МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

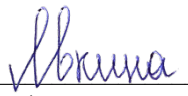
Кафедра Прикладная математика и информатика»
(наименование кафедры)

09.03.03 «Прикладная информатика»
(код и наименование направления подготовки, специальности)

Бизнес-информатика
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка программного обеспечения автоматизированной системы транспортной логистики»

Обучающийся А. О. Явкина
(И.О. Фамилия)

(личная подпись)

Руководитель Д.Г. Токарев
(И.О. Фамилия)
(личная подпись)

Тольятти 2024

Аннотация

Тема: «Разработка программного обеспечения автоматизированной системы транспортной логистики».

Объектом исследования является процесс разработки программного обеспечения для автоматизации транспортной логистики. В ходе исследования рассматриваются задачи создания эффективной системы управления транспортными потоками, а также разработки программной реализации на базе современных технологий.

Структура выпускной квалификационной работы (ВКР) представлена введением, тремя главами, заключением и списком литературы.

Во введении обосновывается актуальность исследования, выявляется потребность в разработке программного обеспечения для оптимизации процессов транспортной логистики.

Первая глава посвящена обзору существующих технологий и методов в области транспортной логистики, а также анализу требований к разрабатываемой системе. В результате проведенного анализа формулируются основные принципы и задачи, стоящие перед разработчиками.

Во второй главе проводится сравнение подходов и инструментов для разработки программного обеспечения автоматизированной системы транспортной логистики, описываются их преимущества и недостатки, и принимается решение о выборе стека технологий для разработки.

Третья глава посвящена проектированию архитектуры автоматизированной системы транспортной логистики, определению функциональности интерфейса пользователя, взаимодействия с внешними системами. Описывается процесс разработки и тестирования программного обеспечения на фреймворке Symfony с использованием языка программирования PHP.

В заключении подводятся итоги проведенного исследования, описываются выводы о достижении поставленных целей и обсуждаются

перспективы дальнейшего развития автоматизированных систем транспортной логистики.

Данная работа включает пояснительную записку объемом 85 страницы, содержащую 46 рисунков, 8 таблиц, а также список из 19 источников на русском языке и 8 источников на иностранном языке. Кроме того, приложены 3 дополнительных материала.

Содержание

Введение

В современном мире эффективная транспортная логистика играет ключевую роль в успехе бизнеса и бесперебойном функционировании экономики. Управление транспортными потоками необходимо для своевременной доставки товаров, снижения издержек и поддержания конкурентоспособности на рынке. Однако, сложность современных поставочных цепей и постоянно меняющиеся требования потребителей представляют существенные вызовы для традиционных логистических практик.

Появление передовых технологий, совместно с растущей доступностью данных, открыло новые возможности для оптимизации процессов транспортной логистики. Автоматизированные программные системы предлагают потенциал для оптимизации операций, улучшения принятия решений и повышения общей эффективности управления транспортными сетями. Путем использования аналитики реального времени, прогностического моделирования и интеллектуальных алгоритмов эти системы могут адаптироваться к динамическим условиям, предвидеть нарушения и оптимизировать распределение ресурсов.

Целью бакалаврской работы является разработка программного обеспечения автоматизированной системы управления транспортной логистикой. Исследование сосредоточено на проектировании и внедрении автоматизированной системы, которая сможет эффективно управлять сложностями современных поставочных цепей, соответствуя специфическим потребностям и требованиям предприятий в различных отраслях.

Через сочетание обзора литературы, проектирования системы и разработки программного обеспечения данная работа ставит перед собой следующие задачи:

– провести анализ существующих технологий, методов и лучших практик в области транспортной логистики;

– идентифицировать основные проблемы и возможности в автоматизации процессов управления транспортом;
– разработать архитектуру программного обеспечения, которая интегрируется беспрепятственно с существующими логистическими системами и инфраструктурой;
– разработать прототип автоматизированной системы транспортной логистики с использованием фреймворка Symfony и языка программирования PHP;
– оценить эффективность разработанной системы через тестирование и валидацию на реальных сценариях.

Достижение этих целей позволит получить ценные знания о потенциальных преимуществах автоматизированных систем транспортной логистики и внести вклад в развитие этой важной области исследований.

Объект исследования – программное обеспечение автоматизированной системы транспортной логистики.

Предмет исследования – процесс разработки и внедрения программного обеспечения для управления транспортными потоками и оптимизации логистических операций.

В первой главе проводится анализ существующих технологий и методов в транспортной логистике, обзор литературы, посвященной основным принципам логистики, оценка существующих программных решений и излагаются функциональные требования к новой системе. В ней также обосновывается необходимость создания новой автоматизированной системы путем выявления ограничений существующих решений и способов решения этих проблем с помощью новой системы.

Во второй главе сравниваются и выбираются инструменты для разработки программного обеспечения, устанавливаются критерии выбора инструментов, рассматриваются различные инструменты для разработки веб- и мобильных приложений. Затем объясняется выбранный стек технологий,

подробно описываются причины выбора конкретных инструментов и технологий.

В третьей главе описывается процесс проектирования разрабатываемой системы, описывается архитектура и приводятся схемы реализации, а также процесс самой реализации программного обеспечения на выбранном фреймворке Symfony с использованием языка программирования PHP. Рассматриваются технические аспекты разработки, приводится выбор инструментов разработки, написание кода, тестирование и оптимизация системы. Также предоставляется анализ результатов работы разработанного решения на основе контрольных примеров.

Данная тема совпадает с профессиональными компетенциями, поскольку она позволяет не только погрузиться в актуальные технологии и методы в области транспортной логистики, но и развить навыки анализа, проектирования и разработки программного обеспечения. Разрабатываемый проект предоставляет не только возможность применить теоретические знания на практике, а также решить реальные проблемы, стоящие перед современной логистической отраслью. Таким образом, выполнение данной работы будет способствовать развитию профессионального потенциала и подготовке к успешной карьере в сфере информационных технологий и логистики.

1 Анализ предметной области

1.1 Обзор существующих технологий и методов в области транспортной логистики

1.1.1 Введение в транспортную логистику

Транспортная логистика представляет собой комплексную систему управления потоками товаров и информации в рамках поставочной цепочки. Ее главная цель состоит в том, чтобы обеспечить оптимальное перемещение товаров от места хранения к месту потребления, учитывая при этом различные факторы, такие как время, стоимость, расстояние и качество обслуживания.

Основными задачами транспортной логистики являются:

- выбор оптимальных видов транспорта и маршрутов доставки грузов;
- планирование и организация перевозок с учетом сроков и объемов грузов;
- контроль за движением и сохранностью грузов на всех этапах транспортировки;
- минимизация логистических издержек, связанных с транспортировкой;
- обеспечение своевременной доставки грузов в соответствии с требованиями заказчиков.

Ключевыми участниками в цепочке транспортной логистики являются грузоотправители, перевозчики, экспедиторы, таможенные органы, склады и другие стороны, задействованные в процессе доставки. Эффективное взаимодействие и координация действий всех звеньев цепи поставок является важным условием для обеспечения бесперебойного движения грузопотоков.

В современном мире, где международная торговля и глобализация играют все более важную роль, транспортная логистика становится неотъемлемой частью успеха бизнеса. Она позволяет компаниям оперативно

реагировать на изменения рыночной ситуации, минимизировать издержки и обеспечивать высокий уровень обслуживания клиентов.

Однако современная транспортная логистика сталкивается с рядом вызовов, включая рост объемов грузоперевозок, увеличение конкуренции, нестабильность цен на топливо, изменения в законодательстве и требования клиентов к качеству обслуживания. Для эффективного преодоления этих вызовов необходимо постоянное совершенствование и инновации в области транспортной логистики.

В этой работе мы проведем обзор существующих технологий, методов и лучших практик в области транспортной логистики с целью выявления современных трендов и перспективных направлений развития в этой области. Мы также рассмотрим возможности применения информационных технологий и автоматизации для оптимизации транспортных процессов и повышения эффективности логистических операций.

1.1.2 Технологии управления транспортными потоками

В современной транспортной логистике широко применяются различные технологии для управления транспортными потоками с целью повышения эффективности и оптимизации логистических операций. В этом разделе мы рассмотрим некоторые из ключевых технологий, которые активно применяются в сфере управления транспортными потоками:

- GPS-отслеживание: GPS-технологии позволяют отслеживать местоположение транспортных средств в режиме реального времени;
- системы управления транспортными потоками (ITS): ITS представляют собой комплексные информационные системы, которые объединяют в себе различные технологии и методы для управления транспортными потоками, такие элементы, как датчики дорожного движения, системы светофорного регулирования и др.;
- системы маршрутизации и планирования транспортных маршрутов: эти системы используют алгоритмы оптимизации для автоматического расчета наиболее эффективных маршрутов

доставки грузов с учетом таких факторов, как расстояние, время в пути, стоимость топлива и пробки на дорогах;

– системы управления транспортными ресурсами: эти системы позволяют логистическим компаниям эффективно управлять своим транспортным парком, включая распределение ресурсов, планирование обслуживания и технического обслуживания, а также контроль за исполнением задач и соблюдением сроков.

Это лишь небольшой обзор технологий, используемых для управления транспортными потоками в современной транспортной логистике. Эти технологии играют ключевую роль в оптимизации логистических операций и обеспечении высокого уровня обслуживания клиентов.

1.1.3 Системы управления транспортными ресурсами

В современной логистике системы управления транспортными ресурсами (Transport Management System/TMS) играют ключевую роль в эффективном планировании и контроле за транспортными операциями. Эти системы предоставляют комплексный инструментарий для управления транспортными ресурсами, включая грузовой транспорт, склады, терминалы и персонал.

Основные функции систем управления транспортными ресурсами включают в себя:

– планирование транспортных маршрутов: TMS позволяют оптимизировать маршруты доставки грузов с учетом различных факторов, таких как время в пути, стоимость топлива, условия дорог и требования клиентов;

– управление транспортным парком: системы управления транспортными ресурсами предоставляют инструменты для контроля за состоянием и местоположением транспортных средств, планирования их использования и распределения задач между водителями;

– мониторинг и отчетность: TMS предоставляют возможность мониторинга выполнения транспортных операций в режиме реального времени, а также генерации отчетов о выполненных задачах, затратах на транспортировку и других ключевых показателях производительности;
– управление заказами и поставками: системы управления транспортными ресурсами автоматизируют процессы приема и обработки заказов, а также планирования и отслеживания поставок товаров от поставщиков до конечных потребителей;
– интеграция с другими системами: TMS обеспечивают интеграцию с другими информационными системами, такими как системы управления складом, системы учета и управления персоналом, что позволяет создать единую информационную платформу для управления всеми логистическими процессами компании.

