ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ «ГЕНЕЗИС»

АДРЕС: 111141, Г. МОСКВА, ВН. ТЕР. Г. МУНИЦИПАЛЬНЫЙ ОКРУГ ПЕРОВО, УЛ. ПЛЕХАНОВА, Д. 9

genesis1ooo@yandex.ru

ОПИСАНИЕ ЖИЗНЕННОГО ЦИКЛА

программы для ЭВМ “Genesis”

ПРОЕКТИРОВАНИЕ

Цель — усовершенствование и расширение возможностей существующей платформы.

Процесс проектирования улучшений включает следующие этапы:

1. **Анализ текущего состояния:**
   - Оценка существующего ПО, включая функциональность, пользовательский опыт и отзывы учащихся и преподавателей.
   - Идентификация сильных и слабых сторон ПО, анализ основных потребностей и запросов пользователей.

2. **Определение целей улучшений:**
   - Устанавливаются конкретные цели для улучшения ПО.

3. **Проектирование улучшений:**
   - Разработка концепции улучшений, включая дизайн интерфейса, функциональные возможности и технологические изменения.
   - Создание пользовательских сценариев, прототипов и мокапов для визуализации изменений.
   - Уточнение деталей улучшений, таких как новые функции, обновленные разделы курсов, усовершенствованные методы обратной связи.

4. **Техническое проектирование:**

○ Определение технических аспектов улучшений, включая выбор технологий, интеграцию новых функций с ПО, архитектурные изменения и тестирование новых компонентов.

КОНСТРУИРОВАНИЕ

Цель – разработка и реализация новых компонентов, функций и структур системы.

1. **Разработка новых компонентов:**

○ Написание программного кода, создание новых модулей, функций или интерфейсов в соответствии с разработанными спецификациями.

○ Тестирование новых компонентов на уровне их работоспособности и соответствия заданным функциональным требованиям.

2. **Интеграция новых элементов:**

○ Внедрение новых компонентов в текущую систему онлайн-обучения, проверка их совместимости и взаимодействия с другими элементами.

○ Тестирование интегрированных компонентов на уровне их работы в рамках системы в целом.

3. **Уточнение и улучшение:**

○ Реакция на обратную связь и результаты тестирования новых компонентов для их улучшения и доработки.

○ Корректировка ошибок, оптимизация производительности и удовлетворение требований пользователей.

СБОРКА

Цель — собрать и объединить разработанные компоненты кода в единое целое, готовое к установке и использованию.

1. **Подготовка окружения для сборки:**

○ Установка необходимых инструментов и компиляторов, создание окружения для сборки программного обеспечения, включая выбор инструментов для автоматизации процесса.

2. **Компиляция и сборка:**

○ Компиляция и сборка всех разработанных компонентов и исходного кода, включая файлы, библиотеки и другие необходимые ресурсы.

○ Объединение отдельных частей программы в единый исполняемый файл или комплект файлов.

ТЕСТИРОВАНИЕ

Цель — проверка функциональности, производительности и качества ПО.

1. **Подготовка тестовых данных:**

   ○ Создание тестовых сценариев, тестовых случаев и данных для проверки различных функций и возможностей платформы.

   ○ Подготовка тестовых сред для воспроизведения различных сценариев использования.

2. **Выполнение тестирования:**

   ○ Запуск тестовых сценариев для проверки функциональности и корректности работы системы.

   ○ Использование различных видов тестирования: функциональное, нагрузочное, совместимость с разными устройствами и т.д.

   ○ Регистрация результатов тестирования и выявление ошибок, недочетов или несоответствий требованиям.

3. **Анализ и отладка:**

   ○ Изучение выявленных проблем и ошибок, их классификация и отслеживание в системе управления ошибками (баг-трекере).

   ○ Корректировка ошибок, отладка кода, повторное тестирование исправленных компонентов.

4. **Тестирование производительности:**

   ○ Проведение тестирования производительности для оценки скорости, надежности и общей производительности системы в условиях реальной нагрузки.

