Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Создание веб-приложения для контроля и учета крупного рогатого скота»

по дисциплине «Проектный практикум»

Куратор: Шестеров М. А. _____

Студенты команды Реактор

Вершинина О. А. _____

Кочергин М. А. _____

Кочнев С. В. _____

Михайлов Д. В. _____

Топор А. Д. _____

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1.1 Цель и задачи проекта

Целью проекта является разработка специализированного веб-приложения Cattle-Track для автоматизации контроля и учета крупного рогатого скота (КРС) в мясных скотоводческих хозяйствах.

Основные задачи проекта:

- создание интуитивно понятного интерфейса, адаптированного под пользователей с разным уровнем цифровой грамотности,
- реализация функциональных модулей для учета поголовья, репродуктивных событий, ветеринарных обработок, кормления и динамики привесов,
- обеспечение автоматизированной генерации отчетов в соответствии с отраслевыми стандартами,
- интеграция аналитических инструментов для прогнозирования сроков откорма и оптимизации производственных процессов.

Актуальность проекта

Актуальность разработки обусловлена следующими факторами:

- неэффективность ручного учета: большинство фермерских хозяйств по-прежнему используют бумажные журналы или таблицы excel, что приводит к ошибкам, потере данных и усложнению аналитики,
- дефицит специализированных решений: существующие на рынке программные продукты ориентированы преимущественно на молочное скотоводство, тогда как потребности мясных ферм остаются неохваченными,

– рост требований к стандартизации: ужесточение норм ветеринарного и производственного контроля требует внедрения цифровых систем учета, обеспечивающих прозрачность и достоверность данных.

Разработка Cattle-Track позволит устранить эти проблемы, повысив эффективность управления скотоводческими предприятиями.

Область применения программного продукта

Программный продукт предназначен для использования в следующих сферах:

– фермерские хозяйства (малые, средние и крупные) – для автоматизации учета поголовья, контроля здоровья животных и планирования ресурсов,
– ветеринарные службы – для мониторинга вакцинаций, диагностики и лечения скота,
– агрохолдинги – для централизованного управления данными с предприятий.

Приложение также может быть интегрировано с внешними системами (например, ERP или бухгалтерскими программами) для расширения функциональных возможностей.

Ожидаемые результаты и планируемые достижения

По завершении проекта ожидается:

– снижение трудозатрат на ведение учета за счет автоматизации рутинных операций,
– минимизация ошибок при фиксации данных и формировании отчетности,

- улучшение контроля за здоровьем скота за счет своевременной фиксации информации о вакцинациях и заболеваниях,
- повышение рентабельности производства благодаря аналитике кормления и динамики привесов.

После завершения этапа разработки всех модулей будет проведено масштабное тестирование с последующей доработкой продукта на основе обратной связи от пользователей.

Основная часть

Анализ требований заказчика и формирование бэклога

Выявление ключевых требований

В ходе предварительного этапа разработки был проведен детальный анализ требований заказчика, компании Cattle Expert, а также целевой аудитории, включающей фермеров, ветеринаров, зоотехников и управляющих хозяйствами. Основной целью являлось создание системы, способной автоматизировать процессы учета крупного рогатого скота, повысить точность данных, обеспечить своевременное выполнение ветеринарных мероприятий и упростить формирование отчетности.

На основании собранных данных был сформирован список ключевых требований к системе, включающий:

- регистрацию новых животных с учетом их идентификационных данных,
- ведение учета всех животных на предприятии,
- модуль репродуктивного учета (осеменения, проверки, отёлы),
- учет ежедневных ветеринарных и хозяйственных действий,
- контроль кормления и анализ эффективности откорма,
- генерацию отчетов.

Для уточнения потребностей пользователей проведены интервью с представителями целевых групп (фермеры, зоотехники, ветеринары). Выявлены дополнительные пожелания:

- мобильная адаптация интерфейса (для работы в полевых условиях),
- возможность кастомизации полей (например, добавление дополнительных способов идентификации животного),
- интуитивный UX/UI-дизайн для работы пользователей с низкой цифровой грамотностью.

Составление бэклога

Для реализации поставленных задач был составлен backlog проекта, включающий следующие этапы:

- разработка технического задания (тз) для каждого модуля;
- проектирование интерфейсов пользователя (UX/UI дизайн);
- создание архитектуры базы данных Postgresql с учетом хранения всех необходимых данных;
- реализация фронтенд-части на React с учетом адаптивности и удобства использования;
- разработка бэкенд-логики на C# с обеспечением взаимодействия с базой данных;
- проведение тестирования модулей внутри команды разработчиков;
- внедрение системы на пилотных фермах для получения обратной связи;
- итеративное доработки системы на основе полученных отзывов.

Анализ аналогов и конкурентных решений

Проведенный бенчмаркинг (таблица 1) показал, что существует небольшое количество специализированных решений для мясных ферм – так как большинство приложений заточено под молочное направление.

Таблица 1. Анализ аналогов

Критерий	Cattle Track	СЕЛЭКС	Cattlemax	Breedr
Адаптация на русский язык	+	+	-	-
Интуитивно понятный UX/UI-дизайн	+	-	+	+

Наличие Веб-версии приложения	+	-	+	+
Модуль репродуктивного учета	+	-	+	+
Модуль учета ежедневных(ветеринарных) действий	+	-	+	+
Модуль генерации отчетности	+	+	-	-
Возможность импортировать животных из других источников	+	-	-	-

На российском рынке существует единственный аналог приложения ("Селэкс"), который обладает базовым функционалом по формированию стандартных отчетов. Однако данный продукт характеризуется устаревшим интерфейсом и недостаточной гибкостью в настройке под конкретные потребности мясных хозяйств.