Применение систем управления транспортными ресурсами позволяет компаниям повысить эффективность и надежность своих транспортных операций, сократить издержки на логистику и обеспечить высокий уровень обслуживания клиентов. Далее мы рассмотрим конкретные примеры таких систем и их влияние на логистические процессы в различных отраслях.

1.1.4 Применение аналитики данных в транспортной логистике

Аналитика данных и машинное обучение играют все более важную роль в оптимизации транспортных операций и повышении эффективности логистических процессов. Применение алгоритмов машинного обучения позволяет компаниям анализировать большие объемы данных, выявлять скрытые закономерности и шаблоны, а также прогнозировать будущие события с высокой точностью.

Некоторые из способов применения аналитики данных в транспортной логистике включают в себя:

– прогнозирование спроса: алгоритмы машинного обучения могут анализировать исторические данные о продажах и поставках, а также

внешние факторы, такие как погода, экономические показатели и сезонные тренды, чтобы прогнозировать будущий спрос на товары и оптимизировать уровень запасов;

– оптимизация маршрутов: аналитика данных позволяет оптимизировать маршруты доставки грузов, учитывая различные факторы, такие как расстояние, время в пути, условия дорог и пробки. Это позволяет сократить время доставки, снизить издержки на топливо и повысить общую эффективность транспортных операций;
– прогнозирование сроков поставки: аналитика данных позволяет предсказывать сроки поставки грузов с высокой точностью, учитывая различные факторы, такие как время в пути, загрузка складов и прочее. Это помогает компаниям планировать свои логистические операции и предотвращать задержки в доставке.

Применение аналитики данных в транспортной логистике позволяет компаниям повысить эффективность и надежность своих транспортных операций, снизить издержки на логистику и обеспечить высокий уровень обслуживания клиентов. Далее мы рассмотрим конкретные примеры применения аналитики данных в различных отраслях и их влияние на логистические процессы.

1.1.5 Вызовы и тренды в транспортной логистике

Транспортная логистика сталкивается с рядом вызовов и трендов, которые влияют на способы, которыми компании организуют и управляют своими логистическими операциями. Некоторые из основных вызовов и трендов включают в себя:

– устойчивость: увеличение внимания к вопросам устойчивости и экологической ответственности требует от компаний разработки более эффективных и экологически чистых методов транспортировки и управления логистическими операциями;
– цифровая трансформация: внедрение цифровых технологий, таких как интернет вещей (IoT), блокчейн и искусственный интеллект (ИИ),

меняет способы управления и контроля за транспортными потоками, повышая их эффективность и надежность;

– гибридные модели доставки: рост онлайн-торговли и изменение потребительского поведения приводят к появлению новых гибридных моделей доставки, таких как доставка из магазина, доставка на тот же день и самовывоз, которые требуют от компаний более гибких и интегрированных логистических решений;
– глобализация и комплексность цепей поставок: расширение географии бизнеса и рост сложности цепей поставок требует от компаний более гибких и адаптивных подходов к управлению транспортными потоками, включая развитие новых транспортных маршрутов и стратегий доставки.

Анализ вызовов и трендов в транспортной логистике позволяет компаниям адаптироваться к изменяющимся условиям рынка и разрабатывать стратегии, направленные на повышение конкурентоспособности и удовлетворение потребностей клиентов. Далее мы рассмотрим примеры успешной реализации таких стратегий и их влияние на логистические процессы в различных отраслях.

1.2 Обзор существующих программных решений

Согласно учебно-методическому пособию «Основы логистики» [15], для автоматизации и оптимизации процессов транспортной логистики на рынке представлен широкий спектр программных продуктов. Их можно классифицировать по следующим основным категориям:

а) системы управления транспортной логистикой (Transportation Management Systems, TMS):
    1) предназначены для планирования, организации и контроля перевозок грузов;

2) основные функции: маршрутизация, управление парком транспортных средств, отслеживание грузов, расчет стоимости доставки и т.д.;
3) примеры: Яндекс.Маршрутизация, Якурьер, Мегалогист.

б) системы управления складской логистикой (Warehouse Management Systems, WMS):
1) обеспечивают автоматизацию процессов приемки, хранения, комплектации и отгрузки товаров;
2) включают функции учета запасов, управления складскими операциями, интеграции с ERP-системами;
3) примеры: 1С:WMS Логистика. Управление складом, Фолио WMS, CloudShop.

в) комплексные ERP-системы с модулями логистики:
1) интегрируют управление различными бизнес-процессами, включая логистику;
2) обеспечивают планирование, организацию и контроль всей цепочки поставок;
3) примеры: 1С: ERP, Галактика ERP, МойСклад.

Для сравнения в нашей работе мы выбрали следующие TMS-системы:

– Яндекс.Маршрутизация,
– Якурьер,
– Мегалогист,
– собственная система.

Для сравнения существующих систем наряду с реализацией собственной системы возьмем следующие критерии:

– стоимость лицензии: указываются тарифы разового платежа или тарифы на ежемесячной (или ежегодной) подписочной основе;
– максимальное количество курьеров по тарифам: доступное по всем тарифам системы;

– максимальное количество заказов по тарифам: доступное по всем тарифам системы;
– выкуп программы: есть ли возможность выкупа лицензии и установки ПО на своей инфраструктуре;
– построение маршрутов: возможность построения маршрутов, доступная на любых тарифах сервиса.

Сравнение представлено в таблице 1.

Таблица 1 – Сравнение TMS-систем

Параметр сравнения	Название TMS-системы			
	Яндекс.Маршрутизация	Якурьер	Мегалогист	Собственная система
Стоимость лицензии	Тарифы: ограничение по пользователям - до 100 курьеров в квартал – 951048 руб.; ограничение по заказам - до 90000 заказов в квартал – 566100 руб.	Тариф от 10 водителей – 8000 руб. 2 месяца бесплатного доступа	Стоимость безлимитной лицензии на программу – 360 тыс. руб.	Стоимость (разработка, поддержка и оплата оборудования - до 200 тыс. руб. в месяц)
Максимальное количество курьеров	до 100 за год	до 10	Нет ограничений	Нет ограничений
Максимальное количество заказов	до 90 тыс. в квартал	Нет ограничений	до 10 тыс. в месяц	Нет ограничений
Выкуп программы	Нет (SaaS решение)	от 5 млн. руб.	от 100 тыс. руб.	-
Построение маршрутов	Есть	Начиная с PRO-тарифа	Есть	Есть

Анализ преимуществ и ограничений существующих решений показывает, что они обладают широкими функциональными возможностями, но зачастую требуют значительных финансовых и временных затрат на

внедрение. Кроме того, многие из них ориентированы на крупные предприятия и не всегда подходят для нужд малого и среднего бизнеса. Это создает потребность в разработке более гибких и доступных автоматизированных систем транспортной логистики.

1.3 Требования к разрабатываемой системе

При разработке автоматизированной системы управления транспортной логистикой необходимо учитывать ряд ключевых требований, выявленных на основе изучения потребностей пользователей и анализа факторов, влияющих на эффективность логистических процессов.

Согласно источнику «Логистика: учебное пособие» [2], эффективная логистика является неотъемлемой частью успешной стратегии увеличения продаж. Хорошо организованная и оптимизированная система доставки товаров позволяет снизить затраты, улучшить качество обслуживания клиентов и повысить конкурентоспособность компании.

Согласно учебному пособию «Транспортная логистика» [13], первостепенной задачей является выявление основных проблем и «болевых точек» пользователей - логистов, экспедиторов, менеджеров по перевозкам и других заинтересованных сторон. Это позволит определить приоритетные функциональные требования к разрабатываемой системе.

Ключевые функциональные требования к автоматизированной системе транспортной логистики включают:

- планирование и оптимизация маршрутов доставки;
- управление парком транспортных средств;
- отслеживание местоположения и статуса грузов;
- интеграция с системами складского учета и ERP;
- расчет стоимости и сроков доставки.

Помимо этого, важно учитывать и нефункциональные требования, такие как масштабируемость, надежность, удобство интерфейса, безопасность данных и т.д.

Согласно источнику, ключевыми факторами, влияющими на эффективность транспортной логистики, являются:

– скорость и своевременность доставки;
– оптимизация маршрутов и загрузки транспортных средств;
– минимизация логистических издержек;
– обеспечение сохранности грузов;
– гибкость и адаптивность логистических процессов.

Учет данных факторов при разработке системы позволит повысить ее практическую ценность для пользователей и обеспечить соответствие их потребностям.

1.4 Обоснование необходимости разработки автоматизированной системы

Анализ существующих программных решений для управления транспортной логистикой, представленный в предыдущем разделе, позволяет выявить ряд проблем и ограничений, обуславливающих необходимость разработки новой автоматизированной системы.

Согласно источнику [6], многие коммерческие системы ориентированы в первую очередь на крупные предприятия и требуют значительных финансовых и временных затрат на внедрение. Это создает барьеры для их использования малым и средним бизнесом, нуждающимся в более доступных и гибких решениях.

Согласно учебному пособию «Транспортная логистика» [13], многие коммерческие TMS, WMS и ERP-системы ориентированы в первую очередь на крупные предприятия и требуют значительных финансовых и временных

затрат на внедрение. Это создает барьеры для их использования малым и средним бизнесом, нуждающимся в более доступных и гибких решениях.

Кроме того, источник [13] отмечает, что существующие программные продукты не всегда обеспечивают полную автоматизацию и оптимизацию ключевых логистических процессов, таких как планирование маршрутов, управление парком транспортных средств, отслеживание грузов и т.д. Это снижает их эффективность и не позволяет в полной мере реализовать преимущества автоматизации.

Таким образом, разработка новой автоматизированной системы управления транспортной логистикой позволит преодолеть ограничения существующих решений и обеспечить следующие ключевые преимущества:

- повышение эффективности планирования и организации перевозок за счет оптимизации маршрутов и загрузки транспортных средств;
- сокращение логистических издержек, связанных с транспортировкой, хранением и обработкой грузов;
- улучшение контроля и прозрачности движения материальных потоков на всех этапах цепочки поставок;
- повышение качества обслуживания клиентов за счет обеспечения своевременной и надежной доставки.

Внедрение разрабатываемой автоматизированной системы транспортной логистики позволит предприятию достичь значительных операционных и финансовых выгод, повысить конкурентоспособность и адаптивность к изменяющимся рыночным условиям.

Вывод к главе 1

В первой главе был проведен анализ предметной области транспортной логистики и существующих программных решений для автоматизации логистических процессов.