   ○ Идентификация узких мест и возможных проблем в производительности.

5. **Проверка безопасности:**

   ○ Проведение тестирования на безопасность для обнаружения уязвимостей и защиты от возможных атак.

   ○ Анализ уровня защищенности системы и внедрение улучшений для обеспечения безопасности.

6. **Документирование результатов:**

   ○ Запись всех результатов тестирования, включая найденные ошибки, улучшения и рекомендации.

   ○ Создание отчета о тестировании с подробным описанием результатов и рекомендаций по улучшению.

7. **Утверждение и релиз:**

   ○ Окончательное утверждение системы на основе результатов тестирования.

   ○ Подготовка к выпуску улучшенной версии системы, готовой для развертывания и использования.

ПРОЦЕСС ПОДДЕРЖКИ ПО

Цель — обеспечение непрерывной работы, поддержка пользователей и решение проблем

1. **Поддержка пользователей:**

   ○ Предоставление помощи пользователям по всем вопросам, связанным с использованием платформы обучения: от технических проблем до консультаций по функциональности.

   ○ Создание каналов коммуникации для обратной связи, таких как онлайн-чаты, электронная почта, телефонная поддержка и форумы.

2. **Управление запросами и инцидентами:**

   ○ Регистрация всех запросов и проблем пользователей, их классификация и приоритезация по уровню важности и воздействия на пользователей.

   ○ Отслеживание инцидентов от начала до решения и документирование процесса их решения.

3. **Обновления и исправления:**

   ○ Поддержка обновлений и исправлений ПО, включая установку новых версий, патчей и исправлений уязвимостей.

   ○ Планирование и контроль процесса обновлений, чтобы минимизировать простои системы и обеспечить безопасность.

4. **Документация и обучение:**

   ○ Подготовка документации для пользователей по использованию ПО, FAQ, руководств и видео-уроков.

   ○ Проведение обучающих сессий для персонала.

МЕНЕДЖМЕНТ КОНФИГУРАЦИИ ПО

Цель — управлении изменениями, состоянием и версиями компонентов программной системы на протяжении всего ее жизненного цикла. Главные задачи: контролируемая среда, целостность ПО, прозрачность процессов, оптимизация, безопасность и устойчивость, история изменений.

1. **Конфигурация**

   ○ Определены компоненты (составные части) ПО, включая их версии, взаимосвязи и зависимости.

   ○ Присвоены версии для отслеживания изменений.

2. **Контроль версий**

   ○ Используется GIT для контроля версий, отслеживания изменений, управления ветвлением кода, удобного внесения и отката изменений.

   ○ Изменения в исходном коде связываются с задачами, задачи связываются с версиями.

3. **Управление изменениями**

   ○ Оценка и классификация предложенных изменений в ПО по степени важности, риску, срочности.

   ○ Принятие решений о внесении изменений на основе установленных критериев.

   ○ Реализация, верификация и выпуск новых версий.

4. **Управление библиотекой конфигураций:**

   ○ Организация хранения и управления библиотекой компонентов ПО.

   ○ Обеспечение доступности версий и истории изменений для использования и восстановления предыдущих версий ПО.

5. **Документирование конфигурации:**

   ○ Ведется документация при каждом изменении состава или версии ПО.

6. **Конфигурационное аудирование:**

   ○ Проверка соответствия текущего состояния ПО установленным конфигурационным требованиям.

   ○ Аудит для проверки целостности, соответствия стандартам и правильности внесенных изменений.

ТРЕБОВАНИЯ К СПЕЦИАЛИСТУ

опыт работы с php и mysql от 1 года, html, css, js от 1 года, базовое знание директив веб-сервера apache, базовое знание управления сервером через консоль.