На основе анализа зарубежных решений были заимствованы идеи по реализации модулей аналитики, учета репродуктивных процессов и кормления. В результате было принято решение о создании собственного программного продукта с учетом специфики мясного скотоводства в России, что позволяет обеспечить более точное соответствие требованиям пользователей и нормативным актам.

Архитектура программного продукта

Выбор технологического стека

Разрабатываемое веб-приложение построено по многоуровневой архитектуре, что обеспечивает его масштабируемость, надежность и удобство сопровождения. Основные компоненты системы включают:

Frontend — реализован на React с использованием современных библиотек (например, AntDesign) для создания интуитивно понятного пользовательского интерфейса. Такой подход обеспечивает кроссплатформенную адаптивность и удобство работы как на настольных компьютерах, так и на мобильных устройствах.

Backend — реализован на C# с использованием ASP.NET Core framework. Он отвечает за обработку бизнес-логики, взаимодействие с базой данных PostgreSQL, а также обеспечение безопасности системы.

База данных — использует PostgreSQL для хранения всей информации о животных, операциях, мероприятиях и отчетах. Структура базы данных проектировалась с учетом нормализации данных для повышения эффективности запросов и обеспечения целостности информации.

Связь между компонентами осуществляется через REST API-интерфейсы, что позволяет легко расширять функциональность системы и интегрировать ее с внешними сервисами при необходимости (рисунок 1).

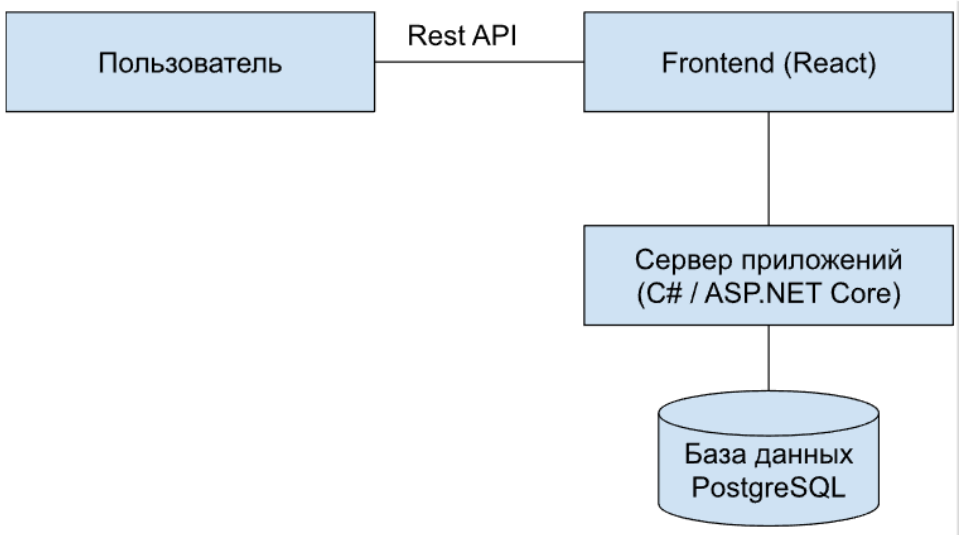

Рисунок 1. Архитектура сервиса

Обоснование выбора архитектурного решения основано на необходимости обеспечения высокой производительности при работе с большим объемом данных, а также возможности дальнейшего масштабирования системы по мере роста числа пользователей и расширения функционала. Использование современных технологий фронтенда и бэкенда способствует созданию удобного в использовании продукта с минимальными требованиями к аппаратным ресурсам конечных пользователей.

Сценарий использования приложения

Для понимания логики взаимодействия пользователя с приложением был разработан сценарий использования:

1) авторизация:
    a. пользователь вводит логин и пароль.
2) регистрация животных:
    a. *ручной ввод*: открывается форма с полями (метка, порода, фото и т.д.).
    b. *импорт*: загрузка файла (поддержка валидации формата).
3) учет животных и карточка животного:
    a. просмотр списка с фильтрацией и сортировкой,
    b. редактирование карточки (например, добавление родителя).
4) ежедневные действия:
   пример: вакцинация → выбор животного → дата → препарат.
5) репродуктивный учет:
    a. фиксация событий:
        i. осеменение → привязка к быку-производителю,
        ii. отел → автоматическое создание записи о теленке.
6) контроль привесов + кормление:
    a. связь модулей: вес → расчет норм корма → график привесов.
7) отчеты:

a. выгрузка отчётов.

На рисунке 2 представлена диаграмма для описания взаимодействия пользователя с приложением.

