   * тестирование регистрации - система должна предоставлять возможность зарегистрировать новый аккаунт * система должна предоставлять возможность редактирования своего аккаунта * тестирование получения списка пользователей - система должна предоставлять список пользователей пользователю с ролью “Сваха” * тестирование создания состязания - система должна предоставлять возможность пользователю с ролью “сваха” создавать состязание * тест авторизации - проверка корректности работы аутентификации и авторизации: провести авторизацию под пользователями с разными ролями и убедиться в корректности работы механизма авторизации * проверка прав доступа - авторизация под пользователями с разными ролями и проверка доступности защищенных согласно ролевой модели ресурсов. * тест создания голосования - пользователь с ролью “Сваха” может перевести состязание в состояние “голосование”, создав при этом голосование; * тест участия в голосовании - проверка записи голоса и невозможности проголосовать второй раз в рамках одного и того же голосования. |
| Oracles:  (Источники) | SRS, UC |
| Required Tools:  (Инструменты) | JUnit |
| Success Criteria:  (Критерий успеха) | Тесты, покрывающие критическую функциональность (необходимую для выполнения UC), завершаются успешно. |

Тестирование установки не проводится ввиду отсутствия установки - для пользователей, сервис представляет из себя веб-приложение, которое не требуется установки каких-либо зависимостей на клиентское устройство.

Тестирование конфигурации также можно пропустить: бекенд часть будет развёрнута на одном оборудовании с единой конфигурацией ролью Implementer, который в команде является тем же человеком, что и разработчик.

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

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

Переход проекта в бета-стадию будет служить сигналом к старту функционального тестирования по UseCase. Нефункциональное тестирование может проводится при внесении любых изменений в код, как точечно, так и в рамках регрессионного тестирования.

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

Все тесты выполнены успешно, отсутствуют ошибки, помеченные как критические и ошибки, мешающие выполнить UseCase. Некритические ошибки можно исправить после релиза. Опираться на процент тестового покрытия тут не стоит, так как зачастую, оно избыточно.

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

Проведение планового тестирования функциональных возможностей приостанавливается, если соответствующий функционал еще не реализован, например, из-за задержек в графике разработки. Тестирование возобновляется после внедрения соответствующего функционала.

**7.** **Deliverables (Ожидаемые результаты тестирования)**

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

zip-архив с:

* автоматически-сгенерированными отчётами соответствующими утилитами
  + Selenium test doc (html)
  + Junit test doc (html)
  + JMeter test doc (xml)
  + План выполнения запросов Postgres (txt)
* текстовыми файлами с описанием исправлений, ссылками на задачу, технические подробности (опционально), при ручном тестировании. По одному файлу на тест.

Как распаковать zip:

| Распаковать ZIP-архив можно различными способами в зависимости от операционной системы. На Windows:  * **Встроенный инструмент:**   + Откройте проводник (File Explorer).   + Навигируйтесь к месту, где находится ZIP-файл.   + Щелкните правой кнопкой мыши на архиве.   + Выберите "Извлечь все..." (Extract All...).   + Укажите путь для распаковки и нажмите "Извлечь" (Extract). * **При помощи программы WinRAR или 7-Zip:**   + Если у вас установлены WinRAR или 7-Zip, вы можете открыть архив с их помощью и извлечь файлы.  На Linux:  * **Командная строка:**   + Откройте терминал.   + Используйте команду unzip: unzip имя\_файла.zip   + Если у вас нет unzip, вы можете установить его при помощи пакетного менеджера вашего дистрибутива. * **Архиваторы в графическом интерфейсе:**   + В большинстве дистрибутивов Linux есть архиваторы с графическим интерфейсом, такие как File Roller (в GNOME) или Ark (в KDE). Вы можете использовать их для работы с ZIP-архивами.  На macOS:  * **Встроенный инструмент:**   + Найдите ZIP-файл в Finder.   + Дважды щелкните по нему. macOS автоматически извлечет содержимое в ту же папку. * **Архиваторы в графическом интерфейсе:**   + Аналогично Linux, на macOS есть архиваторы с графическим интерфейсом, такие как Archive Utility. Вы можете также использовать сторонние программы, такие как The Unarchiver.   Пожалуйста, учтите, что точные шаги могут немного различаться в зависимости от версии операционной системы и используемого программного обеспечения. |
| --- |

**7.2** **Perceived Quality Reports (Оценка качества)**

Численные критерии:

1. Процент покрытия тестами
2. Количество успешных/неуспешных тестов к общему количеству
3. Количество заблокированных тестов (тех, которые невозможно выполнить из-за того, что функциональность ещё не реализована). Такие тесты могут существовать, если мы применим TDD концепцию.

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

Найденная ошибка будет зафиксирована в системе Github Issues, ей будет присвоен приоритет, описаны шаги воспроизведения, назначен ответственный за ошибку.

