Функциональная спецификация

1. История проекта

Проект был инициирован в связи с необходимостью кастомного решения в сфере отслеживания задач. Основными целями стали:

- Удобство использования ввиду персонализированного подхода.
- Повышение эффективности работы людей в повседневной жизни.

Этапы реализации проекта:

1. Анализ текущих бизнес-процессов и выявление проблем.
2. Сбор и анализ требований пользователей.
3. Разработка концептуального и логического проекта системы.
4. Реализация клиентских и серверных модулей.
5. Тестирование и отладка.
6. Внедрение в повседневную жизнь.
7. Постоянная поддержка и улучшение системы.

2. Цели дизайна

2.1. Требования пользователей

Система должна обеспечивать следующие возможности:

- Создание новой задачи.
- Удаление уже существующей задачи.
- Обновление содержания существующей задачи.
- Выставление приоритезации задачи.

2.2. Системные требования

Система должна поддерживать:

- Платформы:
  - Web.
- Архитектура сервера:
  - x86, x64.
- Минимальные характеристики:
  - Процессор: 2 ядра, 2 ГГц.
  - Оперативная память: 4 ГБ.
  - Место на диске: 250 ГБ.
  - Стабильное подключение к Интернету.

2.3. Сценарии использования

1. Пользователь:
   - Заходит в приложение, создает группу задач или одну конкретную.
   - Выставляет всю необходимую информацию для новых задач.
   - Изменяет уже существую задачи (всю информацию в них).
   - Удаляет задачи по мере необходимости.

3. Исключенные возможности и неподдерживаемые сценарии

- Система не поддерживает интеграцию с устаревшими операционными системами (например, Windows XP).
- Не реализована поддержка работы в оффлайн-режиме.

4. Предположения и зависимости

- Серверная часть размещена в облачном дата-центре.
- Пользователи имеют стабильное интернет-соединение.

5. Проект решения

5.1. Концептуальный проект

Система включает три основные компонента:

1. Серверная часть:
   - Центральная база данных для хранения информации о задачах.
   - REST API для взаимодействия с клиентскими приложениями.

2. Клиентская часть:
   - Веб-приложение.

5.2. Логический проект

- База данных:
  - Таблицы для хранения данных о пользователях, записях, расписаниях и уведомлениях.
- Приложения:
  - Web приложение на React.js.
  - Бэкенд на C# (с использованием ASP.NET Core).

5.3. Физический проект

- Сервер размещен в облачной инфраструктуре (например, Azure).
- Хранение данных на отказоустойчивых серверах с ежедневным резервным копированием.
- HTTPS для шифрования данных.

6. Требования к инсталляции и деинсталляции

- Установка:
  - Любой современный браузер.

- Доступ:
  - Веб-приложение доступно по уникальному URL.

- Деинсталляция:
  - Для веб-приложения удаление не требуется. Можно ограничить доступ в сети.

Архитектура системы управления задачами

1. Введение

Архитектура системы управления задачами разработана с учетом требований надежности, масштабируемости, простоты использования и поддержки. Система состоит из трех основных уровней:

1. Клиентский уровень (Frontend)
2. Серверный уровень (Backend)
3. Уровень базы данных (Database)

Каждый уровень имеет четко определенные функции и взаимодействует с другими через стандартизированные интерфейсы.

2. Архитектурный стиль

Архитектура системы базируется на многослойной архитектуре, где каждый уровень отвечает за выполнение строго определенных задач:

- Презентационный уровень (Frontend) отвечает за взаимодействие с пользователем.
- Логический уровень (Backend) обеспечивает обработку бизнес-логики и управление данными.
- Уровень данных (Database) отвечает за хранение и управление данными.

Для обмена данными между уровнями используется REST API(это способ взаимодействия сайтов и веб-приложений с сервером), обеспечивающий универсальный способ коммуникации.

3. Компоненты системы

3.1. Клиентский уровень (Frontend)

Клиентский уровень представлен веб-приложением, обеспечивающими доступ к функционалу системы.

Технологии:
- Веб-приложение: React.js для разработки пользовательского интерфейса.