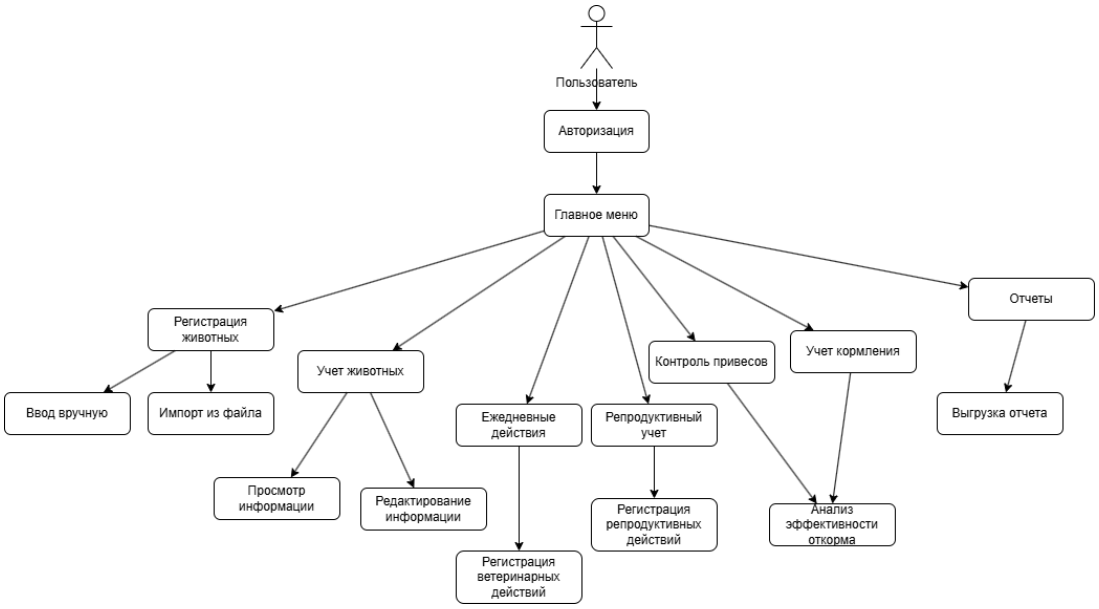

Рисунок 2. Сценарий использования приложения

Отчет о процессе разработки приложения Cattle-Track

Методология разработки

Использованная методология: Гибридная (Scrum + Kanban)

Scrum для итеративной разработки (спринты по 2 недели).

Kanban для визуализации задач (доска в Jira).

Ключевые этапы:

1) подготовка:
    a. сбор требований, создание бэклога, проектирование архитектуры.
2) разработка:
    a. реализация модулей по приоритетам в рамках разных версий приложения (от 0.0.1 до 0.3.0).
3) тестирование:

a. промежуточные тесты после каждого релиза.

4) внедрение:
    a. пилотное тестирование на фермах, сбор обратной связи.
5) доработка:
    a. доработка функционала приложения после получения обратной связи.

Роли команды:

- тимлид: координация, аналитика,
- разработчики: 2 бэкенд (c#), 2 фронтенд (react),
- тестировщик: проверка функционала, баг-репорты.

Процесс разработки

Процесс разработки приложения был разделен на модули (таблица 2) с указанием сроков разработки и ответственных за разработку модулей Frontend и Backend разработчиков.

Таблица 2. Процесс разработки по модулям

Страница	Описание	Срок	Ответственные
Регистрация животных	Внесение данных о новом животном	2 нед	Топор Арина, Михайлов Данил
Учет животных	Общий реестр, фильтрация, поиск, редактирование	2 нед	Кочнев Сергей, Кочергин Михаил
Инфраструктура	Управление группами и доп. полями	1 нед	Топор Арина, Михайлов Данил
Репродуктивный учет	Фиксация осеменений, диагностик, отелов	2 нед	Топор Арина, Михайлов Данил
Ежедневные действия	Вакцинации, обработки, перемещения. лечение	2 нед	Кочнев Сергей, Кочергин Михаил
Главная	Дашборд с ключевыми метриками	1 нед	Топор Арина, Михайлов Данил

Контроль привесов	Анализ динамики роста, эффективности откорма	1 нед	Кочнев Сергей, Кочергин Михаил
Учет кормления	Расчет рационов, контроль расходов	1 нед	Кочнев Сергей, Кочергин Михаил
Карточка животного	Детальная информация по каждой особи	1 нед	Топор Арина, Михайлов Данил
Отчеты	Автоматическая генерация документов	1 нед	Кочнев Сергей, Кочергин Михаил

Весь процесс работы над проектом, включая разработку, был разбит по этапам (таблица 3).

Таблица 3. Процесс работы над проектом по этапам

Этап	Действия	Сроки
Анализ	Интервью с заказчиками, бенчмаркинг	2 недели
Прототипирование	Создание ТЗ, дизайн UI в Figma, согласование с заказчиком	1 неделя
Бэкенд-разработка	Реализация API (C#), миграции PostgreSQL	8 недель
Фронтенд	Верстка (React), интеграция с API	8 недель
Тестирование	Юнит-тесты, ручные тесты, исправление багов, тестирование пользователями	2 недели
Доработка	Возможная доработка приложения после получения обратной связи от пользователей.	2 недели

Отчеты членов команды команды

Вершинина Ольга (Тимлид и Аналитик)

На начальном этапе проекта мною был проведен комплексный анализ существующих аналогов и изучены потребности заказчиков. В рамках Customer Development (CustDev) проведены интервью с потенциальными пользователями, что позволило сформировать четкие требования к функционалу системы.

На основе собранных данных:

- разработана дорожная карта (roadmap) проекта с определением ключевых этапов разработки,
- составлено техническое задание (тз) для дизайнера и разработчиков, включающее функциональные и нефункциональные требования,
- проведены согласования с заказчиками для утверждения требований и сроков реализации.