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

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

Ввиду отсутствия выделенной тестовой фермы, все тесты будут выполняться на dev-окружении (личном оборудовании разработчиков).

| **Resource**  **(Ресурс)** | **Quantity**  **(Количество)** | **Name and Type**  **(Название и тип)** |
| --- | --- | --- |
| Helios | 1 | 128GB ram / 560GB space / Intel Xeon E5-2643 3.3GHz / 16 cores |

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

| **Software Element Name**  **(Название)** | **Version**  **(Версия)** | **Type**  **(Тип)** |
| --- | --- | --- |
| JDK | 11+ | Среды выполнения кода бекенда |
| PostgreSQL | 14+ | База данных |
| Firefox | 117+ | Браузер |

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

| **Tool Category or Type**  **(Тип программы)** | **Tool Brand Name**  **(Название)** | **Vendor**  **(Производитель)** | **Version**  **(Версия)** |
| --- | --- | --- | --- |
| IDE (для разработки, отладки, удобного запуска тестов) | IDEA Ultimate | JetBrains | 2020+ |
| JS-движок, (разработка, отладка, удобный запуск тестов) | NodeJS | OpenJS Foundation | 17+ |
| Инструмент автоматизация тестирования веб-приложений\* (возможно будет применено ручное тестирование) | Selenium | Software Freedom Conservancy | 4+ |
| Инструмент unit-тестирования | JUnit | JUnit-Team | 4+ |
| Инструмент нагрузочного тестирования | JMeter | Apache Foundation | 5+ |
| Инструмент тестирования API | Swagger | SmartBear Software | 3+ |
| Github | Хранение кода, агрегирование проблем | Github, Inc | Dec 2023 |

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

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

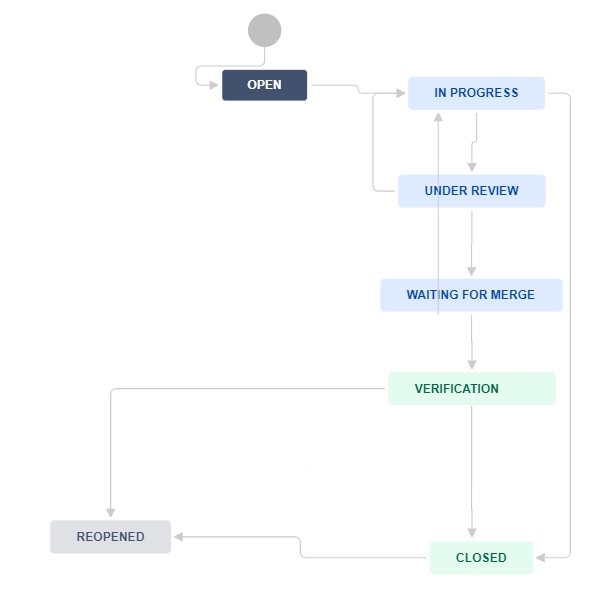
В таблице ниже, под написанием тестов подразумевается написание любых тестов (интеграционных, функциональных, unit и т.д.)

| **Role**  **(Роль)** | **Minimum Resources Recommended**  **(Минимально необходимое количество людей)** | **Specific Responsbilities**  **(Обязанности)** |
| --- | --- | --- |
| Tester | 1 | Выполнение ручного тестирования, написание автотестов при наличии соответствующих навыков, агрегирование информации об ошибках. |
| Implementer | 2 | Написание тестов на frontend/backend, поддержка другого разработчика, если он занят написанием функциональности. |
| Project Manager | 1 | Определяет, какие ошибки будут пофикшены в ближайший цикл разработки. |

**10.** **Management Process and Procedures (Управление)**

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

1. Тестировщик просматривают отчёты о тестировании, сгенерированные соответствующей программой.
2. Ошибки ранжируются по критичности и сложности исправления.
3. Ошибки назначаются на разработчика, ответственного за проблемную функциональность ИЛИ, в случае его занятости, на другого разработчика, знакомого с технологическим стеком.
4. Проводится исправление ошибки.
5. Проводится ревью исправлений.
6. Тикет ошибки закрывается или отправляется на доработку.



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

При выявлении недочетов тестировщик обязан составить соответствующую задачу, включив в нее описание проблемы, и передать ее разработчикам, указав уровень приоритета. Разработчики в свою очередь осуществляют выполнение задач в порядке установленного приоритета. Для логирования задач б использоваться какая-либо система. Ввиду небольшой команды разработки и размещения кода на Github, удобнее всего будет использовать фичу сервиса - Github Issues.

**10.3**  **Approval and Signoff (Утверждение плана тестирования)**

В процессе утверждения плана тестирования участвуют RUP-роли Tester, Implementer, Project Manager. Tester заводит ошибки в системе и назначает им приоритеты. Implementer определяет сроки выполнения и порядок исправлений в зависимости от неисправной функциональности. Project Manager утверждает план тестирования на ближайший цикл работ в зависимости от глобального приоритета команды на этот цикл, желаний заказчика и задач на расширение функциональности.