Транспортная логистика играет ключевую роль в обеспечении эффективного перемещения материальных потоков в цепочках поставок. Основными задачами транспортной логистики являются выбор оптимальных маршрутов и видов транспорта, планирование и организация перевозок, контроль за движением грузов, а также минимизация логистических издержек.

На рынке представлен широкий спектр программных продуктов для управления транспортной логистикой, включая системы управления транспортной логистикой (TMS), системы управления складской логистикой (WMS) и комплексные ERP-системы с логистическими модулями. Однако многие из этих решений ориентированы на крупные предприятия и требуют значительных затрат на внедрение, что создает барьеры для их использования малым и средним бизнесом.

Анализ требований к разрабатываемой автоматизированной системе транспортной логистики показал необходимость реализации таких функций, как планирование маршрутов, управление парком транспортных средств, отслеживание грузов, интеграция с другими системами и расчет стоимости доставки. Также важно обеспечить масштабируемость, надежность, безопасность и удобство использования системы.

Разработка новой автоматизированной системы управления транспортной логистикой позволит преодолеть ограничения существующих решений, повысить эффективность логистических процессов, сократить издержки и улучшить качество обслуживания клиентов. Внедрение такой системы обеспечит предприятию значительные операционные и финансовые выгоды, а также повысит его конкурентоспособность.

2 Сравнение и выбор инструментов для разработки программного обеспечения

2.1 Критерии выбора инструментов разработки

При выборе инструментов для разработки программного обеспечения автоматизированной системы транспортной логистики необходимо руководствоваться рядом критериев, которые позволят обеспечить высокое качество, производительность и масштабируемость конечного продукта. Согласно учебному пособию «Проектирование информационных систем» [4], к ключевым критериям выбора инструментов разработки относятся:

а) функциональные возможности:
   1) планирование маршрутов,
   2) управление транспортными средствами,
   3) отслеживание грузов,
   4) интеграция с другими системами и т.д.

б) производительность и масштабируемость,

в) кроссплатформенность,

г) стоимость, соответствующая бюджету,

д) перспективы развития,

е) интеграция с другими системами,

ж) безопасность.

Руководствуясь данными критериями, можно произвести обоснованный выбор оптимального стека технологий для разработки программного обеспечения автоматизированной системы транспортной логистики.

2.2 Обзор инструментов для разработки веб-приложений

2.2.1 Языки программирования

При разработке веб-приложений широко используются такие языки программирования, как PHP, Python, Java, JavaScript и другие. Согласно учебному пособию «Разработка веб-приложений» [16], выбор языка зависит от требований к производительности, масштабируемости и сложности приложения.

PHP – популярный язык сценариев на стороне сервера, известный своей простотой и удобством использования. На нем написана значительная часть веб-страниц и веб-приложений, включая крупные платформы. PHP широко поддерживается веб-серверами, что делает его практичным выбором для веб-приложений любого масштаба, как монолитных, так и микросервисных или гибридных типов архитектур.

Python является популярным языком с простым синтаксисом и множеством библиотек, в том числе для веб-разработки. Он хорошо подходит для создания высокопроизводительных веб-приложений.

Java – мощный объектно-ориентированный язык, который часто используется для создания корпоративных веб-приложений с высокими требованиями к производительности и масштабируемости. Популярными фреймворками для Java являются Spring, JSF и другие.

JavaScript – язык, изначально предназначенный для клиентской веб-разработки, но с появлением Node.js получивший возможность использования и на стороне сервера. Широко применяется для создания современных одностраничных веб-приложений (SPA) с использованием фреймворков React, Angular, Vue.js.

2.2.2 Фреймворки

Для ускорения и упрощения процесса веб-разработки широко используются различные фреймворки, предоставляющие готовые решения и библиотеки для реализации типовых задач.

Для PHP существует множество фреймворков, упрощающих разработку, таких как Symfony, Laravel, CodeIgniter и другие. Они предоставляют готовые компоненты и следуют современным парадигмам веб-разработки [25].

Для Python популярными фреймворками являются Django и Flask. Django предлагает «batteries-included» философию и обширный набор возможностей, в то время как Flask является более легковесным и гибким решением [14].

Популярными фреймворками для Java являются Spring, JSF и другие [24]. Лидирующие позиции занимает фреймворк Spring, обеспечивающий комплексную модульную платформу для создания корпоративных приложений.

Для JavaScript наиболее распространенными являются Angular, React и Vue.js. Каждый из этих фреймворков имеет свои преимущества и недостатки, а выбор конкретного решения зависит от требований проекта, имеющейся экспертизы команды разработчиков и других факторов.

2.2.3 Базы данных

Важной составляющей веб-приложений являются системы управления базами данных (СУБД). Здесь можно выделить реляционные (MySQL, PostgreSQL, Oracle) и нереляционные (MongoDB, Cassandra, Redis) СУБД [10].

Реляционные СУБД основаны на реляционной модели данных и широко используются в корпоративных приложениях. Нереляционные (NoSQL) СУБД предоставляют альтернативные способы хранения и обработки данных, что может быть более эффективным для определенных типов приложений.

2.2.4 Средства разработки

Для эффективной разработки веб-приложений необходимо использовать специализированные средства, такие как интегрированные среды разработки (IDE) и редакторы кода. Популярными IDE являются PhpStrorm (PHP), PyCharm (Python), IntelliJ IDEA (Java), WebStorm (JavaScript). Широко используются также редакторы кода, такие как Visual Studio Code, Sublime Text и другие.

Этот обзор дает общее представление об основных инструментах для разработки веб-приложений, их преимуществах и областях применения. В следующих разделах мы можем более подробно рассмотреть конкретные технологии и провести их сравнительный анализ.

2.3 Обзор инструментов для мобильной разработки

В современных условиях мобильные приложения становятся все более важной составляющей программных решений, в том числе и для систем управления транспортной логистикой. Существуют два основных подхода к разработке мобильных приложений [9]:

2.3.1 Нативная разработка

Данный подход предполагает использование языков и инструментов, специфичных для каждой мобильной платформы. Для платформы Android это языки Java и Kotlin, для iOS - Swift и Objective-C.

Преимущества нативной разработки:

– высокая производительность и быстродействие приложений;
– полный доступ к нативным API и возможностям устройств;
– соответствие руководствам по дизайну конкретной платформы.

Недостатки:

– необходимость разрабатывать отдельные версии приложения для каждой платформы;
– более высокие затраты на разработку и поддержку.

2.3.2 Кроссплатформенная разработка

Данный подход позволяет создавать единое приложение, которое может работать на различных мобильных платформах. Для этого используются специальные инструменты и фреймворки, такие как React Native [21], Flutter [1], Xamarin [27] и другие.

Преимущества кроссплатформенной разработки:

– единая кодовая база для всех платформ;

– сокращение затрат на разработку и поддержку;
– более быстрая разработка и вывод приложения на рынок;

Недостатки:

– потенциальные проблемы с производительностью и совместимостью;
– ограниченный доступ к нативным API и возможностям устройств;
– необходимость следовать общим руководствам по дизайну.

2.4 Выбор стека технологий

На данном этапе разработки автоматизированной системы управления транспортной логистикой создание нативных настольных приложений для Windows или Mac, или мобильных приложений для платформ Android и iOS связано со значительными затратами ресурсов и усилий. Это обусловлено необходимостью разрабатывать и поддерживать отдельные версии приложения для каждой платформы с использованием специфичных языков программирования и инструментов.

В связи с этим, для обеспечения кроссплатформенности и доступа к системе управления транспортной логистикой для всех заинтересованных сторон (логистов, экспедиторов, менеджеров по перевозкам и др.) было принято решение сосредоточиться на веб-разработке. Это позволит создать единое веб-приложение, доступное через браузер на различных устройствах, включая настольные компьютеры, ноутбуки, планшеты и смартфоны.

Веб-разработка обеспечивает ряд преимуществ:

– кроссплатформенность без необходимости создания отдельных нативных приложений;
– более быстрая и экономичная разработка по сравнению с мобильными приложениями;
– единая кодовая база для всех платформ и устройств;

– возможность использования системы через веб-браузер без установки дополнительного ПО.

Таким образом, на текущем этапе проекта основные усилия будут сосредоточены на разработке веб-приложения для системы управления транспортной логистикой. Это позволит обеспечить необходимый функционал и доступность системы для всех заинтересованных сторон при оптимальном распределении ресурсов.

В рамках данного проекта по созданию автоматизированной системы управления транспортной логистикой было принято решение использовать PHP в сочетании с фреймворком Symfony, а также Twig и EasyAdmin для реализации фронтенда. Этот выбор обоснован следующими ключевыми факторами:

Преимущества Symfony, Twig и EasyAdmin:

– Symfony обеспечивает высокую производительность и масштабируемость для создания веб-приложений. Symfony выгодно отличается от других PHP-фреймворков надёжностью и зрелостью. Этот фреймворк появился в 2005 году, то есть — существует гораздо дольше, чем большинство других фреймворков PHP. Он популярен благодаря соответствию веб-стандартам и шаблонам проектирования PHP.
– Twig предоставляет удобный инструмент для работы с шаблонами и отображения данных.
– EasyAdmin упрощает создание административного интерфейса и управление данными в приложении
– обширная экосистема Symfony и активное сообщество разработчиков обеспечивают поддержку и развитие фреймворка

На бэкенде в качестве языка программирования будет использоваться PHP с применением Symfony. Данный выбор обусловлен следующими факторами:

– надежность и масштабируемость Symfony для создания корпоративных веб-приложений
– модульная архитектура Symfony, позволяющая эффективно организовать разработку
– обширный функционал Symfony для быстрой и удобной работы с данными и бизнес-логикой
– активное сообщество Symfony и наличие обучающих материалов облегчают процесс разработки и поддержки приложения

Сочетание PHP с Symfony, Twig и EasyAdmin обеспечит создание эффективного, масштабируемого и удобного веб-приложения для управления транспортной логистикой, отвечающего требованиям проекта.

Вывод к главе 2

В данной главе был проведен обзор и сравнительный анализ различных инструментов и технологий для разработки программного обеспечения автоматизированной системы управления транспортной логистикой.

Для выбора оптимального стека технологий были определены ключевые критерии, такие как функциональные возможности, производительность, масштабируемость, кроссплатформенность, стоимость, поддержка сообщества, перспективы развития, интеграция с другими системами и безопасность.

На основе этих критериев и требований к разрабатываемой системе было принято решение использовать веб-разработку для обеспечения кроссплатформенного доступа к системе через веб-браузер на различных устройствах.

Изучив представленные источники, можно сделать вывод о том, что фронтенд от Symfony с Twig и EasyAdmin представляет собой эффективный инструмент для разработки веб-приложений на PHP. Symfony широко применяется для создания разнообразных веб-приложений, включая корпоративные системы и сервисы, API, системы управления контентом и интернет-магазины.