В процессе работы выполнялись следующие задачи:

Координация дизайна: Контроль этапов создания UI/UX-дизайна (с привлечением аутсорс-дизайнера), включая вайрфреймы, прототипирование и финальные макеты.

Декомпозиция задач: Разбиение функциональных требований на подзадачи для разработчиков с использованием системы управления проектами (Jira).

Контроль исполнения: Мониторинг выполнения задач, проведение код-ревью и устранение блокеров.

Тестирование и обратная связь: После завершения разработки модулей организовывалось их тестирование реальными пользователями (владельцами ферм). На основе их фидбека проводились доработки системы.

В рамках проекта была спроектирована и реализована реляционная база данных на PostgreSQL, размещенная на удаленном сервере.

Основные выполненные работы:

1) проектирование схемы БД (рисунок 3): определение сущностей, связей, индексов и ограничений для обеспечения оптимальной производительности.
2) реализация хранимых процедур и функций: разработаны Pl/PgSQL-функции и триггеры, что позволило:
    a. повысить производительность: выполнение сложных вычислений на стороне БД снизило объем передаваемых данных между сервером приложений и базой,
    b. централизовать бизнес-логику: критические операции (агрегация данных, валидации) выполняются непосредственно в БД, минимизируя риски рассинхронизации,
    c. обеспечить безопасность: настроены роли и права доступа, ограничивающие несанкционированные операции,
    d. гарантировать целостность данных: триггеры автоматически проверяют корректность данных при изменениях,
    e. снизить нагрузку на бэкенд: оптимизированные запросы и использование индексов ускорили обработку данных.
3) Миграции и управление версиями: Реализована система миграций для поддержания актуальности схемы БД.

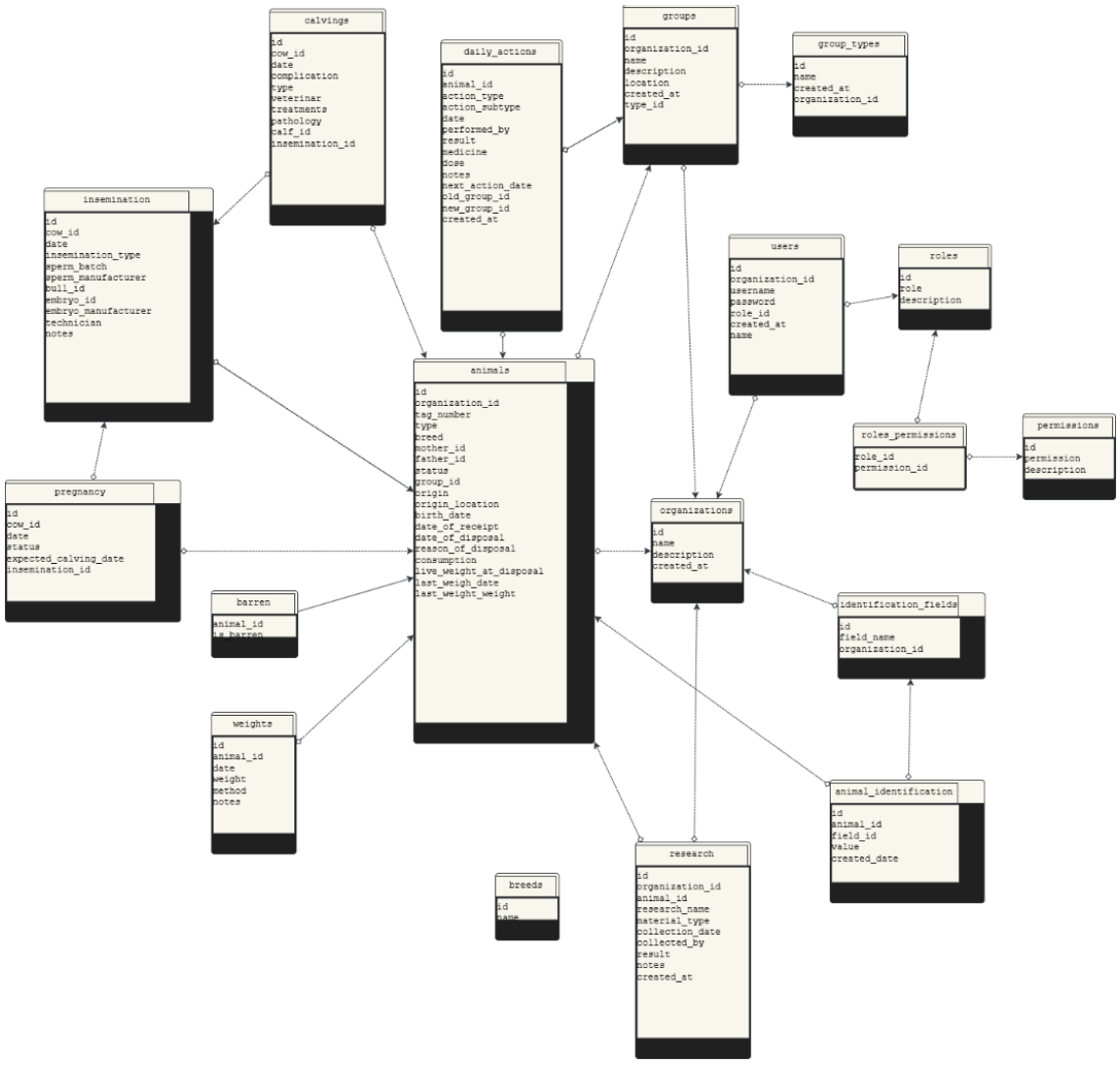

Рисунок 3. Диаграмма архитектуры базы данных

В результате проделанной работы:

- успешно проведен анализ требований и согласован проект с заказчиками,
- организован процесс разработки, включая контроль дизайна и декомпозицию задач,
- разработана эффективная архитектура бд с оптимизированными хранимыми процедурами,
- обеспечено тестирование системы с последующей доработкой на основе пользовательской обратной связи.

