Требования к проекту

1 Введение

Название продукта: AdventureTime Management System.
Контекст проекта: AdventureTime Management System — это веб-приложение, разработанное для управления туристическими данными, включая туры, страны и транспортные средства. Продукт предназначен для упрощения управления туристическими предложениями, позволяя администраторам добавлять, редактировать, удалять и связывать сущности (например, страны с турами и транспорт с турами) через интуитивно понятный интерфейс. Продукт будет взаимодействовать с RESTful API, предоставляемым сервером на базе Spring Boot, для обработки данных в реальном времени.
Что продукт будет делать:

- Предоставлять графический интерфейс для CRUD-операций (создание, чтение, обновление, удаление) над сущностями: туры, страны, транспорт.
- Поддерживать управление связями между сущностями (например, добавление/удаление стран в турах, установка/удаление транспорта для туров).
- Обеспечивать визуализацию данных с анимациями и модальными окнами для удобного взаимодействия.

Чего продукт делать не будет:

- Не будет поддерживать авторизацию и аутентификацию пользователей (предполагается, что доступ ограничен на уровне серверной конфигурации).
- Не будет предоставлять функции бронирования туров или финансовых операций.
- Не будет поддерживать мультиязычность или мобильные приложения на данном этапе.

2 Требования пользователя

2.1 Программные интерфейсы

- **Внешние системы**: Продукт взаимодействует с RESTful API, предоставляемым сервером на базе Spring Boot, запущенным на http://localhost:8080.
- **Библиотеки/сервисы**:
  - **React**: Используется для создания фронтенд-интерфейса с динамическими компонентами.
  - **React Router DOM**: Для маршрутизации между страницами (туры, страны, транспорт).
  - **Axios**: Для выполнения HTTP-запросов к API.
  - **React Bootstrap**: Для стилизации и создания модальных окон с анимациями.
  - **React Icons**: Для добавления иконок к кнопкам.

2.2 Интерфейс пользователя

Описание: Система предоставляет веб-интерфейс с навигационной панелью для переключения между разделами (туры, страны, транспорт). Каждый раздел включает компактные кнопки с иконками (добавить, редактировать, удалить) и модальные окна для добавления/редактирования данных. Интерфейс использует анимации (например, подъем элементов при наведении) и современный дизайн с тенями и закругленными углами.

Таблица "Действие пользователя - Реакция":

Действие пользователя	Реакция системы
Нажатие кнопки "Добавить тур"	Открывается модальное окно для ввода данных
Ввод данных и нажатие "Добавить"	Добавление тура через API, обновление списка
Нажатие "Редактировать"	Открывается модальное окно с данными тура
Изменение данных и "Сохранить"	Обновление тура через API, закрытие модалки
Нажатие "Удалить"	Удаление тура через API, обновление списка
Выбор страны и "Добавить страну"	Добавление страны к туру, обновление списка
Удаление страны из тура	Удаление связи через API, обновление списка

Действие пользователя Реакция системы

2.3 Характеристики пользователей

- **Администраторы**:
  - Уровень образования: Среднее специальное или высшее (в области IT или туризма).
  - Опыт: Базовые знания работы с веб-приложениями и REST API.
  - Техническая грамотность: Уверенные пользователи ПК, способные работать с интерфейсами и понимать структуру данных (туры, страны, транспорт).
- **Разработчики (при необходимости)**:
  - Уровень образования: Высшее (IT).
  - Опыт: Знание React, JavaScript, Spring Boot.
  - Техническая грамотность: Высокая, включая навыки отладки и настройки API.
  -

2.4 Предположения и зависимости

- Предполагается, что сервер Spring Boot работает и предоставляет корректные эндпоинты (/tours, /countries, /transport, и т.д.).
- Требуется стабильное интернет-соединение для взаимодействия с API.
- Зависит от правильной конфигурации CORS на сервере для доступа с http://localhost:3000.
- Актуальность данных зависит от своевременного обновления бэкенда.

3 Системные требования

3.1 Функциональные требования

1. **FR-001**: Система должна позволять добавлять новые туры через форму с полями "Название", "Описание" и "Длительность (дни)".
2. **FR-002**: Система должна позволять редактировать существующие туры через модальное окно с теми же полями.
3. **FR-003**: Система должна поддерживать удаление туров по идентификатору.
4. **FR-004**: Система должна отображать список туров с информацией о связанных странах и транспорте.
5. **FR-005**: Система должна позволять добавлять и удалять страны из туров через выпадающий список и кнопки.
6. **FR-006**: Система должна поддерживать установку и удаление транспорта для туров через выпадающий список.
7. **FR-007**: Система должна обновлять данные в реальном времени после выполнения CRUD-операций.
8. **FR-008**: Система должна предоставлять навигацию между разделами (туры, страны, транспорт) через меню.

3.2 Нефункциональные требования

3.2.1 Атрибуты качества

- **Надёжность**:
  - **Важность**: Обеспечение стабильной работы при частых запросах к API.
  - **Измерение**: Система должна выдерживать не менее 100 запросов в минуту без сбоев (тестирование нагрузкой).
- **Простота использования**:
  - **Важность**: Интерфейс должен быть интуитивно понятен администраторам с минимальным обучением.
  - **Измерение**: Время, необходимое для выполнения базовой задачи (например, добавление тура), не должно превышать 30 секунд для нового пользователя.
- **Производительность**:
  - **Важность**: Быстрая загрузка данных для обеспечения комфортной работы.

    - **Измерение**: Время ответа на запрос к API не должно превышать 2 секунды (тестирование с использованием инструментов вроде Postman).
- **Безопасность**:
    - **Важность**: Защита данных от несанкционированного доступа (хотя авторизация не реализована на фронтенде).
    - **Измерение**: Данные должны передаваться через HTTPS, что зависит от серверной конфигурации.
- **Адаптивность**:
    - **Важность**: Поддержка работы на разных разрешениях экрана (настольные ПК).
    - **Измерение**: Интерфейс должен корректно отображаться на разрешениях от 1280x720 до 1920x1080 пикселей.