# Лабораторная работа №5

**Анализ рисков и характеристик качества ПО при внедрении**

1. **Цель работы**
   1. Изучить процесс анализа рисков ПО при разработке и внедрении.
2. **Литература**
   1. Зверева В. П., Сопровождение и обслуживание программного обеспечения компьютерных систем : учебник для студ. учреждений сред. проф. Образования / В. П. Зверева, А. В. Назаров. – М. : Издательский центр «Академия», 2018. – 256 с.
3. **Подготовка к работе**
   1. Повторить теоретический материал (см. п.2).
   2. Изучить описание лабораторной работы.
4. **Основное оборудование**
   1. Персональный компьютер.
5. **Задание**
   1. Выполнить анализ рисков при разработке и внедрении ПО по плану, представленному в приложении п.9;
   2. Составить таблицу оценки и минимизации рисков внедрения ПО:

|  |  |  |
| --- | --- | --- |
| Риск | Приоритет | Способ минимизации |
| Технические риски | | |
|  |  |  |
|  |  |  |
| Организационные риски | | |
|  |  |  |
|  |  |  |
| Экономические риски | | |
|  |  |  |
|  |  |  |
| Юридические риски | | |
|  |  |  |
|  |  |  |

* 1. Составить отчет по проделанной работе.

1. **Порядок выполнения работы**
   1. Повторить теоретический материал п. 3.1;
   2. Ответить на контрольные вопросы п. 8;
   3. Заполнить отчет п. 7.
2. **Содержание отчета**
   1. Титульный лист;
   2. Цель работы;
   3. Таблица п. 5.2
   4. Ответы на контрольные вопросы п. 6.3;
   5. Вывод по проделанной работе.
3. **Контрольные вопросы**
   1. Какие технические риски при разработке и внедрении ПО существуют?
   2. Какие организационные риски при разработке и внедрении ПО существуют?
   3. Какие экономические риски при разработке и внедрении ПО существуют?
   4. Какие юридические риски при разработке и внедрении ПО существуют?
4. **Приложение**

Проект разработки и внедрения программного обеспечения для task-tracking и Kanban-досок

1. Определение требований

1.1. Анализ целевой аудитории

* Кто будут пользователи: внутренние сотрудники компании, команды разработчиков, проектные менеджеры, или внешние клиенты.
* Какие задачи решает ПО: управление проектами, отслеживание задач, мониторинг рабочего времени, совместная работа.
* Какие существующие инструменты пользовались популярностью в вашей организации.

1.2. Функциональные требования

* Создание, отслеживание, редактирование задач.
* Настройка досок Kanban с различными этапами (Backlog, To-Do, In Progress, Done).
* Присвоение задач пользователям.
* Возможность создания подзадач.
* Приоритизация задач.
* Интеграция с другими системами (например, Jira, GitHub, Discord).
* Отслеживание прогресса (диаграммы Ганта, статистика выполнения задач).
* Возможность комментирования и вложения файлов.
* Автоматические уведомления и напоминания.

1.3. Нефункциональные требования

* Масштабируемость: поддержка роста числа пользователей.
* Безопасность: многослойная аутентификация, контроль доступа.
* UX/UI: интуитивно понятный интерфейс, адаптация под мобильные устройства.
* Производительность: высокая скорость загрузки даже при большом количестве задач и пользователей.

2. Техническое проектирование

2.1. Архитектура приложения

* Модель клиент-сервер: серверная часть на облаке или локальных серверах, клиентская часть в виде web-приложения.
* Микросервисная архитектура: для улучшенной масштабируемости и возможности раздельного обновления компонентов.
* Backend: REST API для взаимодействия клиента с сервером.
* Frontend: JavaScript-фреймворки (React, Angular, Vue.js) для динамичного интерфейса.

2.2. Выбор технологий

* Backend: Node.js с Express.js или .NET Core на C#.
* Frontend: React или Angular.
* База данных: PostgreSQL, MySQL или MongoDB для хранения задач, пользователей и данных досок.
* Управление задачами в реальном времени: WebSocket или другие технологии для обновления данных в режиме реального времени.

2.3. Интеграции

* API для интеграции с другими системами (GitHub, GitLab, Slack, Email).
* Встраивание системы отчетов и диаграмм.
* Поддержка CI/CD для автоматического развертывания и обновлений.

3. Разработка

3.1. Модульная разработка

* Модуль авторизации: OAuth 2.0 для авторизации через Google, GitHub, или другие провайдеры.
* Модуль управления задачами: CRUD операции для создания, редактирования и удаления задач.
* Kanban-доска: интерактивный интерфейс для перетаскивания карточек задач между колонками.
* Модуль уведомлений: e-mail, push-уведомления для напоминаний о дедлайнах.
* Модуль отчетности: графики выполнения задач, статус команд и пользователей.

3.2. Тестирование

* Написание unit-тестов для ключевых модулей.
* Интеграционные тесты для проверки взаимодействия между модулями.
* Регрессионное тестирование после каждого нового функционала.

4. Внедрение

4.1. Этапы внедрения

* Пилотный проект: внедрение на небольшой группе пользователей для получения обратной связи и устранения ошибок.
* Расширенное внедрение: поэтапное распространение на все подразделения компании.
* Техническая поддержка: обеспечение каналов для быстрого решения проблем (чат-поддержка, базы знаний).

4.2. Обучение пользователей

* Проведение обучающих сессий для сотрудников.
* Разработка документации и видеоинструкций по использованию.

5. Мониторинг и сопровождение

* Система мониторинга для отслеживания производительности и использования.
* Регулярные обновления ПО для исправления ошибок и улучшения функционала.
* Анализ фидбэка пользователей для улучшения UX и функциональных возможностей.

6. Оценка успеха проекта

* Ключевые показатели эффективности (KPI)
* Увеличение производительности команд.
* Снижение времени на выполнение задач.
* Удовлетворенность пользователей (опросы, рейтинги).

Пример временного плана (Gantt Chart)

* Определение требований — 2 недели.
* Архитектура и дизайн — 3 недели.
* Разработка — 10 недель.
* Тестирование — 4 недели.
* Внедрение — 2 недели.
* Мониторинг и сопровождение — на постоянной основе.

Проект может быть гибким, с возможностью добавления новых функциональных модулей в процессе эксплуатации.