Дальнейшие действия включают:

- масштабирование системы с учетом новых требований,
- дальнейшую оптимизацию запросов и мониторинг производительности бд,
- доработка модулей по запросу заказчиков.

Отчет Кочергин Михаил (Backend разработчик и тестировщик)

Во время разработки использовался следующий стек:

- инструменты разработки: C#, ASP.NET, postgresql, ORM Entity Framework, Nginx

- система контроля версий: Git

- инструменты документации: OpenApi Swagger

- инструменты тестирования: Postman, DevTools

- инструменты сборки и доставки: Docker, docker-compose

Процесс разработки делился на версии, в рамках которых реализовывался отдельный функционал приложения. Для удобства был налажен GitFlow. С каждой версией приложение собиралось и доставлялось на сервер. Перед релизом новых версий проводилось функциональное тестирование с использованием техник тест-дизайна.

В версии 0.0.1 была произведена инициализация backend-сервера, настроена ORM с привязкой модели к созданной базе данных, настроена документация Swagger. Реализована авторизация на Cookie, разделение прав пользователей. Написана логика и rest-запросы модуля Учёт животных. Перед деплоем настроен Nginx и написаны docker-файлы для бэкенда и фронтенда и общий docker-compose файл.

В версии 0.2.0 в связи с изменениями в требованиях была дописана логика модуля Учёт животных. В связи с изменениями структуры БД менялся код на бэке. Написана логика модуля Ежедневные действия. Проводилось тестирование вышеперечисленных модулей, а также модулей Регистрация

животных, Инфраструктура, Репродуктивный учёт. На сервер установлены подписанные сертификаты.

В патч-версии 0.2.1 исправлялись баги и вносились минорные правки по уже реализованным модулям.

Отчет Михайлов Данил (Backend разработчик)

В течение текущего учебного семестра я выполнял функции backend-разработчика в составе команды.

Основные достижения включали разработку контроллеров для обработки HTTP-запросов, а также реализацию POST- и GET-методов API для следующих модулей: регистрация животных, инфраструктура и репродуктивный учёт.

Проект разработан с использованием фреймворка ASP.NET и языка программирования C#. В процессе разработки применялись следующие библиотеки: EntityFrameworkCore (для работы с базой данных), Npgsql (для взаимодействия с PostgreSQL) и CSVHelper (для обработки CSV-файлов).

Мной были реализованы следующие контроллеры:

- **AnimalsController** – обеспечивает базовые операции регистрации, удаления и модификации данных о животных;
- **GroupsController** – предоставляет CRUD-функционал для работы с группами животных;
- **ReproductiveController** – содержит методы, позволяющие осуществлять полный цикл репродуктивных процессов, включая осеменение и отёл.

Наиболее сложной задачей стала реализация импорта данных о животных из платформы **СЕЛЕКС** в формате CSV. Процесс потребовал многоэтапной валидации данных для обеспечения корректного сохранения связей между сущностями и соблюдения требуемого формата. Для гарантии

целостности данных весь алгоритм был инкапсулирован в транзакцию, что исключило возможность конфликтов при повторной загрузке файла.

Также значительные трудности вызвала разработка модуля репродуктивного учёта из-за его сложной бизнес-логики, включающей множество валидаций и проверок данных на различных этапах обработки.

Отчёт Топор Арина (Frontend разработчик)

На начальном этапе разработки приоритетной задачей стало внедрение стабильного и прозрачного процесса управления версиями. Для этого была реализована стратегия GitFlow, которая обеспечивает структурированное ведение истории изменений, облегчает проведение тестирования и позволяет нескольким разработчикам параллельно работать над различными функциональными блоками проекта.

Была предложена структура эпиков и подзадач, ориентированная на ключевые бизнес-функции системы. Это позволило команде синхронизировать планирование и реализацию задач, а также упростить процесс контроля исполнения спринтов. В дальнейшем данный подход стал стандартом в нашей команде.

Учитывая планы по разработке мобильного приложения, было принято решение использовать связку Ionic + Capacitor, которая обеспечивает кроссплатформенную совместимость и позволяет использовать общую кодовую базу как для web-, так и для mobile-версий.

Для управления состоянием приложения и взаимодействия с сервером был выбран RTK Query — решение на базе Redux Toolkit, которое существенно упрощает реализацию REST-запросов и кэширование данных.

Для обеспечения единообразного интерфейса и ускорения разработки UI-части совместно с дизайнером было принято решение использовать Ant Design UI Kit. Он предоставляет обширный набор готовых компонентов, легко

кастомизируется и хорошо документирован, что позволяет быстро внедрять изменения и поддерживать согласованность в дизайне.

Учитывая однородную структуру пользовательского интерфейса, одной из первоочередных задач стала реализация универсального layout-компонента, который будет использоваться на всех страницах приложения.

