**Test-plan**

**CodePET**

Версия: 1.0 Дата создания: 26.01.2024

[**История версий 3**](#_p2p8tf1e4cnr)

[**Введение 4**](#_3v631n5ponzc)

[**Описание продукта 4**](#_dplxgiw8d5nl)

[**Области тестирования 5**](#_uvbarpkvl3a9)

[**Области, не подвергаемые тестированию**](#_hqqocvdk8v1b) **5**

[**Тестовая стратегия и подходы 6**](#_t7plkbgbcnyc)

[**Этапы тестирования 8**](#_5pxwwslhsv7n)

[**Критерии 9**](#_43padjl89os4)

[Критерии начала тестирования 9](#_crznomacubnq)

[Критерии завершения тестирования 9](#_dgwu4yytuf7)

**Фиксация ошибок** ……………………………………………………………………...…………………………………………………..10

[**Обязанности**](#_x39mb590buu1) **11**

[**Расписание 1**](#_z5ucsdsc67p7)**2**

**Завершение процесса тестирования….………………….………………………………………………………….13**

[**Окружение тестируемой системы 1**](#_pl1wwzx0q1ui)**3**

[**Риски 1**](#_dxpxkf3w6x0j)**4**

[**Документация 1**](#_ylj97ocll55h)**5**

[Функциональное и нефункциональное тестирование 1](#_rsc0ww3cz9uj)5

[**Метрики 1**](#_orq9wgp5u8z6)**5**

[**Глоссарий 1**](#_4jimfhmqlxir)**5**

# 

# 

# 

# 

# 

# 

# **История версий**

| **Дата изменения** | **Автор изменения** | **Внесённые изменения** |
| --- | --- | --- |
| 26.01.2024 | Федоров В.С | Создание тест-плана |
| 28.01.2024 | Кузьмина Ю.А. | Дополнение тест-плана |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# **Введение**

Данный документ описывает процессы, связанные с проведением функционального и нефункционального тестирования продукта. Тестирование будет фокусироваться на функциональных аспектах приложения - от процессов регистрации и авторизации, до функциональности создания, просмотра, добавления и поиска pet-проектов. Также будет выполнено тестирование UI/UX интерфейса и usability.

Продуктом является веб-приложение CodePET, разработанное для поиска, организации и управления пет-проектами в области разработки программного обеспечения.

Целью тестирования веб-приложения CodePET является проверка соответствия всем заявленным требованиям основного функционала приложения и его дизайна, убедиться, что весь функционал исправлен и понятен пользователям; выявить возможные проблемы работы приложения, и повысить качество разрабатываемого продукта.

# **Описание продукта**

**CodePET** - это веб-приложение, разработанное для поиска, организации, хранения и управления пет-проектами в области разработки программного обеспечения. Оно предназначено для выпускников школ программирования, которые хотят получить практический опыт и совершенствовать свои навыки путем участия в реальных проектах. А также оно будет интересно для опытных разработчиков которые хотят реализовать что-то новое, для менеджеров проектов и для компаний, которые хотят создать тестовое MVP нового продукта.

**Цель** CodePET - предоставить выпускникам школ программирования практический опыт работы над реальными проектами, а также помочь им развиваться и совершенствовать свои навыки разработки ПО.

Платформа создает возможность для сотрудничества, обмена знаниями и построения портфолио, что может помочь выпускникам школ программирования в их будущей карьере в IT-сфере.

**Ценность** - Участие в проекте поможет получить реальный опыт работы, поучаствовать в разработке приложения с нуля, поработать в кросс-функциональной команде и положить интересный кейс в портфолио.

# 

# **Области тестирования**

Будет проведено системное функциональное тестирование следующих компонентов:

* **фич-лист** - общий документ команды, описывающий функции разрабатываемого приложения, приоритеты реализации, ограничения и допуски, критерии оценки и сроки;
* **требования** - документ, описывающий требования к разрабатываемому приложению: макет и структура приложения, правила навигации, допускаемые значения;
* **GUI -** графический интерфейс пользователя;
* **сервис регистрации -** возможность пользователей регистрироваться на платформе приложения;
* **сервис авторизации и аутентификации -** возможность подтверждения пользователей в системе;
* **сервис поиска -** возможность находить информацию по ключевому вводу;
* **сервис контактов**
* **личный кабинет -** стартовая страница, отображающая все проекты пользователя с возможностью их просмотра, а также профиль с личными настройками;
* **сервис проектов -** библиотека хранящихся проектов, возможность предпросмотра выбранных проектов, просмотр статуса и фильтрация проектов, подписки и уведомления

# **Области, не подвергаемые тестированию**

* Unit-тестирование
* База данных
* Тестирование безопасности
* Тестирование производительности
* Стрессовое тестирование

# **Тестовая стратегия и подходы**

**Подготовка к тестированию**

Приоритетным способом проведения тестирования принято ручное тестирование.

