Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

ЛАБОРАТОРНАЯ РАБОТА №5

По дисциплине «Разработка и анализ требований»

Выполнила:

Студентка 2 курса 7 группы

Яскевич Валерия Александровна

2025 Минск

**1. Пользовательская история**

Для приложения "Благотворительный фонд" одной из ключевых функций для пользователей (волонтеров) является возможность делать пожертвования на благотворительные проекты. Это позволяет пользователям поддерживать инициативы фонда и участвовать в достижении их целей

**Пользовательская история**: как зарегистрированный пользователь, я хочу сделать пожертвование на благотворительный проект, чтобы внести свой вклад в достижение его финансовой цели.

**2. Разработка критериев приемки**

**2.1. Уточняющие вопросы заказчику**

Чтобы понять все детали функции пожертвований, я бы задал заказчику следующие вопросы:

1. Какие способы оплаты доступны для пользователей (например, только банковские карты или также банковские переводы)?
2. Каковы минимальная и максимальная суммы пожертвования?
3. Нужно ли отправлять пользователю подтверждение (например, квитанцию) по электронной почте после успешного пожертвования?
4. Какие сообщения об ошибках должны показываться при сбоях в оплате (например, если карта недействительна или на счете недостаточно средств)?
5. Есть ли требования к скорости обработки платежа (например, максимальное время транзакции)?
6. Какие меры безопасности должны быть реализованы для защиты платежных данных?
7. Должна ли система поддерживать разные валюты или языки в форме пожертвования?
8. Что происходит, если проект уже собрал необходимую сумму? Можно ли продолжать принимать пожертвования?
9. Должна ли быть возможность анонимного пожертвования (без указания имени донора)?
10. Нужно ли обновлять прогресс сбора средств в реальном времени после каждого пожертвования?

**2.2. Предполагаемые ответы**

Поскольку реальных ответов заказчика нет, я сделаю обоснованные предположения, основываясь на типичных требованиях для благотворительных приложений и контексте проекта:

1. Поддерживаются только банковские карты (Visa, MasterCard).
2. Минимальная сумма пожертвования — 1 BYN, максимальная — 10,000 BYN.
3. После успешного пожертвования пользователю отправляется электронное письмо с квитанцией, где указаны сумма, дата и название проекта.
4. Ошибки отображаются четкими сообщениями, например: "Неверные данные карты", "Недостаточно средств" или "Срок действия карты истек".
5. Платеж должен обрабатываться не дольше 5 секунд при нормальной нагрузке.
6. Платежные данные защищаются с помощью SSL-шифрования и токенизации (соответствие стандартам PCI DSS).
7. Система работает только с белорусскими рублями и на русском языке.
8. Если проект достиг финансовой цели, новые пожертвования отклоняются с соответствующим уведомлением.
9. Пользователь может выбрать опцию анонимного пожертвования, чтобы его имя не отображалось публично.
10. Прогресс сбора средств обновляется в реальном времени после каждой успешной транзакции.

**2.3. Критерии приемки (свод правил)**

На основе вопросов и ответов я составил критерии приемки, разделив их на положительные (ожидаемое поведение системы), негативные (обработка ошибок) и нефункциональные (дополнительные требования, такие как производительность и безопасность).

**Положительные критерии**:

1. Пользователь может выбрать любой активный благотворительный проект из списка доступных в приложении.
2. Пользователь может ввести сумму пожертвования в диапазоне от 1 до 10,000 BYN.
3. При корректных данных банковской карты и успешной оплате система сохраняет пожертвование в базе данных и увеличивает собранную сумму проекта.
4. После успешного пожертвования пользователь видит в интерфейсе сообщение: "Ваше пожертвование успешно обработано".
5. Система отправляет пользователю электронное письмо с квитанцией, содержащей сумму, дату, название проекта и уникальный идентификатор транзакции.
6. Пожертвование добавляется в историю транзакций пользователя, доступную в личном кабинете.
7. Если пользователь выбирает анонимное пожертвование, его имя не отображается в публичных списках доноров проекта.

**Негативные критерии**:

1. Если пользователь вводит сумму меньше 1 BYN или больше 10,000 BYN, система показывает сообщение: "Сумма должна быть от 1 до 10,000 BYN".
2. Если пользователь вводит неверные данные карты (например, неправильный номер, CVV или срок действия), система отображает сообщение: "Неверные данные карты. Проверьте и попробуйте снова".
3. Если на карте недостаточно средств, система показывает сообщение: "Недостаточно средств на карте".
4. Если проект уже достиг своей финансовой цели, система отклоняет пожертвование и показывает сообщение: "Проект достиг цели. Пожертвования больше не принимаются".