В процессе тестирования было выявлено, что при добавлении или удалении полей идентификации на странице Регистрации животных данные в интерфейсе обновлялись не мгновенно. Баг был оперативно устранён, однако этот случай послужил отправной точкой к переосмыслению логики обновления данных: в каких случаях следует перезапрашивать их с сервера.

На этапе тестирования модуль Регистрация животных показал высокую стабильность — минимальное количество багов по сравнению с другими разделами. Тем не менее, после выхода версии 0.2.0 заказчик сформулировал ряд дополнительных требований, которые были доработаны.

Модуль репродуктивного учета оказался самым требовательным в разработке, так как исходные требования от заказчика не прошли достаточной валидации на реальных данных. В процессе реализации выявлялось множество уточняющих сценариев и кейсов, что потребовало гибкости в проектировании формы и логики отображения.

В ходе проекта была проведена комплексная работа по созданию и развитию системы для ветеринарного учёта, включающей регистрацию животных и репродуктивный учёт. В процессе были решены задачи на разных уровнях — от инфраструктурных аспектов до глубокой бизнес-логики.

Отчёт Кочнев Сергей (Frontend разработчик)

Перед началом разработки была проведена настройка базовой инфраструктуры проекта.

Проект был инициализирован с использованием Vite для быстрой настройки окружения с поддержкой TypeScript.

Были добавлены ключевые библиотеки, а также проведена базовая настройка:

- конфигурация роутинга,
- настройка хранилища Redux с RTK Query,
- подключение стилей и темы Ant Design.

Авторизацию было решено сделать с использованием HTTP-Only Cookies. Основная цель — защита от XSS-атак и упрощение управления сессиями. Форма авторизации была вынесена в отдельный компонент, где была описана логика отправки данных на сервер и обработка полученного ответа. На этапе тестирования, в данном модуле не было выявлено багов.

Модуль "Учет животных" включает следующие разделы: Коровы; Нетели; Быки; Телки; Бычки. Каждый раздел включал в себя таблицу с информацией о конкретном типе животных. Таблица имеет возможность редактирования полей, сортировки животных по всем столбцам. Для функции редактирования были созданы кастомные поля, в которых при двойном нажатии открывается ввод или выбор нового значения. В процессе тестирования было выявлено, что при переходе в другой раздел не обновлялся номер текущей страницы таблицы. Баг был оперативно устранен.

Модуль «Ежедневные действия» включает в себя следующие ключевые разделы: Осмотры; Вакцинации и обработки; Лечение; Перевод; Выбытие; Исследования; Присвоение номеров. Каждый из разделов включает в себя:

- фильтрация и выбор животных,
- форма для создания ежедневного действия,
- история ежедневных действий по текущему разделу,
- фильтрация и выбор.

Для каждого раздела реализована своя форма, с конкретными полями.

История Ежедневых действия реализована в виде таблицы (рисунок 4) с возможностью сортировки действий. История также сделана общим

компонентом, в который передается номер раздела. По итогам тестирования были выявлены такие баги:

- в групповых действиях выбранные животные пропадали при пагинации,
- таблицы истории автоматически не сортировалась по дате,
- данные ошибки были успешно исправлены.