Процесс решения проблем с ПО

1. Регистрация и классификация проблемы:

1. **Обнаружение проблемы.** Процесс начинается с выявления проблемы в ПО. Способы обнаружения проблем:
    - Пользователи - сообщают об ошибках на контактный email genesis1ooo@yandex.ru Ответственный специалист задает уточняющие вопросы, чтобы узнать: где возникла проблема, при каких условиях, как воспроизвести проблему.
    - Внутренние команды - сообщают об ошибках ответственному за обслуживание ПО через корпоративную систему коммуникации. В сообщении указывают: где возникла проблема, при каких условиях, как воспроизвести проблему.
    - Системные уведомления - это система критических уведомлений в ПО (колокольчик). Ответственный специалист за обслуживание системы проверяет «колокольчик» 1 раз в неделю.
2. **Классификация проблемы.** Проблемы классифицируются по критичности.
    - Критические - стопорящие проблемы. Проблемы, без которых невозможна дальнейшая работа ключевых элементов (оплата, авторизация, доступ к контенту, переходы пользователя внутри системы). Срок решения: максимально быстро.
    - Стандартные. Проблемы, которые не влияют на ключевые функции, но из-за них приходится тратить больше времени или они доставляют дискомфорт в работе пользователей. Срок решения: 2 недели.
    - Незначительные. Проблемы, связанные с ошибками в текстах, оформление текста, мелкие проблемы в дизайне. Срок решения: 1 месяц.
3. **Регистрация проблемы.**

- Ответственный. За регистрацию проблемы отвечает специалист по обслуживанию системы.
- Срок регистрации. Проблема регистрируется в течение рабочего дня с момента ее обнаружения.
- Место регистрации. корпоративная система учета задач по работе с ПО
- Заполнение информации. Ответственный специалист вносит поступившую информацию об ошибке:
  - Заголовок в стиле «Что?, Где?, Когда?» (что произошло, где произошло, когда произошло).
  - Классификация.

2. Анализ проблемы и установление критериев решения:

1. **Исследование.** Ответственный специалист анализируют проблему, чтобы понять ее корень и причины. Процесс анализа:
   - воспроизведение ошибки на основе описания проблемы.
   - изучение ошибок, которые выводит система в «режиме разработчика» при вызове проблемной функции.
   - изучение лога ошибок.
   - изучение кода.
2. **Установление критериев решения.**
   - Специалист определяет как должна работать проблемная функция после решения проблемы.
   - Определяет конкретные исправления или улучшения, которые нужно внести.
3. **Внесение информации.** Подробная информация о проблеме заполняется в карточку задачи.

3. Решение проблемы:

1. **Смена статуса.** Когда специалист берет проблему в работу, то меняет статус проблемы на «в работе»
2. **Разработка исправления.** Ответственный специалист создает исправление для проблемы на основе анализа. Исправление может включать в себя внесение изменений в исходный код ПО и базу данных.
3. **Тестирование исправления.** После внесения изменений необходимо провести тестирование, чтобы убедиться, что исправление работает правильно и не вызывает новых проблем.

4. Тестирование:

1. **Смена статуса.** Когда специалист тестирует исправленную проблему, то меняет статус проблемы на «тестирование»
2. **Проверка исправления.** Исправление тестируется на копии проекта, в различных сценариях использования, чтобы убедиться в его надежности.
3. **Загрузка.** Исправление загружается на рабочую ПО.
4. **Верификация.** Проверка, что проблема больше не возникает, и что другие аспекты ПО не нарушены из-за внесенных изменений.

5 Закрытие проблемы:

- Оценка результата: Оценка выполненных работ и решения проблемы в соответствии с установленными критериями.
- Закрытие. Проблема закрывается в системе управления проблемами и выставляется статус «Решено».

Требования к специалисту

Для исправления ошибок по системе требуется программист не ниже уровня middle.

Стэк системы

- Бекенд: php, mysql
- Фронтенд: html, css, js
- Сервер: unix

Требования к программисту:

- опыт работы с php и mysql от 3 лет,
- html, css, js от 3 лет,
- базовое знание директив веб-сервера apache.
- базовое знание управления сервером через консоль.