Для реализации баз данных планируется использовать реляционные СУБД, такие как PostgreSQL или MySQL, которые хорошо подходят для структурированных данных и обеспечивают требуемую производительность.

Выбранный стек технологий (PHP, Symfony, PostgreSQL/MySQL) соответствует современным тенденциям веб-разработки, обеспечивает необходимую производительность, масштабируемость, безопасность и кроссплатформенность для создания автоматизированной системы управления транспортной логистикой.

3 Разработка программного обеспечения автоматизированной системы транспортной логистики

3.1 Проектирование архитектуры системы

Архитектура программного обеспечения определяет общую структуру системы, ее основные компоненты, принципы их взаимодействия и ограничения, накладываемые на эти взаимодействия. Правильно спроектированная архитектура является ключевым фактором для обеспечения требуемых функциональных возможностей, производительности, масштабируемости и гибкости системы [3].

3.1.1 Диаграмма вариантов использования

Диаграмма вариантов использования (use case diagram) является одним из важнейших артефактов при проектировании программного обеспечения. Она визуально представляет функциональные требования к системе с точки зрения различных ролей пользователей (акторов) и их взаимодействия с системой [4].

На диаграмме вариантов использования будут отображены основные действующие лица (актеры) автоматизированной системы транспортной логистики, такие как Водитель, менеджер по логистике (Логист), Администратор, Внешние системы при интеграции с системой. Для каждого актера будут определены соответствующие варианты использования (use cases), описывающие функции и задачи, которые они могут выполнять в системе.

Диаграмма вариантов использования для разрабатываемой системы представлена на рисунке 1:

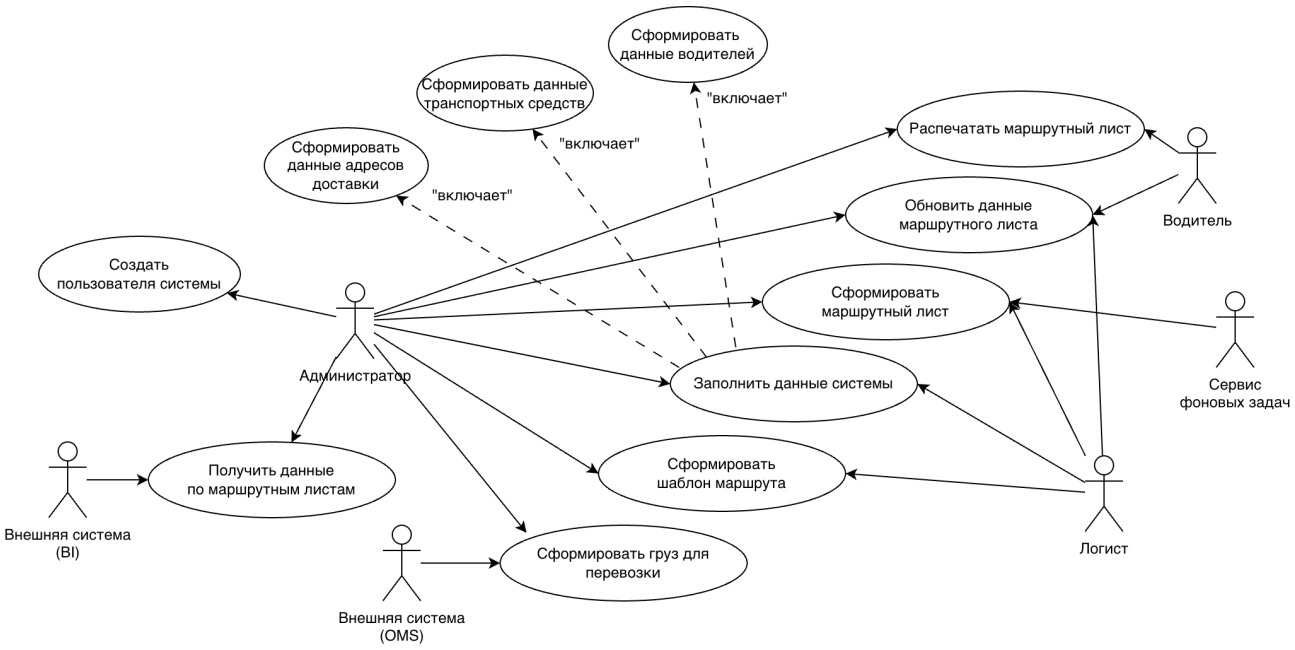


Рисунок 1 – Диаграмма вариантов использования системы

3.1.2 Диаграмма бизнес-процессов

Для описания и моделирования бизнес-процессов, которые должны быть автоматизированы в рамках разрабатываемой системы управления транспортной логистикой, будут использованы современные нотации и методологии проектирования процессов.

Согласно источнику [8], широкое применение находят такие нотации, как IDEF0, ARIS и другие. Они позволяют провести моделирование существующих процессов в нотации «как есть» (As-Is), а затем спроектировать желаемые процессы в нотации «как должно быть» (To-Be) с учетом требований к внедряемой информационной системе [7].

Для построения контекстных диаграмм деятельности организации может быть использована нотация IDEF0 и программное средство BPWin, как это описано в источнике на примере медицинской клиники.

Основными бизнес-процессами, которые должны быть автоматизированы в системе управления транспортной логистикой, являются:

а) планирование маршрутов перевозок:

1) формирование шаблонов маршрутов;

2) формирование маршрутных листов;

б) управление справочниками системы:

1) справочник складов/пунктов назначения;
2) справочник транспортных средств;
3) справочник водителей;

в) взаимодействие с внешними системами:

1) прием информации о грузах на перевозку;
2) передача информации по маршрутным листам (для аналитики и отчетности).

Далее будет проведено детальное описание перечисленных бизнес-процессов с использованием выбранной нотации IDEF0, диаграмма представлена на рисунке 2:

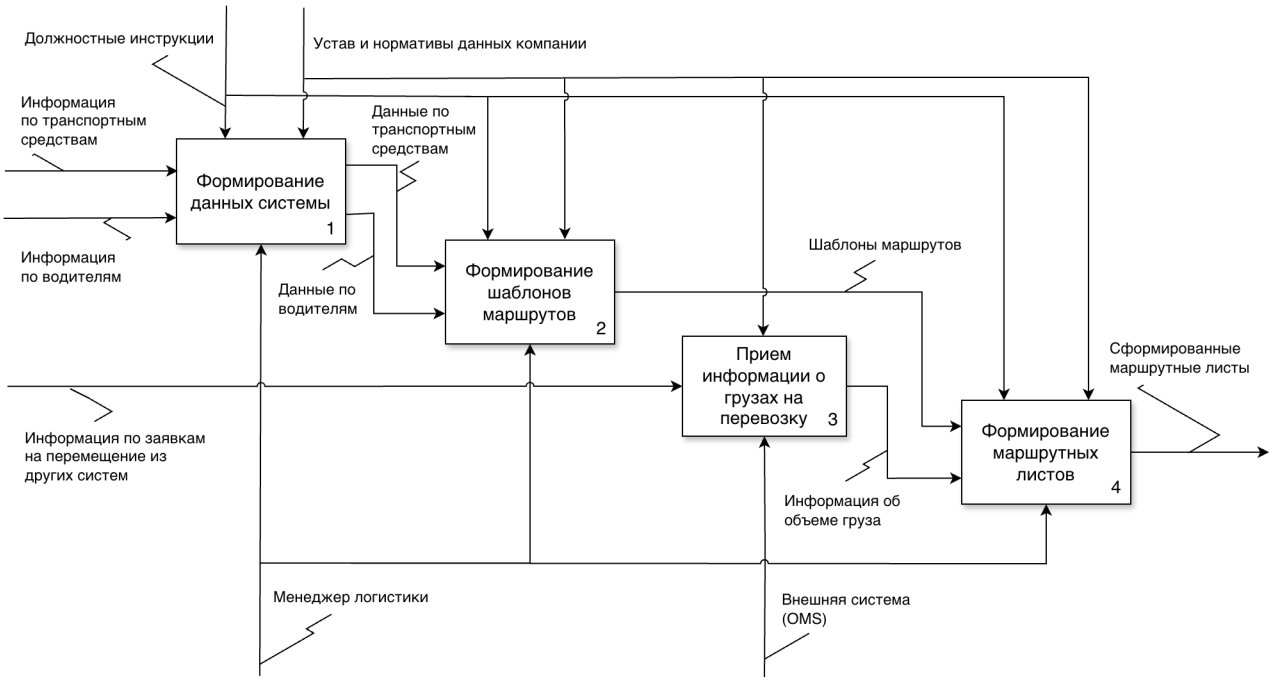


Рисунок 2 – Диаграмма бизнес-процессов разрабатываемой системы

3.1.3 Обзор методологий проектирования

При проектировании архитектуры автоматизированной системы управления транспортной логистикой были рассмотрены следующие распространенные методологии [11][12]:

– клиент-серверная архитектура: данный подход предполагает разделение системы на два основных компонента: клиентское приложение, предоставляющее пользовательский интерфейс, и серверную часть, выполняющую основную бизнес-логику и управляющую данными. Взаимодействие между клиентом и сервером осуществляется по сети с использованием определенных протоколов;
– многоуровневая архитектура: это развитие клиент-серверного подхода, при котором серверная часть дополнительно разделяется на несколько логических уровней, таких как уровень представления, бизнес-логики, интеграции и данных. Каждый уровень отвечает за определенный набор функций и взаимодействует со смежными уровнями по строго определенным интерфейсам;
– микросервисная архитектура: данная методология предполагает разделение системы на множество небольших независимых сервисов, каждый из которых реализует определенную бизнес-функцию. Сервисы взаимодействуют друг с другом через легковесные механизмы, часто используя REST API. Это обеспечивает гибкость, масштабируемость и возможность независимого развертывания компонентов.

3.1.4 Выбор архитектурного подхода

Для разрабатываемой системы управления транспортной логистикой была выбрана многоуровневая архитектура, основанная на классическом клиент-серверном подходе. Данный выбор обусловлен следующими факторами:

– четкое разделение ответственности между уровнями способствует модульности и облегчает сопровождение системы;
– многоуровневая архитектура хорошо подходит для создания масштабируемых корпоративных приложений;

– наличие уровня бизнес-логики позволяет реализовать сложные алгоритмы управления транспортной логистикой;
– возможность повторного использования компонентов на разных уровнях;
– обеспечение безопасности за счет изоляции критических данных на уровне сервера.