Разработчиками команды будет настроен тестовый стенд.

Необходимо провести системное тестирование для проверки функциональности, взаимосвязи между всеми компонентами приложения на уровне пользовательского интерфейса, выявления багов, связанных с работой приложения.

Версия приложения будет условно разделена на несколько сервисов, разной логики (см. раздел “Области тестирования”, компоненты, подлежащие тестированию). Тест-кейсы или чек-листы будут написаны для этих сервисов. Запланировано последовательное проведение тестирования на основе фич-листа.

В качестве системы управления тестированием используется инструмент QASE io. Он позволяет планировать ручное тестирование, хранить информацию о проведенных проверках, осуществлять очередность проведения тестов, в соответствии с планом тестирования, а также предоставлять сводную информацию в виде отчетов. QASE io интегрирован с баг-трекинговым ботом в GitHub, что позволяет оперативно и информативно заводить отчеты об ошибках, обнаруженных в процессе тестирования и контролировать процесс их устранения.

Каждое поведение приложения отличное от ожидаемого в процессе ручного тестирования будет включено в баг-репорт. Найденные ошибки будут оформлены в системе QASE io и переданы в GitHub.

В процессе функционального тестирования приложения сначала будут выполняться позитивные проверки, потом - негативные.

**Будут проведены следующие виды тестирования:**

**по целям:**

* функциональное ( API, User Interface)
* нефункциональное (UI, кроссбраузерное и кроссплатформенное тестирование)

**по видам, связанным с изменениями в программном продукте:**

* **smoke** - тестирование основного функционала, а также базовые негативные проверки; выявление критических ошибок на начальном этапе;
* **re-test** - тестирование исправленных функциональностей от разработчиков;
* **critical path** - тестирование всех стандартных действий пользователя, которые он совершает, используя приложение;
* **sanity** - проверка работоспособности нового функционала, сервиса или внесенных изменений в существующий функционал;
* **regression** - тестирование всей функциональности системы после внесенных изменений в продукте, с целью проверки, что ранее разработанный функционал исправен.

**по формализации:**

* исследовательское тестирование

**по видам входных данных**:

* позитивное
* негативное

В процессе тестирования будут применены следующие техники тест-дизайна: *классы эквивалентности*, *граничные значения, матрица трассировки, таблица принятия решений, и техника “Причины и следствия”.*

# 

# **Этапы тестирования**

1. **Сбор требований**

1.1. Сбор всех требований к продукту и приведение их к доступной и понятной форме

1.2. Создание матрицы трассировки, которая будет отображать полноту покрытия требований проверками.

1. **Анализ требований**

2.1. Обсуждение User Flow

2.2. Создание Mind map

1. **Планирование тестирования**

3.1. Создание тест-плана

3.2. Утверждение тест-плана

3.3. Определение пространства для распределения задач (канбан-доска в GitHub)

1. **Проектирование тестов**

4.1. Создание чек-листов с высокоуровневыми проверками

4.2. Создание тестовых данных

1. **Настройка тестового окружения**

5.1. Настройка докера или сервера

5.2. Настройка Postman

1. **Выполнение тестов**

6.1. Выполнение проверок на основании чек-листов в ручном режиме

6.2. Запуск проверок в ручном режиме

1. **Сообщение о дефектах и их отслеживание**

7.1. Верификация дефектов

7.2. Создание баг-репортов в QASE

7.3. Устранение дефектов

7.4. Re-test

1. **Завершение тестирования**

8.1. Создание отчетов по тестированию

8.2. Оценка процесса тестирования

# **Критерии**

## **Критерии начала тестирования**

1. Утверждение требований. Все требования проверены и утверждены. Разработка тест-плана, тест-кейсов, чек-листов начинается по окончании составления требований.
2. Завершение разработки функциональности. Функциональность полностью разработана и встроена в систему приложения.
3. Подготовка тестового окружения. Выполнение тест-кейсов начинается после развертывания ПО на стенде для тестирования.
4. Тестовая документация утверждена.
5. Тестовые данные доступны команде тестировщиков.
6. Дефекты с высоким приоритетом устранены (критерий для проведения повторного тестирования)