**Нефункциональные критерии**:

1. **Производительность**: Пожертвование обрабатывается за 5 секунд или быстрее при нагрузке до 100 одновременных пользователей.
2. **Безопасность**: Все платежные данные передаются через защищенное SSL-соединение и хранятся в токенизированном виде для соответствия стандартам PCI DSS.
3. **Удобство использования**: Форма пожертвования имеет интуитивный интерфейс с подсказками (например, формат ввода номера карты: "XXXX-XXXX-XXXX-XXXX").
4. **Доступность**: Форма пожертвования соответствует стандартам WCAG 2.1, обеспечивая поддержку экранных читалок для пользователей с ограниченными возможностями.

Эти критерии покрывают основные сценарии использования функции пожертвований, включая успешные и ошибочные случаи, а также учитывают важные нефункциональные аспекты, такие как безопасность и удобство.

**3. Критерии в формате Given-When-Then**

Формат Given-When-Then (Дано-Когда-Тогда) помогает описать поведение системы в конкретных сценариях, что удобно для тестирования. Я разработал три критерия, охватывающих ключевые случаи использования функции пожертвований.

1. **Успешное пожертвование**:
   * **Дано**: Пользователь авторизован в системе и находится на странице благотворительного проекта, который еще не достиг своей финансовой цели.
   * **Когда**: Пользователь вводит сумму 50 BYN, корректные данные банковской карты (номер, CVV, срок действия) и нажимает кнопку "Сделать пожертвование".
   * **Тогда**: Система подтверждает транзакцию, обновляет собранную сумму проекта, показывает сообщение "Ваше пожертвование успешно обработано" и отправляет пользователю квитанцию по электронной почте.
2. **Ошибка из-за неверных данных карты**:
   * **Дано**: Пользователь авторизован и находится на странице активного благотворительного проекта.
   * **Когда**: Пользователь вводит сумму 100 BYN, но указывает неверный CVV-код карты, затем нажимает "Сделать пожертвование".
   * **Тогда**: Система отклоняет транзакцию и показывает сообщение "Неверные данные карты. Проверьте и попробуйте снова".
3. **Попытка пожертвования на завершенный проект**:
   * **Дано**: Пользователь авторизован и находится на странице проекта, который уже достиг своей финансовой цели.
   * **Когда**: Пользователь вводит сумму 200 BYN, корректные данные карты и нажимает "Сделать пожертвование".
   * **Тогда**: Система отклоняет транзакцию и показывает сообщение "Проект достиг цели. Пожертвования больше не принимаются".

Эти сценарии четко описывают, как система должна реагировать в типичных ситуациях, и могут быть использованы для написания автоматических тестов.

**4. Какой формат записи критериев приемки лучше и почему**

В лабораторной работе требуется сравнить два формата записи критериев приемки: свод правил и Given-When-Then. Рассмотрим их преимущества и недостатки в контексте функции пожертвований.

* **Свод правил**:
  + **Плюсы**:
    - Компактный и понятный формат, подходящий для быстрого ознакомления.
    - Удобен для передачи заказчику или включения в документацию.
    - Хорошо описывает общие требования, включая нефункциональные аспекты (например, безопасность).
  + **Минусы**:
    - Может быть недостаточно детализированным для сложных сценариев.
    - Не всегда удобен для тестировщиков, так как требует ручного преобразования в тестовые случаи.
    - Меньше подходит для автоматизации тестирования.
* **Given-When-Then**:
  + **Плюсы**:
    - Четко структурирует сценарии, разделяя предусловия (Дано), действия (Когда) и ожидаемые результаты (Тогда).
    - Идеально подходит для тестирования, особенно для автоматизированных тестов (например, с использованием SpecFlow в C#).
    - Обеспечивает высокую детализацию для конкретных случаев, что упрощает проверку поведения системы.
  + **Минусы**:
    - Может быть громоздким, если нужно описать много сценариев.
    - Требует больше времени на написание по сравнению со сводом правил.
    - Менее удобен для передачи заказчику, который не знаком с этим форматом.