3.1.5 Описание компонентов системы

Автоматизированная система управления транспортной логистикой будет состоять из следующих основных компонентов:

а) клиентская часть приложения (фронтенд): фронтенд будет реализован с использованием библиотеки EasyAdmin, входящей в состав фреймворка Symfony. EasyAdmin предоставляет готовые средства для быстрого создания административных интерфейсов на основе сущностей Doctrine ORM. Это позволит сгенерировать пользовательский интерфейс для взаимодействия с системой, включая отображение данных, навигацию, ввод информации и отправку запросов на сервер;
б) серверная часть (бэкенд): бэкенд системы будет реализован на языке программирования PHP с использованием фреймворка Symfony. Он будет включать следующие основные компоненты:
    1) уровень представления (контроллеры) для обработки запросов от клиентского приложения и формирования ответов;
    2) уровень аутентификации и авторизации пользователей;
    3) уровень бизнес-логики с сервисами для планирования маршрутов, управления транспортом, интеграции с внешними системами и т.д.;
    4) уровень представления (REST API) для обработки запросов от фронтенда и других внешних сервисов;
    5) уровень доступа к данным с использованием Doctrine ORM для взаимодействия с базой данных и управления сущностями;

6) уровень данных с использованием СУБД (PostgreSQL) для хранения информации о маршрутах, транспортных средствах, грузах и т.п.;

в) вспомогательные сервисы;

1) дополнительные уровни, такие как фоновый сервис обработки очередей задач, интеграционные шлюзы и др.

Взаимодействие между компонентами системы осуществляется через четко определенные интерфейсы с использованием протоколов HTTP/HTTPS.

Спроектированная многоуровневая архитектура обеспечивает необходимую гибкость, масштабируемость и безопасность для создания автоматизированной системы управления транспортной логистикой, отвечающей современным требованиям.

3.2 Проектирование модели данных

3.2.1 Проектирование логической модели данных

Для хранения и обработки данных в рамках автоматизированной системы управления транспортной логистикой необходимо спроектировать схему данных для дальнейшего создания базы данных на основе этой схемы. В контексте нашего проекта можно выделить следующие основные сущности и их атрибуты:

а) упаковка (Pack)

1) идентификатор упаковки
2) объем упаковки (содержит общий объем упаковки)
3) планируемая дата доставки
4) фактическая дата доставки

б) транспортное средство (Vehicle)

1) идентификатор транспортного средства
2) тип транспортного средства (грузовик, фургон и т.д.)
3) грузоподъемность

4) регистрационный номер
5) признак активности

в) тип транспортного средства (VehicleType)
1) идентификатор типа
2) название типа (грузовик, фургон и т.д.)

г) водитель (Driver)
1) идентификатор водителя
2) имя водителя
3) телефон водителя
4) признак активности

д) пункт назначения (Destination)
1) идентификатор пункта назначения
2) тип пункта назначения (магазин, распределительный центр и т.д.)
3) адрес пункта назначения
4) признак активности

е) шаблон маршрута (RouteTemplate)
1) идентификатор шаблона маршрута
2) пункт отправления (склад)
3) назначенное транспортное средство
4) список пунктов назначения
5) расписание (выбор дней недели)
6) признак активности

ж) маршрутный лист (RouteSheet)
1) идентификатор маршрутного листа
2) маршрут
3) транспортное средство
4) водитель
5) общий вес груза
6) общий объем груза

7) список упаковок для доставки
8) список пунктов назначения с временем прибытия/отправления
9) дата и время начала маршрута
10) дата и время окончания маршрута
11) статус

и) статус маршрутного листа (RouteSheetStatus)
1) идентификатор статуса
2) наименование

Данная модель данных позволяет отразить основные сущности и их взаимосвязи в рамках процессов планирования маршрутов, управления транспортным парком, отслеживания грузов и формирования маршрутных листов.

На этапе логического проектирования будут определены все необходимые таблицы, их атрибуты, первичные и внешние ключи, а также ограничения целостности данных. На основании данной информации составлена схема модели данных, схема представлена на рисунке 3:

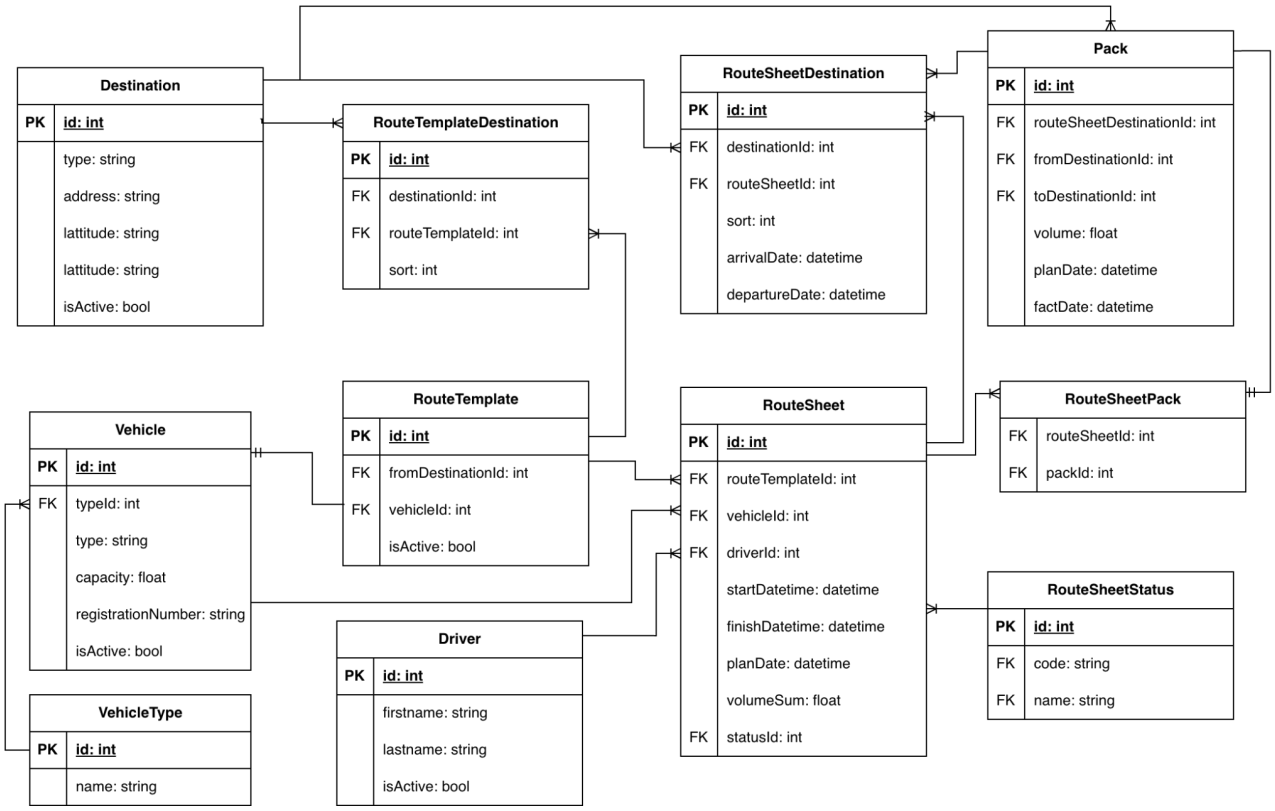


Рисунок 3 – Предварительная модель данных

После определения схемы данных будет произведено физическое проектирование с созданием схем баз данных и таблиц в соответствии с выбранной СУБД.

3.2.2 Проектирование физической схемы базы данных

Для реализации баз данных планируется использовать реляционную СУБД PostgreSQL, которая обеспечивает высокую производительность, надежность и соответствие стандартам SQL. PostgreSQL поддерживает все необходимые типы данных, ограничения целостности, а также обладает развитыми возможностями для работы с пространственными данными, что может быть полезно для реализации потенциальных функций маршрутизации и отслеживания транспорта, которые могут быть спроектированы и запрограммированы после реализации первой версии проекта.

Для реализации физической схемы базы данных и генерации соответствующих миграций будет использоваться фреймворк Symfony и его компонент Doctrine ORM [17]. Doctrine ORM представляет собой библиотеку

отображения данных для языка программирования PHP, которая обеспечивает абстракцию над реляционными базами данных и позволяет работать с ними через концептуальные сущности.

В рамках Symfony предусмотрен механизм автогенерации миграций на основе определенных сущностей и их отображений [20]. Этот процесс, известный как Code-First, позволяет сначала спроектировать сущности в коде приложения, а затем сгенерировать соответствующие миграции для создания или изменения структуры базы данных.

Процесс проектирования физической схемы базы данных будет выглядеть следующим образом:

- определение сущностей (Entity) в коде приложения на языке PHP с использованием аннотаций или XML для описания их свойств, связей и ограничений;
- выполнение команды генерации миграций в Symfony, которая анализирует определенные сущности и создает соответствующие файлы миграций;
- просмотр и проверка сгенерированных файлов миграций, содержащих SQL-инструкции для создания или изменения таблиц, индексов и ограничений в базе данных;
- применение миграций к целевой базе данных PostgreSQL с помощью команды выполнения миграций в Symfony.

Такой подход, основанный на Code-First [19] и автогенерации миграций, обеспечивает согласованность между моделью данных в коде приложения и физической схемой базы данных. Он также позволяет отслеживать изменения в схеме базы данных и применять их к различным средам (разработка, тестирование, производство) в контролируемом и воспроизводимом порядке.