## **Критерии завершения тестирования**

Тестирование завершается при одновременном соблюдении следующих условий:

1. выполнены запланированные чек-листы
2. 95% чек-листов пройдены успешно (статус “Passed”) при этом нет дефектов Blocker и Critical
3. 100% покрытие требований
4. на каждое требование должна быть разработана минимум 1 проверка (тест-кейс или чек-лист).

**Фиксация ошибок**

Все дефекты (баги), выявленные в процессе проведения тестирования, будут классифицированы по следующим критериям серьезности:

- **Trivial** - не существенный дефект, не влияет на функционал, но доставляет визуальное неудобство;

- **Minor** - не относится к внутреннему функционалу, но доставляет неудобство в использовании приложения;

* **Major** - существенный дефект, затрудняет работу основного функционала или некоторых функций;
* **Critical** - критический дефект, нарушает работу основного функционала, либо делает невозможным дальнейшую работу продукта;
* **Blocker** - блокирующий дефект, полностью останавливающий работу приложения (не имеет возможности его обойти).

Приняты следующие градации приоритетности дефектов:

* **Hight** - высокий приоритет. Дефект должен быть исправлен в первую очередь, несмотря на другие задачи;
* **Medium** - средний приоритет. Дефект должен быть исправлен согласно общей очередности;
* **Low** - низкий приоритет. Исправление дефекта в последнюю очередь.

**Требования к оформлению баг-репортов:**

1. Наличие уникального номера (id)
2. Название бага должно содержать суть ошибки и место ее возникновения
3. Указан приоритет бага
4. Указана серьезность бага
5. Предусловие - подробные сведения об условиях, предшествующих возникновению ошибки (опционально )
6. Описаны подробные шаги воспроизведения бага
7. Описан ожидаемый результат поведения приложения
8. Описан фактический результат поведения приложения
9. Указано тестовое окружение
10. Приложен скрин ошибки или видео её воспроизведения
11. Указан автор созданного баг-репорта

**Обязанности команды**

| Роль | Число сотрудников | Зона ответственности |
| --- | --- | --- |
| Ответственный | 1 | Общее руководство процессом тестирования, анализ статистики |
| Тестировщик | 4 | Участие в разработке тест-плана, выполнение тест-кейсов, чек-листов, заведение багов, составление отчетов, анализ статистики |

# 

# 

# 

# 

# **Расписание**

| Задача | Время на выполнение | Дата начала | Дата окончания |
| --- | --- | --- | --- |
| Анализ требований к продукту | 1 день | 23.01.2024 | 24.01.2024 |
| Выбор тестовой стратегии | 1 день | 24.01.2024 | 25.01.2024 |
| Составление User Flow, Mind Map, тест-плана и чек-листов | 6 дней | 25.01.2024 | 30.01.2024 |
| Выполнение чек-листов, репорт багов | 3 месяца | 26.01.2024 | 26.04.2024 |
| Анализ полученных результатов, подведение итогов | 1 день | 27.04.2024 | 28.04.2024 |

# Задачи будут распределены между членами команды согласно их занятости самостоятельно, ответственным команды, или продакт-менеджером. Задачи будут детально формироваться продакт-менеджером на основании фич-листа согласно текущего спринта. Распределение задач ведется в репозитории проекта, на канбан-доске в GitHub.

# Временные ресурсы команды определяются исходя из листа доступности.

Технические ресурсы - программное обеспечение на личных ПК членов команды, смартфонах, а также эмуляторы.

# **Завершение процесса тестирования**

В завершение цикла проверки, будет проведено регрессивное тестирование, согласно составленному тест-плану (без проведения новых тестов), с целью исключения сбоя в исполнении функций приложения после отладки багов.