**Рекомендация**: Для функции пожертвований в вашем проекте я рекомендую использовать **формат Given-When-Then** как основной. Вот почему:

* Пожертвования включают множество сценариев (успешные платежи, ошибки, граничные случаи), которые удобно описывать в виде четких сценариев. Это помогает разработчикам и тестировщикам понять, как система должна работать в каждой ситуации.
* Формат Given-When-Then идеально подходит для этапа тестирования, который указан в вашем курсовом проекте (раздел 6, "Лист\_задания.docx"). Он позволяет легко преобразовать критерии в тестовые случаи, включая автоматизированные тесты, что актуально для C#-проекта.
* Сценарии в этом формате понятны и для заказчика, если объяснить их структуру, так как они описывают реальные пользовательские действия.

Однако **свод правил** можно использовать как дополнение для:

* Описания общих требований, особенно нефункциональных (например, производительность или безопасность).
* Включения в пояснительную записку или презентацию для заказчика, где нужна компактность.

Комбинация обоих форматов (Given-When-Then для детальных сценариев и свод правил для общих требований) обеспечит максимальную ясность и универсальность.

**5. Ответы на теоретические вопросы**

**5.1. Что означают термины**

* **Definition of Ready (DoR, Определение готовности)**: Это набор критериев, которые подтверждают, что пользовательская история или задача готова к началу разработки. Например, история должна быть четко описана, иметь критерии приемки, быть оцененной по трудозатратам и согласованной с командой. DoR помогает избежать работы над неподготовленными задачами, что экономит время и ресурсы.
* **Definition of Done (DoD, Определение завершенности)**: Это набор критериев, которые определяют, что задача или пользовательская история полностью завершена и соответствует стандартам качества. Например, код написан, протестирован, задокументирован, прошел код-ревью и развернут в тестовой среде. DoD гарантирует, что работа готова к использованию или демонстрации.
* **Acceptance Criteria (Критерии приемки)**: Это конкретные условия, которые пользовательская история должна выполнять, чтобы считаться успешно реализованной. Они описывают, как система должна вести себя с точки зрения пользователя или заказчика, и служат основой для тестирования. Например, для пожертвований это может быть: "Система отправляет квитанцию после успешной оплаты".

**5.2. На каких этапах разработки применяются эти критерии**

* **Definition of Ready**:
  + **Этап**: Планирование (например, планирование спринта в Scrum или подготовка backlog’а).
  + **Применение**: Используется, чтобы убедиться, что задачи четко определены и готовы к разработке. Команда проверяет наличие описания, критериев приемки и других необходимых деталей перед началом работы.
* **Definition of Done**:
  + **Этап**: Завершение задачи или спринта (например, во время ревью спринта или приемки работы).
  + **Применение**: Используется для проверки, что задача выполнена полностью: код написан, протестирован, задокументирован и соответствует стандартам качества. Это подтверждает, что работа готова к демонстрации или развертыванию.
* **Acceptance Criteria**:
  + **Этапы**:
    - **Анализ требований**: Критерии формулируются и согласовываются с заказчиком для понимания ожидаемого результата.
    - **Разработка**: Разработчики используют критерии как руководство для реализации функциональности.
    - **Тестирование**: Тестировщики проверяют, что система соответствует критериям, выполняя тесты.
  + **Применение**: Критерии служат ориентиром для всех этапов, связывая ожидания заказчика с реализацией и проверкой.

**5.3. Какие критерии уникальны для каждой User Story, а какие одинаковы для всех**

* **Уникальны для каждой User Story**:
  + **Acceptance Criteria**: Критерии приемки создаются индивидуально для каждой пользовательской истории, так как они описывают конкретное поведение системы для данной функции. Например, критерии для пожертвований (обработка платежей, отправка квитанций) отличаются от критериев для управления проектами (создание, редактирование).
* **Одинаковы для всех User Story**:
  + **Definition of Ready**: Обычно единый для всего проекта. Например, каждая пользовательская история должна иметь описание, критерии приемки, оценку трудозатрат и быть согласованной с командой. Эти требования не зависят от конкретной функции.
  + **Definition of Done**: Также единый для проекта. Например, все задачи должны быть протестированы, задокументированы и пройти код-ревью. Однако DoD может немного варьироваться для разных типов задач (например, для UI могут быть дополнительные требования к дизайну, а для backend — к производительности).