Процесс совершенствования ПО

1. Выявление и классификация улучшений:

1. **Обнаружение проблемы.** Процесс начинается с выявления потенциальных улучшений. Способы выявления:
    - Пользователи - сообщают о предложениях на контактный email genesis1ooo@yandex.ru. Ответственный специалист задает уточняющие вопросы, чтобы узнать: где возникла проблема, при каких условиях, как воспроизвести проблему.
    - Внутренние команды - предлагают улучшения, которые могут повысить производительность, безопасность, продажи и удобство использования ПО.
    - Мониторинг - система мониторинга производительности и статистики использования ПО может выявлять узкие места и проблемные зоны. Ответственный специалист проверяет системы мониторинга 1 раз в месяц.
2. **Классификация улучшений.** Улучшения классифицируются по важности, срочности и сфере улучшения.
    - Критические / Срочные - улучшения, которые нужно внедрить до определенного события: распродажа, новые программы и т.п. Критерий определения улучшения как срочное — срок реализации диктуется внешними обстоятельствами и остается меньше 1 месяца до срока. Срок перехода к реализации: максимально быстро.
    - Важные - улучшения, которые повысят производительность или функциональность ПО, но не срочно необходимы. Срок перехода к реализации: следующий спринт.
    - Незначительные - улучшения, которые могут улучшить удобство использования, но не имеют негативного влияния на текущую работу. Срок перехода к реализации: принимается в следующий спринт в зависимости от загруженности важными и срочными задачами.
3. **Регистрация улучшений.**
    - Ответственный: Специалист по развитию ПО.
    - Время регистрации: Улучшение регистрируется в течение рабочего дня с момента получения предложения.
    - Место регистрации: корпоративная система учета задач по работе с ПО
    - Заполнение информации.
        - Краткий заголовок, который отражает задачу.
        - Подробное описание задачи с указанием результата.

        - Предварительная классификация: срочность, сфера улучшения

2. Анализ улучшений и установление критериев решения:

1. **Исследование.** Специалист по развитию ПО анализирует улучшение, чтобы понять его целесообразность и потенциальные риски.
    - Анализ и взаимодействие новой функции с другими частями ПО.
    - Определяются возможные варианты реализации.
2. **Установление критериев реализации.**
    - Определяются цели и ожидаемые результаты от улучшения.
    - Устанавливаются параметры, которые будут измеряться для оценки успешности улучшения.
    - Выбирается вариант реализации.
3. **Внесение информации.** Подробное описание улучшения и его критериев реализации заполняется в карточку задачи по развитию ПО.

3. Решение улучшений

1. **Смена статуса.** Когда специалист берет улучшение в работу, то меняет статус на «в работе»
2. **Разработка улучшения.** Ответственный специалист разрабатывает улучшение на основе анализа. Улучшение может включать в себя внесение изменений в исходный код ПО и базу данных.

4. Тестирование:

1. **Смена статуса.** Когда улучшение готово к тестированию, меняется статус на «тестирование»
2. **Проверка улучшения.** Улучшение тестируется на копии проекта в различных сценариях использования, чтобы убедиться в его надежности.
3. **Загрузка.** Исправление загружается на рабочую ПО.
4. **Верификация.** Проверка, что улучшение не вызывает проблем и соответствует установленным критериям.

5 Завершение улучшения:

- Оценка результата: Оценка выполненного улучшения в соответствии с установленными критериями.
- Закрытие. Улучшение закрывается в системе учета улучшений ПО и выставляется статус "Реализовано".

Требования к специалисту

Для исправления ошибок по системе требуется программист не ниже уровня middle.

Стэк системы

- Бекенд: php, mysql
- Фронтенд: html, css, js
- Сервер: unix

Требования к программисту:

- опыт работы с php и mysql от 3 лет,
- html, css, js от 3 лет,
- базовое знание директив веб-сервера apache.
- базовое знание управления сервером через консоль.