В процессе разработки были созданы сущности, на основании модели данных:

- Destination – Пункт назначения
- Driver – Водитель

– Pack – Упаковка
– RouteSheet – Маршрутный лист
– RouteSheetDestination – Пункт назначения маршрутного листа
– RouteSheetStatus – Статус маршрутного листа
– RouteTemplate – Шаблон маршрута
– RouteTemplateDestination – Пункт назначения шаблона маршрута
– User – Пользователь
– Vehicle – Транспортное средство
– VehicleType – Тип транспортного средства

В результате генерации миграции для базы данных построена физическая схема, диаграмма которой изображена на рисунке 4:

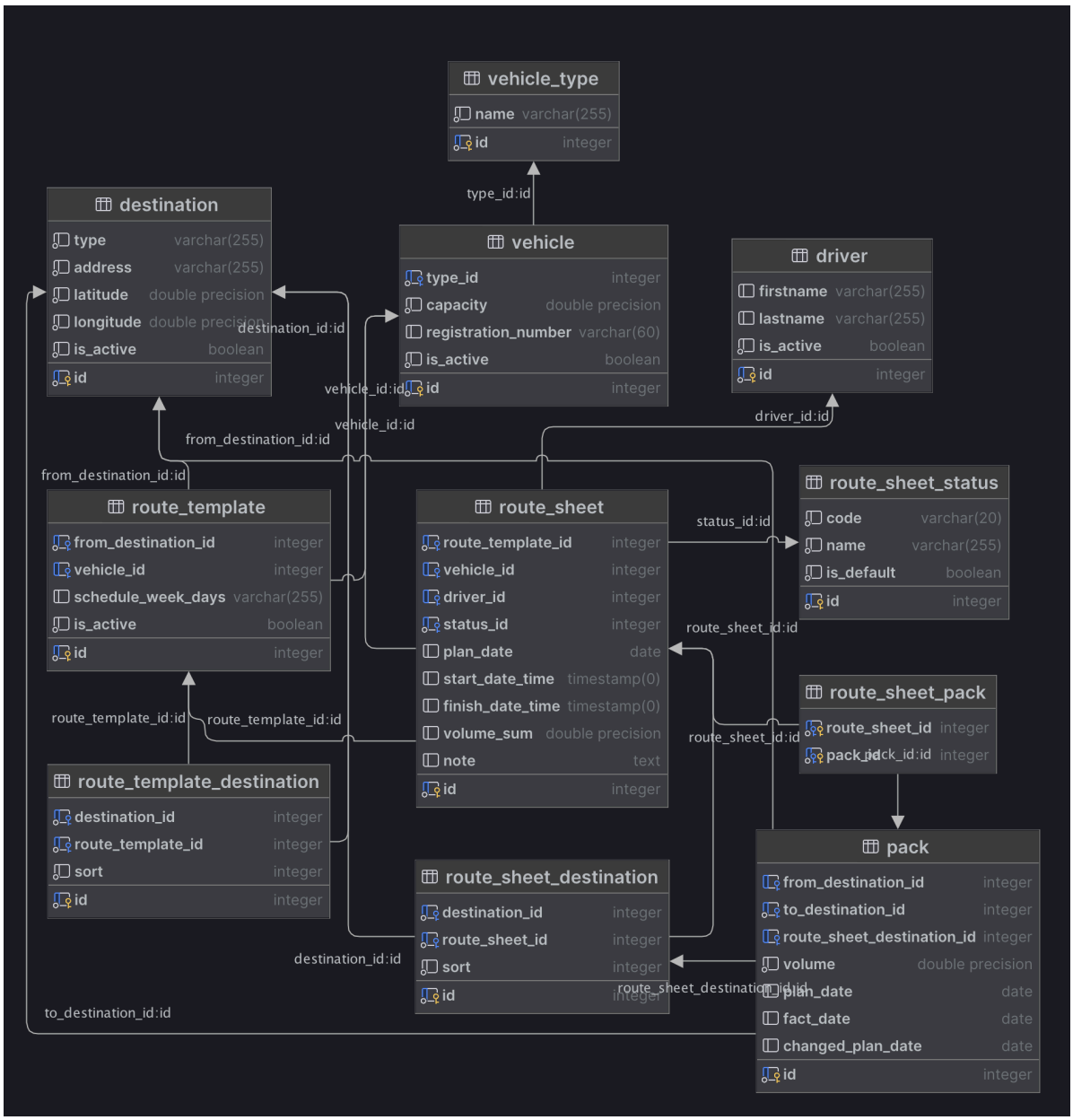


Рисунок 4 – Физическая схема данных в PostgreSQL

В процессе реализации проекта произошли некоторые изменения в сравнении с первоначальной схемой проектирования. Были внесены небольшие коррективы в признаки сущностей на этапе их реализации в кодовой базе. Однако, не произошло никаких глобальных изменений, и все основные зависимости остались неизменными. Таким образом, конечный результат проекта сохранил основные принципы и структуру, заложенные в первоначальном проектировании.

3.3 Разработка программной части

3.3.1 Подготовка окружения разработки

Для реализации данного проекта была применена интегрированная среда разработки (IDE) PhpStorm, которая является мощным инструментом для создания приложений на языке PHP. Эта среда поддерживает множество функций, оптимизирующих процесс разработки, включая рефакторинг, обработку данных, подключение к базам данных, и другие.

В качестве средства для ускорения процесса развертывания и масштабирования приложения в процессе разработки часто используют системы контейнеризации, такие как Docker, OpenShift, Kubernetes.

Docker представляет собой платформу для разработки, доставки и запуска приложений в контейнерах. Контейнеры позволяют упаковать приложение и все его зависимости в единое исполняемое окружение, обеспечивая изолированное и надежное выполнение приложения на любой среде.

Выбор Docker для развертывания приложения обусловлен следующими преимуществами:

- повышение портируемости и независимости приложения от окружения, что облегчает его развертывание на различных платформах;
- упрощение процесса развертывания и масштабирования приложения за счет использования контейнеров, которые создаются со своим внутренним независимым и изолированным контекстом;
- обеспечение изоляции и безопасности приложения, предотвращая конфликты между зависимостями и обеспечивая стабильную работу приложения.

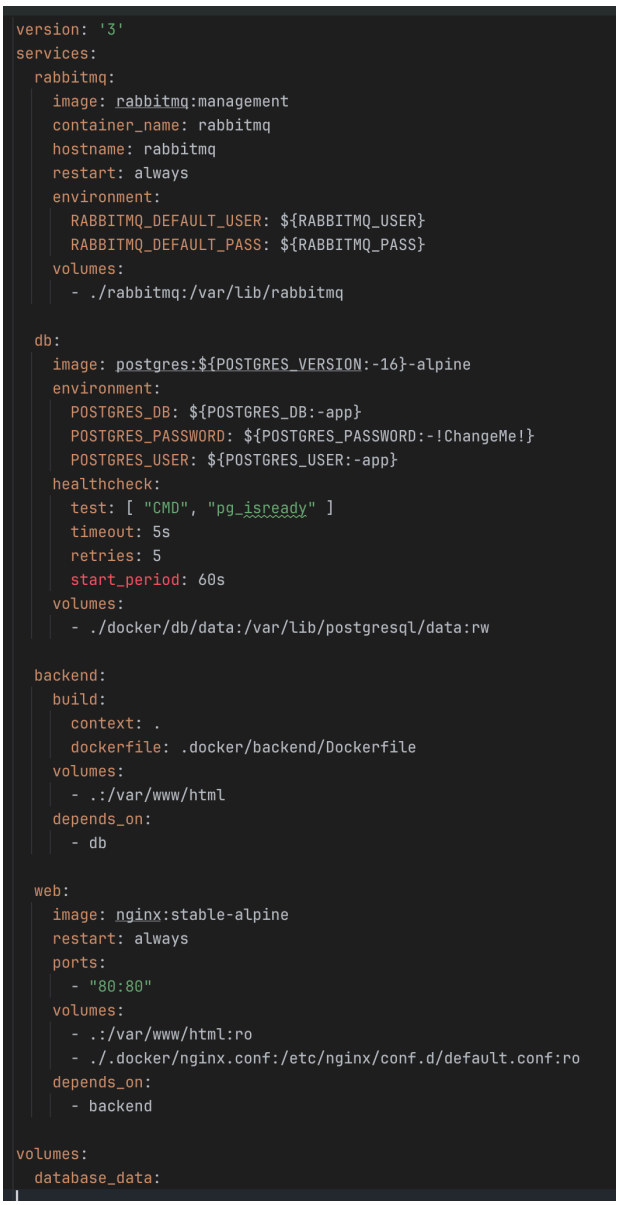
В нашем случае использование Docker было оправдано благодаря более быстрому процессу первоначального развертывания приложения на локальном компьютере для разработки и на тестовом сервере. Для этого

достаточно создать конфигурацию Docker Compose, включающую сервисы, которые должны быть запущены в одном окружении и взаимодействовать между собой, создавая таким образом единое изолированное пространство приложения.

С помощью Docker Compose была настроена конфигурация приложения, состоящая из следующих контейнеров сервисов:

- база данных - db (образ с PostgreSQL);
- сервис очередей rabbitmq (для обработки фоновых процессов генерации, образ RabbitMQ с Management);
- сервис backend (основан на образе с предустановленным окружением php-fpm для работы веб-приложения, Dockerfile описывает создание основного образа на основе образа последней версии языка PHP 8.3 – php:8.3-fpm);
- сервис web (основан на образе с предустановленным окружением nginx для обработки клиентских HTTP-запросов и перенаправляющим на backend сервис).

На рисунке 5 представлена конфигурация файла docker-compose.yaml:



```yaml
version: '3'
services:
  rabbitmq:
    image: rabbitmq:management
    container_name: rabbitmq
    hostname: rabbitmq
    restart: always
    environment:
      RABBITMQ_DEFAULT_USER: ${RABBITMQ_USER}
      RABBITMQ_DEFAULT_PASS: ${RABBITMQ_PASS}
    volumes:
      - ./rabbitmq:/var/lib/rabbitmq

  db:
    image: postgres:${POSTGRES_VERSION:-16}-alpine
    environment:
      POSTGRES_DB: ${POSTGRES_DB:-app}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD:-!ChangeMe!}
      POSTGRES_USER: ${POSTGRES_USER:-app}
    healthcheck:
      test: [ "CMD", "pg_isready" ]
      timeout: 5s
      retries: 5
      start_period: 60s
    volumes:
      - ./docker/db/data:/var/lib/postgresql/data:rw

  backend:
    build:
      context: .
      dockerfile: .docker/backend/Dockerfile
    volumes:
      - .:/var/www/html
    depends_on:
      - db

  web:
    image: nginx:stable-alpine
    restart: always
    ports:
      - "80:80"
    volumes:
      - .:/var/www/html:ro
      - ./.docker/nginx.conf:/etc/nginx/conf.d/default.conf:ro
    depends_on:
      - backend

volumes:
  database_data:
```

Рисунок 5 – Конфигурация развертывания окружения сервиса

Можно заметить, что при использовании контейнеров используется подключение диска, а именно файлы из корневой директории приложения используются как для сервиса backend, так и для web сервиса, чтобы иметь единый код приложения. В общей практике используются различные способы к организации конфигурации развертывания, и обычно для продуктивной разработки образы контейнеров собираются заранее для более быстрого развертывания приложения по “одному клику” с помощью инструментов CI/CD (Continuous Integration/Continuous Delivery) [5], что переводится, как непрерывная интеграция и непрерывное развертывание. CI/CD используется

для ускорения выпуска, снижения рисков производства и повышения качества проектов за счет настроенного автоматизированного процесса сборки и развертывания приложения на сервере или даже кластерах серверов, позволяет более быстро масштабировать приложение за счет копирования собранных образов на большем количестве серверов в моменты пиковых нагрузок или сокращать чрезмерное использование ресурсов, если это не требуется.

