Техническое руководство по эксплуатации Flexberry Next

1. Введение

Настоящий документ содержит информацию, необходимую для эксплуатации веб-приложений, разработанных на основе модуля **Flexberry Next**.

Flexberry Next – это клиентский SPA-фреймворк на базе Next.js, предоставляющий набор готовых компонентов, шаблонов и сервисов для создания современных веб-приложений для бизнеса.

2. Ключевые концепции и архитектура приложения

Приложение на Flexberry Next работает в рамках следующей клиентской архитектуры:

1. **Рендеринг:** поддерживается как **клиентский рендеринг (CSR)**, так и **серверный рендеринг (SSR)** и **статическая генерация (SSG)** Next.js, что позволяет оптимизировать производительность и SEO.
2. **Маршрутизация:** используется файловая маршрутизация Next.js на основе структуры папок pages/ или app/ (в зависимости от версии), а также встроенный маршрутизатор для клиентских переходов.
3. **Состояние:** Управление состоянием интерфейса может осуществляться как через встроенные React-механизмы (хуки, Context API), так и с помощью сторонних менеджеров состояний (подобных MobX), интегрированных в шаблон проекта.
4. **Доступ к данным:** клиент взаимодействует с backend через **RESTful API**. Для работы с данными используется клиентский слой Flexberry, предоставляющий единый интерфейс для выполнения операций чтения, создания, обновления и удаления (CRUD).

3. Работа с основными компонентами пользовательского интерфейса

Приложение использует стандартный набор компонентов Flexberry для отображения и редактирования данных.

3.1. Формы редактирования объектов (EditForm)

- **Назначение:** создание и редактирование экземпляров бизнес-объектов.
- **Эксплуатация:** поля формы автоматически сгенерированы на основе метаданных. Поля валидируются в соответствии с заданными типами (обязательное, email, диапазон и т.д.).
- **Действия:** на форме доступны кнопки «Сохранить» (отправляет данные на сервер), «Закрыть» (возврат к списку).

3.2. Компонент списка (ListForm)

- **Назначение:** Отображение объектов в табличном виде.
- **Эксплуатация:**
    - **Сортировка:** щелчок по заголовку столбца.
    - **Фильтрация:** использование строки поиска или настраиваемых фильтров в заголовках столбцов (где предусмотрено).
    - **Пагинация:** навигация по страницам списка через элементы управления внизу таблицы.
    - **Действия над записями:** для перехода к редактированию используется ссылка/кнопка в строке таблицы или щелчок по строке.

3.3. Компонент выбора из справочника (Lookup)

- **Назначение:** выбор значения из связанного списка (справочника).

- **Эксплуатация:** в поле ввода доступна кнопка с многоточием (…). При нажатии открывается модальное окно со списком доступных значений. После выбора строки и подтверждения, в поле подставляется текстовое представление выбранного объекта (например, наименование), а в модель данных записывается его ключ.

3.4. Компонент редактирования детейлов (GroupEdit)

- **Назначение:** работа с коллекциями зависимых объектов (например, строки заказа в составе заказа).
- **Эксплуатация:** встраивается в форму мастера. Позволяет добавлять, редактировать и удалять строки детейла. Данные обычно редактируются во встроенной таблице или отдельной модальной форме.

---

4. Взаимодействие с Backend (RESTful API)

1. **Эндпоинты:** backend предоставляет структурированные REST API эндпоинты для каждой сущности (например, GET /api/customers, POST /api/orders).
2. **Авторизация:** клиент автоматически добавляет токен авторизации (например, JWT) в заголовок Authorization каждого запроса. Механизм обновления токена реализован прозрачно для пользователя.
3. **Обработка ошибок:** ошибки от сервера (валидация, 404, 500) перехватываются единым обработчиком и отображаются пользователю в унифицированном виде (например, уведомление в углу экрана).
4. **Кеширование и состояние:** данные, полученные с сервера, могут кешироваться на клиенте для уменьшения числа запросов и повышения отзывчивости интерфейса.

5. Настройки и конфигурация

Конфигурационные параметры приложения на этапе эксплуатации могут быть изменены через:

1. **Файлы окружения (Environment Variables):**
    - .env.local, .env.production – содержат переменные, специфичные для среды (URL бэкенда, ключи внешних сервисов, режим отладки).
    - **Ключевые параметры:** NEXT_PUBLIC_BACKEND_ROOT_URL (базовый адрес API), NEXT_PUBLIC_APP_VERSION и другие.
2. **Параметры сборки (Build-time Configuration):** некоторые настройки (например, выбор темы по умолчанию) задаются на этапе сборки приложения командой npm run build.

6. Локализация и темы оформления

1. **Локализация (i18n):**
    - Интерфейс поддерживает переключение языков.
    - Текстовые ресурсы хранятся в JSON-файлах в структуре типа locales/ru/common.json.
    - Текущий язык может переключаться через компонент выбора в интерфейсе или автоматически определяться по браузеру.
2. **Темы оформления:**
    - Поддерживается механизм смены цветовых схем (например, светлая/темная).
    - Тема применяется динамически через CSS-переменные или контекст React.
    - Выбор темы может сохраняться в локальном хранилище браузера (localStorage).

---

7. Расширение функциональности (для разработчиков)

1. **Добавление новой страницы:** создание нового файла .tsx в директории pages/ (или app/) в соответствии с правилами маршрутизации Next.js.
2. **Создание нового компонента:** разработка React-компонента с использованием как нативных элементов, так и готовых компонентов Flexberry (поля, кнопки).
3. **Интеграция нового API:** для вызова нового эндпоинта используется стандартный клиент fetch или обертка axios, настроенная в проекте. Рекомендуется размещать логику вызовов в отдельных сервисных модулях (services/).

8. Контакты для получения технической поддержки

- По вопросам приобретения лицензий и технической помощи обращайтесь по адресу support@flexberry.net.