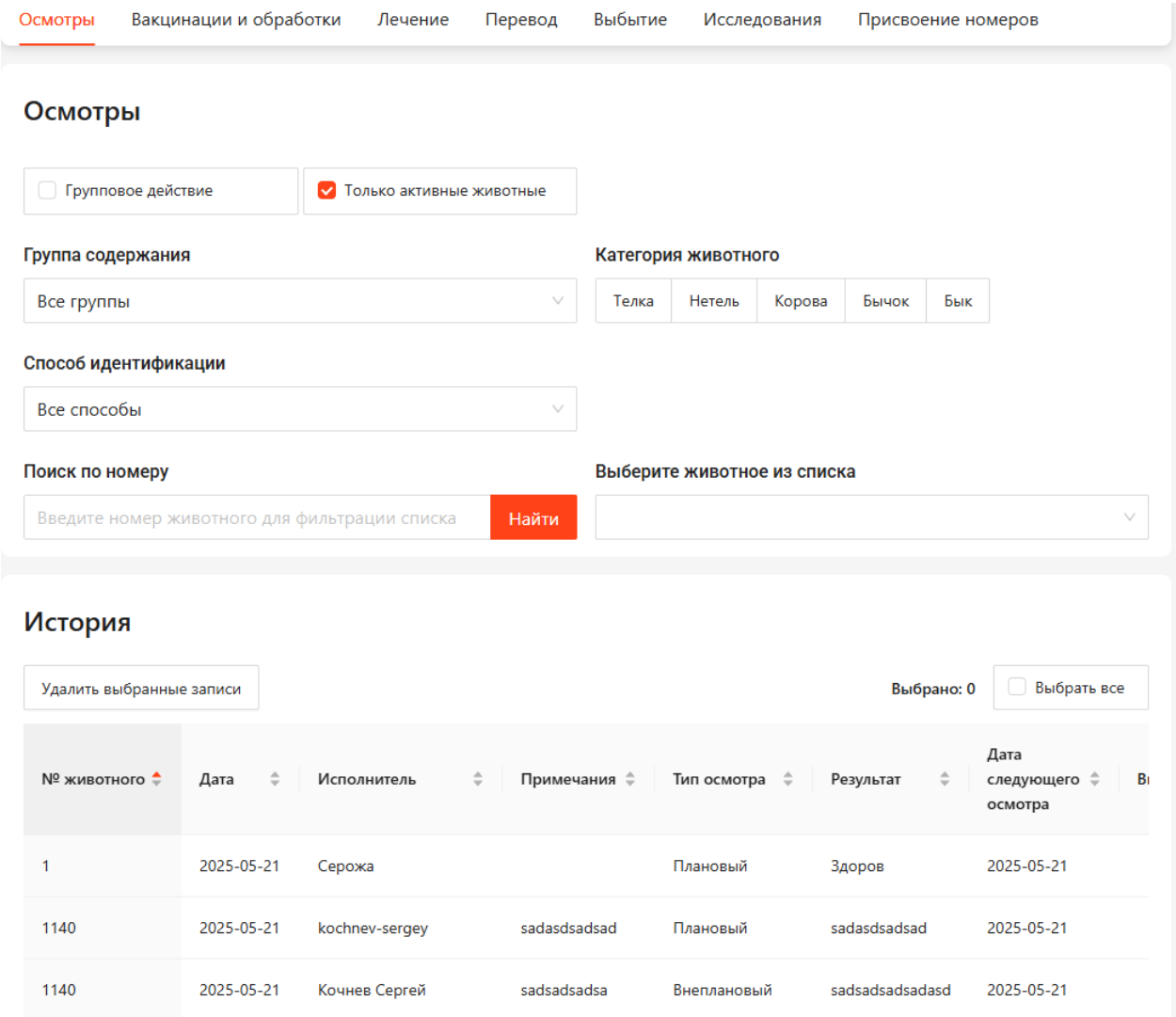

Рисунок 4. Пример интерфейса страницы Ежедневные действия

В рамках проекта была выполнена масштабная работа по разработке и совершенствованию ветеринарной учетной системы. Решение охватило полный цикл задач - от построения базовой инфраструктуры до реализации сложной бизнес-логики, включая функционал авторизации, учета животных и ежедневных действий.

Доработка модулей

В ходе пост-релизного тестирования приложения ключевыми заказчиками – владельцами сельскохозяйственных предприятий – была выявлена необходимость доработки функционала модуля репродуктивного учета. Данный модуль реализует сложный бизнес-процесс, охватывающий полный цикл репродуктивных операций, включая:

- регистрацию осеменений,
- мониторинг стельности (включая диагностические проверки),
- фиксацию отёлов и регистрацию телят.

Пользовательское тестирование позволило идентифицировать следующие недостатки первоначальной реализации:

- недостаточная детализация данных – отсутствовали элементы для присвоения статусов корове, прошедшей осеменение; отсутствие полей для регистрации мертворожденного телёнка.
- отсутствует взаимодействие с другими модулями – не учитывались регистрация Нетелей на странице Регистрации животных; отсутствие информации о Яловых коровах на странице Учета животных.

На основании обратной связи и дополнительного анализа предметной области были внесены следующие изменения:

- добавлена возможность регистрации мертворожденного телёнка,
- добавлена возможность присваивать корове статус «Яловая» и «Подлежит повторному осеменению»,
- добавлена интеграция с модулями Регистрации животных и Учета животных.

Научно-практическая значимость проведенных доработок:

- взаимодействие с конечными пользователями (зоотехниками и ветеринарами) позволило уточнить биологические и технологические нюансы репродуктивных процессов крс, что отразилось в алгоритмизации бизнес-логики,
- внедрение автоматизированных проверок снизило количество ошибок ввода и исключило регистрацию некорректных физиологических событий.

Проведенные доработки модуля репродуктивного учета демонстрируют итеративный подход к разработке ПО, при котором пользовательская обратная связь становится ключевым драйвером улучшений. Учет отраслевой специфики и биологических закономерностей позволил трансформировать первоначальную версию модуля в надежный инструмент для управления репродуктивными циклами в животноводстве.

ЗАКЛЮЧЕНИЕ

Соответствие программного продукта требованиям заказчика и пользователей

Разработанная система **Cattle-Track** в полной мере удовлетворяет ключевые требования заказчика и целевой аудитории, что подтверждается:

1) реализацией критически важного функционала:
    a. автоматизированный учет поголовья с поддержкой групповых операций и кастомизации полей,
    b. корректная работа модулей репродуктивного учета, ветеринарных обработок и контроля кормления, включая сложные бизнес-процессы (осеменение, диагностика стельности, отёлы),
    c. гибкая система отчетности, адаптированная под отраслевые стандарты.
2) учетом пользовательских предпочтений:
    a. интуитивный интерфейс (Ant Design) с мобильной адаптацией, что особенно важно для работы в полевых условиях,
    b. интеграция импорта данных из CSV (включая совместимость с системой "СЕЛЭКС"), упрощающая переход с legacy-решений.
3) соблюдением нормативных требований:
    a. поддержка обязательных идентификационных данных животных и ветеринарных протоколов.

Вывод: Продукт закрывает ключевые боли пользователей — заменяет бумажный учет, минимизирует ошибки ввода и обеспечивает прозрачность данных. Однако выявленная потребность в пост-релизных доработках (например, расширение статусов коров) подтверждает необходимость более глубокого вовлечения конечных пользователей на этапе проектирования сложных модулей.