В нашем случае развертывание приложения через данный подход позволит в будущем разработчикам легко инсталлировать приложение на любом окружении, тестировать на разных серверах, и любые ветки разработки.

При развертывании backend контейнера настроена инсталляция необходимых зависимостей через Composer для работы приложения Symfony.

Пример консольной команды для использования и инсталляции зависимостей:

```
// инсталляция модуля взаимодействия с AMQP очередью
composer require symfony/amqp-messenger
// инсталляция всех зависимостей проекта
composer install
```

Были также настроены контейнеры базы данных PostgreSQL и сервиса очередей RabbitMQ и настроены параметры подключения к ним.

Через IDE также можно подключиться к базе данных и управлять информацией, на рисунке 6 представлено подключение к БД:

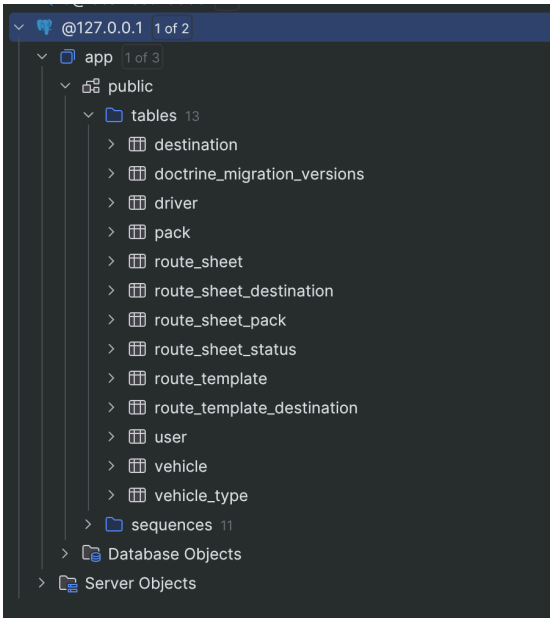


Рисунок 6 – Подключение к базе данных, отображение списка созданных таблиц

Сервис RabbitMQ имеет возможность установки клиентского веб-приложения Management для просмотра информации по очередям, пример обращения к интерфейсу данного приложения представлен на рисунке 7:

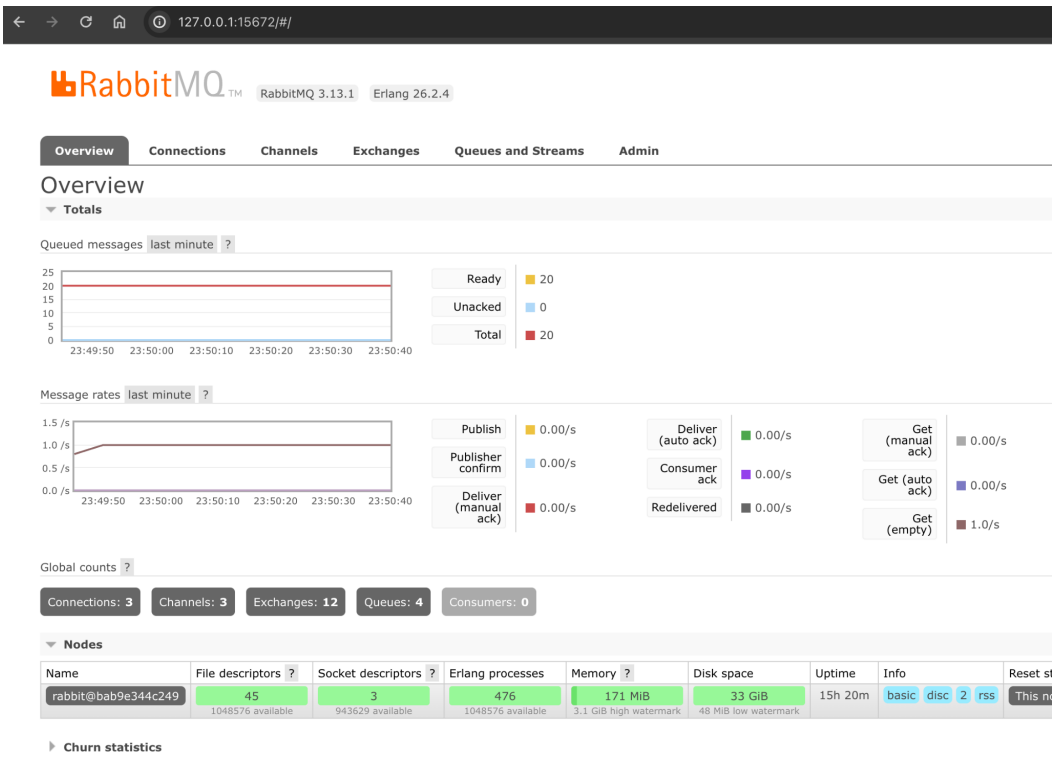


Рисунок 7 – Интерфейс RabbitMQ Management для управления очередями

3.3.2 Создание основных модулей и компонентов

После настройки окружения далее приступим к обзору основного процесса разработки. В начале разработки были реализованы основные модули системы, такие как управление транспортом, управление маршрутными шаблонами, маршрутными листами и другими сущностями.

С помощью подхода Code-First [19], описанным ранее, в первую очередь были созданы классы сущностей и их CRUD-модели и контроллеры, которые позволяют управлять сущностью по всем возможным сценариям - Создание/Просмотр/Редактирование/Удаление. Такие модели позволяют унифицировать логику обработки и валидации данных по всем сущностям системы, разработать единые и гибкие шаблоны интерфейсов, что позволяет в целом ускорить и оптимизировать разработку, тестирование, упростить использование системы и обучение для пользователей. Список реализованных сущностей представлен на рисунке 8 и пример класса сущности на рисунке 9:

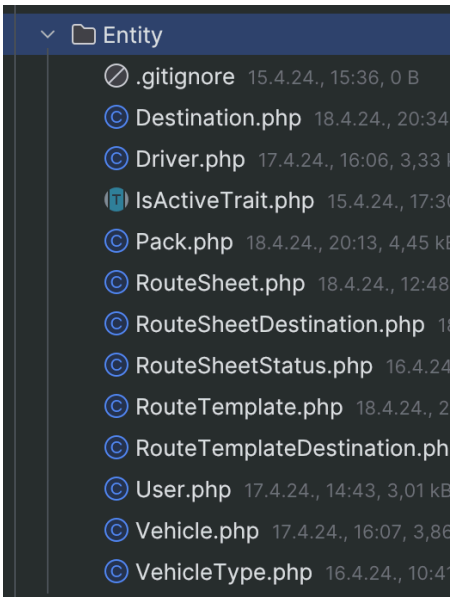
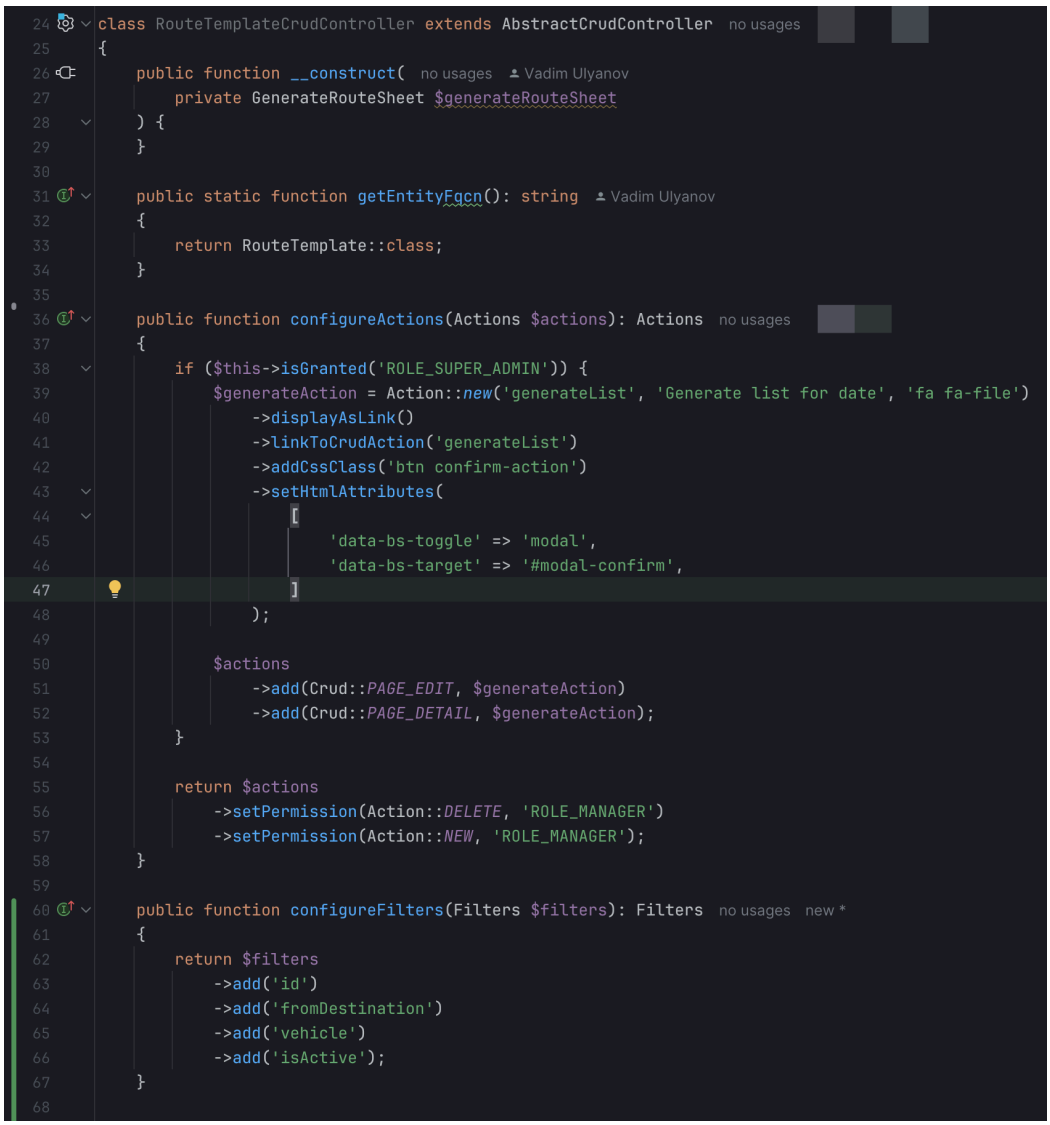


Рисунок 8 – Список созданных сущностей системы

Рисунок 9 – Пример описания сущности с помощью аннотаций PHP 8, Doctrine и Symfony