Функциональность:
- Просмотр списка задач(http).
- Создание задачи(http).
- Удаление задачи(http).
- Обновлении данных задачи(http).
- Уведомления о статусе задачи(http).

3.2. Серверный уровень (Backend)

Серверный уровень реализует бизнес-логику и обеспечивает взаимодействие с базой данных.

Технологии:
- ASP.NET Core Web API.
- EF Core(простая, кроссплатформенная и расширяемая версия технологии доступа к данным Entity Framework с открытым исходным кодом)

Функциональность:
- Обработка http запросов от клиентских приложений.
- Выполнение бизнес-логики.
- Управление задачами.

- Обращение к базе данных через ORM.

3.3. Уровень базы данных (Database)

Уровень базы данных отвечает за хранение, управление и доступ к данным.

Технологии:
- PostgreSQL для управления реляционными данными.

Функциональность:
- Хранение информации о задачах.

4. Взаимодействие между компонентами

4.1. Клиентский уровень и серверный уровень

Взаимодействие реализуется через REST API:
- Клиент отправляет HTTP-запросы (GET, POST, PUT, DELETE) к серверу.
- Сервер обрабатывает запросы, выполняет бизнес-логику и возвращает данные в формате JSON.

4.2. Серверный уровень и уровень базы данных

Серверный уровень взаимодействует с базой данных через ORM (Object-Relational Mapping) (данные,представлены как таблицы - столбцы и строки, приведенная к классовой структуре) :
- Используется EntityFramework для работы с PostgreSQL(это система управления реляционными базами данных с открытым исходным кодом.Реляционная модель данных — логическая модель

данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики, как теория множеств и логика первого порядка).
- Запросы оптимизированы для минимизации времени отклика.

5. Масштабируемость и отказоустойчивость

5.1. Масштабируемость

- Использование контейнеров Docker для развертывания всех компонентов.
- Оркестрация с использованием Kubernetes.

5.2. Отказоустойчивость

- Репликация базы данных для предотвращения потери данных.
- Автоматические бэкапы каждые 24 часа.
- Использование Swagger(это инструмент, который помогает разработчикам создавать, документировать и проверять API(это набор правил, позволяющих одному программному продукту взаимодействовать с другим).Он позволяет программистам, техническим писателям и тестировщикам быстрее создавать, описывать и проверять программный интерфейс.), Prometheus(это набор инструментов с открытым исходным кодом,представляющий собой полноценную систему мониторинга и оповещения), Grafana(свободная программная система визуализации данных, ориентированная на данные систем ИТ-мониторинга.) для анализа запросов и состояния системы.

6. Безопасность

6.1. Аутентификация и авторизация

- Использование JWT(это открытый стандарт для создания токенов(метод аутентификации и авторизации пользователя) доступа, основанный на формате JSON.) для проверки подлинности пользователей.

6.2. Шифрование данных

- Все данные передаются через HTTPS(это безопасный протокол передачи данных).

6.3. Управление доступом

- Ограничение доступа к серверу через IP(это уникальный номер устройства, подключенного к интернету или локальной сети, чаще всего компьютера, сервера или мобильного гаджета).

7. Мониторинг и поддержка

- Использование инструментов мониторинга (Prometheus, Grafana) для отслеживания состояния системы.

8. Заключение

Архитектура системы управления задачами обеспечивает гибкость, масштабируемость и безопасность. Каждый уровень интегрирован таким образом, чтобы обеспечить бесперебойное выполнение всех функций системы и удобство для пользователей.

Внутреннее устройство модуля управления задачами

1. Введение

Модуль уведомлений отвечает за управления задачами. Он позволяет организовывать все задачи в удобном виде, расставлять приоритеты, контролировать сроки выполнения и управлять загруженностью сотрудников или своей собственной.

2. Основные функции модуля

1. Формирование задач:
   - Создание, удаление, обновление задачи.

3. Компоненты модуля

3.1. Генератор задач

Формирует тексты уведомлений:
- Использует текстовые шаблоны, в которых подставляются цель пользователя, приоритет и ее дата планирования.
- Пример шаблона: "{Приоритет} - {Цель} Бегать по утрам {Дата}."