Оценка качества продукта на основе тестирования

Результаты тестирования демонстрируют высокую надежность системы, но также выявляют зоны для оптимизации:

Успешно решенные проблемы:

- критические баги (например, пропадание выбранных животных при пагинации) были оперативно устранены, что свидетельствует об эффективности процессов QA.
- внедрение HTTP-Only Cookies и ролевой модели доступа обеспечило безопасность данных.

Остающиеся риски:

- сложные бизнес-процессы (репродуктивный учет) требуют дополнительных валидаций для исключения некорректных сценариев (например, регистрация мертворожденных телят),
- производительность: Нагрузочное тестирование не проводилось — потенциальные узкие места (например, работа с крупными CSV-файлами) могут проявиться при масштабировании.

Вывод: Система готова к промышленной эксплуатации, но требует:

- расширенного тестирования на реальных данных (особенно для модулей с высокой нагрузкой — учет кормления, привесов),
- мониторинга производительности после увеличения числа пользователей.

Предложения по улучшению и развитию продукта

Для повышения конкурентоспособности и удовлетворенности пользователей рекомендуется:

1) приоритетные доработки:
    a. развитие мобильного приложения (Ionic + Capacitor) для офлайн-работы в условиях слабого интернета,

b. внедрение искусственного интеллекта для прогнозирования сроков откорма на основе динамики привесов.

2) оптимизация процессов:
    a. автоматизация тестирования (например, Selenium для регрессионных проверок),
    b. переход на GraphQL для гибкости API (актуально при интеграции с ERP-системами).
3) стратегические инициативы:
    a. партнерства с ветеринарными клиниками для подключения лабораторных данных,
    b. разработка white-label версии для агрохолдингов с кастомизацией под бренд.

Рекомендации для будущих проектов:

- проводить демо-тестирование прототипов с фермерами на ранних этапах (особенно для модулей с высокой предметной спецификой).
- заложить в бюджет ресурсы на пост-релизную поддержку — практика показала, что 20% функционала требуют доработок после внедрения.

Итоговый вывод

Cattle-Track успешно решает задачи цифровизации мясного скотоводства, демонстрируя соответствие требованиям заказчика и устойчивость к реальным нагрузкам. Для завершения проекта необходимо завершить 4 модуля, после чего перейти к масштабному тестированию. Финальный деплой приложения планируется 12 июня, после чего будет происходить внедрение продукта. Дальнейшее развитие продукта должно фокусироваться на углубленной аналитике и экосистемной интеграции, что повысит его рыночную ценность. Проект подтвердил важность итеративного подхода с участием конечных пользователей на всех этапах жизненного цикла ПО.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Breedr: Case Study - Double M Ranch [Электронный ресурс]. – URL: https://www.breedr.co/case-studies/double-m-ranch (дата обращения: 04.03.2025).

2. CattleMax: Why CattleMax? [Электронный ресурс]. – URL: https://www.cattlemax.com/why-cattlemax (дата обращения: 04.03.2025).

3. Три этапа отёла коров // Своё фермерство [Электронный ресурс]. – URL: https://svoefermerstvo.ru/svoemedia/articles/tri-jetapa-otjola-korov (дата обращения 12.03.2025).

4. Взвешивание КРС: методы и оборудование // VESService [Электронный ресурс]. – URL: https://vesservice.com/company/blog/vzveshivanie-krs/?srsltid=AfmBOorE6-IiF0DAluWWVKejJ7Tx-CUkv7HDLPRkne-rO5KFxloy-PKz (дата обращения: 12.03.2025).

5. Кормление КРС молочных видов: нормы и добавки // Electropastyx [Электронный ресурс]. – URL: https://electropastyx.ru/blog/kormlenie-krs-molochnykh-vidov-zhivotnykh-normy-sutochnogo-pitaniya-i-kakie-kormovye-dobavki-vkhodyat-v-ratsion/ (дата обращения: 12.03.2025).

6. Мясные породы коров // Агропост [Электронный ресурс]. – URL: https://agropost.ru/skotovodstvo/myasnie-porodi-korov/ (дата обращения: 12.03.2025).

7. СелексБиф: автоматизация мясного скотоводства // Плинор [Электронный ресурс]. – URL: https://plinor.ru/selexbeef (дата обращения: 12.03.2025).

ПРИЛОЖЕНИЕ А

(обязательное)

Материалы для работы над приложением Cattle-track

Программный продукт Cattle-Track

- Официальный сайт: https://cattle-track.ru/

  Доступ ограничен (требуется авторизация Логин: localadmin Пароль: secure_password).

Дизайн-макеты интерфейса

- Figma: https://www.figma.com/design/l881m4UR8Hu0PvHBbDaF5B/%D0%A1attleTrack

  Доступ ограничен (требуется авторизация).

Исходный код

- Frontend

  (React): https://github.com/olyannaa/cattle_track_frontend

  Лицензия: MIT.

- Backend (C#): https://github.com/olyannaa/cattle_track_backend

  Лицензия: MIT.

Техническое задание (ТЗ)

- Google

  Docs: https://docs.google.com/document/d/12Uk0Wl7JWFXYC7ZtzizGj0Et17pmmmmxYBQrut0xeIY/edit

  Доступ по ссылке.

Система управления задачами (Jira)

- Проектная доска: https://vega-reactor.atlassian.net/jira/software/projects/CAT/boards/35

  Требуется авторизация в Atlassian.