После завершения тестирования разработанного продукта будет оформлен **Отчет о тестировании**, содержащий сводные результаты процесса тестирования и аналитические выводы о работе CodePET, а также **Отчет об ошибках**, содержащий сведения и статистику найденных и устраненных дефектов.

**Окружение тестируемой системы**

**Параметры:**

Операционная система: Windows 10 20Н2, Windows 11 21Н2

Browser:

* Yandex Browser версия 24.1.0.2570 (64-bit)
* Chrome версия 120.0.6099.225б (64 bit)
* Opera версия 106.0.4998.52
* Mozilla Firefox версия 122.0

Разрешение экрана: 1920\*1080 / 1366\*768

**Инструменты:**

Postman/Swagger - ПО для тестирования API

Figma - ПО для тестирования UX/UI

DevTools - для тестирования функциональностей

QASE Io - для ведения тестовой документации и оформления баг-репортов

GitHub - для хранения репозитория проекта, контроля версий разработки, планирования и организации рабочего процесса

# 

# **Риски**

| Риск | Вероятность возникновения | Способ смягчения |
| --- | --- | --- |
| Задержка выдачи продукта в тест | 70% | выполнение задач согласно приоритетов и сроков |
| Человеческий: временная потеря трудоспособности отдельными членами команды (больничные) | 80% | перераспределение задач |
| Не согласованное внесение изменений в проект или в требования | 10% | обсуждение любых изменений в проекте с командой |
| Тест-менеджмент (обратная связь) | 40% | внесение предложений и улучшений |
| Недостаток знаний и опыта тестировщиков | 40% | самообразование, обмен опытом, помощь коллег |

# 

# **Документация**

## **Функциональное и нефункциональное тестирование**

**Канбан-доска:** <https://github.com/orgs/Pet-projects-CodePET/projects/1/views/4>

**Фич-лист:** <https://docs.google.com/spreadsheets/d/1QXD36DZHGH-bGDUzqYXv3B9DDxJkbzYI0FapSGFM5vk/edit#gid=765184209>

**Требования:** <https://docs.google.com/document/d/1BJeF_fFi5o3zITS83mEWs93HuW7V__C9sm5aEcvypAs/edit#heading=h.oevrxvwrtvo5>

**Дизайн:** <https://www.figma.com/file/OkJgqsGOL7OKB2agjVMcST/CodePET?type=design&node-id=0-1&mode=design&t=rcWXgtVhldZbvz5W-0>

**Mindmap:**

<https://www.mindmeister.com/app/map/2977441503?t=eemdZzivqX>

# **Метрики**

1. *Выполнение чек-листов*  CL выполнение = CL выполненные/CL всего \* 100%
2. *Успешное прохождение чек-листов*  CL успешность = CL успешно выполненных/CL всего выполненных 100%

# 

# **Глоссарий**

API - Application Programming Interface - соглашение между клиентом и сервером. Механизмы, которые позволяют программным компонентам взаимодействовать друг с другом.

GUI - Graphical User Interface - графический пользовательский интерфейс, то отображение приложения, которое видит пользователь.

Mindmap - визуальная структура отображения задач, действий, поведения приложения.

UI (user interface) - тестирование интерфейса приложения с целью проверки функциональности, имитируя действия пользователей.

Usability - исследование, выполняемое с целью определения, удобен ли пользовательский интерфейс для его предполагаемого применения.

User Flow - сценарий взаимодействия пользователя с приложением.

UX (user experience) - пользовательский опыт; способы взаимодействия пользователя с приложением и оценка удобства его использования.

Баг - дефект или ошибка в работе приложения, его интерфейсе, любое некорректное поведение приложения.

Баг-репорт - отчет об ошибке; документ, содержащий детальные сведения о выявленном дефекте или ошибке.

Канбан-доска - инструмент управления проектом. Используется для представления задач, их распределения между участниками команды, контроля за их выполнением и упорядочивания рутинных действий.

Тест-кейс - алгоритм действий, которые необходимо выполнить тестировщику, чтобы проверить функционал. Содержит описание действий для подготовки к проверке, детальное описание последовательных шагов самой проверки и ожидаемый результат проверки.

Чек-лист (CL) — это список пунктов, которые тестировщик должен проверить во время тестирования. Он напоминает, что нужно проверить, но не детализирует, как это делать.