Для каждой сущности был создан собственный административный CRUD-контроллер, который отвечает за обработку запросов и отправку ответов клиенту. CRUD-контроллеры позволяют задать отображение форм сущности, задать правила создания/просмотра/редактирования/удаления сущностей системы, назначать уровни доступа той или иной функции, задать различные параметры и фильтры CRUD-модели и др. Пример создания CRUD-контроллера представлено на рисунке 10:



```php
class RouteTemplateCrudController extends AbstractCrudController  no usages
{
    public function __construct(  no usages  Vadim Ulyanov
        private GenerateRouteSheet $generateRouteSheet
    ) {
    }

    public static function getEntityFqcn(): string  Vadim Ulyanov
    {
        return RouteTemplate::class;
    }

    public function configureActions(Actions $actions): Actions  no usages
    {
        if ($this->isGranted('ROLE_SUPER_ADMIN')) {
            $generateAction = Action::new('generateList', 'Generate list for date', 'fa fa-file')
                ->displayAsLink()
                ->linkToCrudAction('generateList')
                ->addCssClass('btn confirm-action')
                ->setHtmlAttributes(
                    [
                        'data-bs-toggle' => 'modal',
                        'data-bs-target' => '#modal-confirm',
                    ]
                );

            $actions
                ->add(Crud::PAGE_EDIT, $generateAction)
                ->add(Crud::PAGE_DETAIL, $generateAction);
        }

        return $actions
            ->setPermission(Action::DELETE, 'ROLE_MANAGER')
            ->setPermission(Action::NEW, 'ROLE_MANAGER');
    }

    public function configureFilters(Filters $filters): Filters  no usages  new *
    {
        return $filters
            ->add('id')
            ->add('fromDestination')
            ->add('vehicle')
            ->add('isActive');
    }
```

Рисунок 10 – Пример описания CRUD-контроллера

Данные настройки позволяют добиться необходимого отображения данных в административной панели, ограничить функции уровнями доступа функции и др., пример отображения основного списка сущности RouteTemplate представлен на рисунке 11:

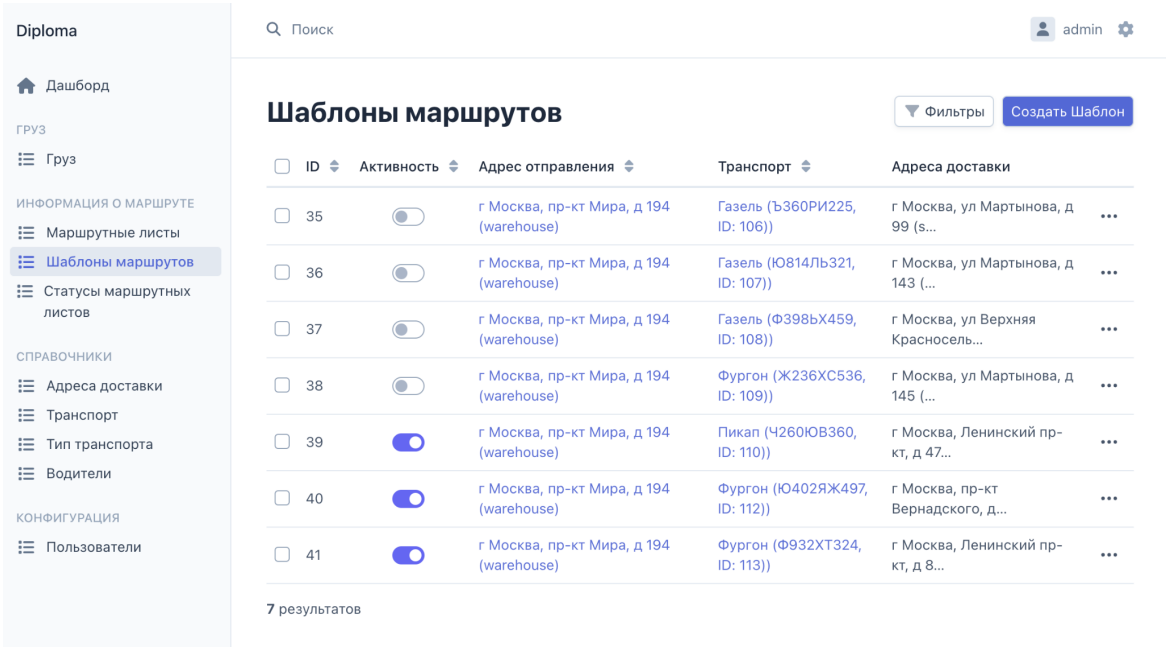


Рисунок 11 – Пример отображения настроенной CRUD-модели по описанию из CRUD-контроллера

3.3.3 Реализация бизнес-логики

Для реализации бизнес-логики были использованы сервисы, которые отвечают за конкретные задачи, такие как расчет маршрутов, управление и другие. Сервисы были созданы в отдельном слое и инжектировались в контроллеры при необходимости.

В рамках данной работы удалось разработать процессы, соответствующие заданным бизнес-процессам, а именно:

а) планирование маршрутов перевозок:

1) формирование шаблонов маршрутов – управление шаблонами происходит с помощью CRUD-контроллера, пример представлен выше;
2) формирование маршрутных листов – управление шаблонами происходит с помощью CRUD-контроллера, а также с помощью фонового процесса генерации. Процесс автогенерации будет описан отдельной схемой;

б) управление справочниками системы:

1) справочник складов/пунктов назначения – управление шаблонами происходит с помощью CRUD-контроллера сущности Destination;
2) справочник транспортных средств – управление шаблонами происходит с помощью CRUD-контроллера сущности Vehicle;
3) справочник водителей – управление шаблонами происходит с помощью CRUD-контроллера сущности Driver;

в) взаимодействие с внешними системами – используется Basic HTTP авторизация на основании имени пользователя и пароля:
1) прием информации о грузах на перевозку – прием информации реализован с помощью API-контроллера ApiController посредством метода POST /api/pack;
2) передача информации по маршрутным листам (для аналитики и отчетности) – передача информации реализована также с помощью API-контроллера ApiController посредством метода GET /api/route-lists;

Отдельно стоит обозначить схему процесса автогенерации маршрутных листов. Схема обозначена на рисунке 12:

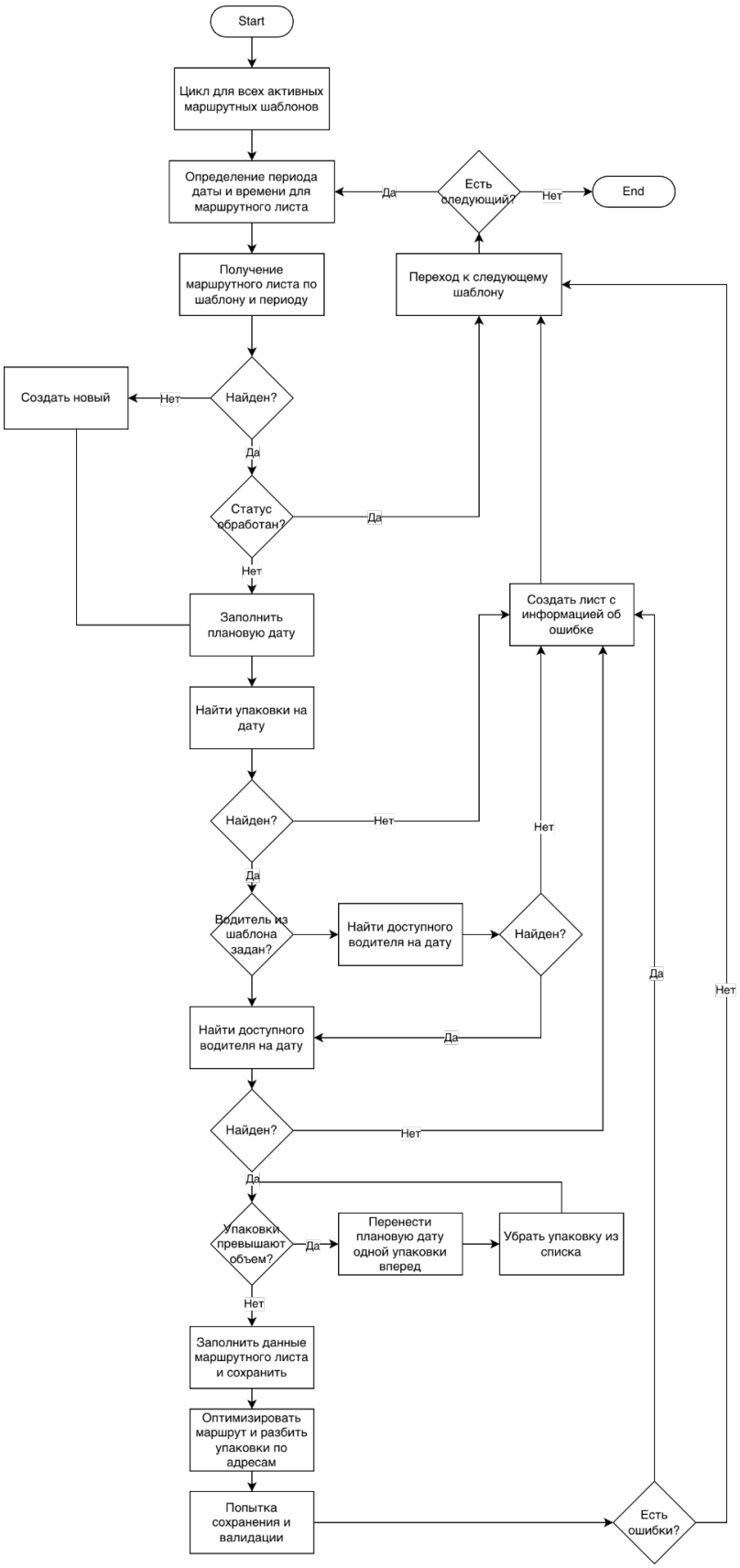


Рисунок 12 – Схема процесса генерации маршрутных листов

По данной схеме был создан класс GenerateRouteSheets, который был настроен как запускаемая фоновая задача раз в определенное время, в нашем случае раз в сутки, это возможно благодаря библиотеке Scheduler для Symfony [22]. Для этого в классе была добавлена аннотация AsPeriodicTask с установкой периода в 1 день (frequency: 1 day), пример настройки класса на рисунке 13:

```php
namespace App\Scheduler\Task;

use ...

#[AsPeriodicTask(frequency: '1 day', method: 'generateRouteSheets')]
class GenerateRouteSheets
{
```

Рисунок 13 – Настройка класса как периодическую задачу

Данную возможность предоставляет модуль Scheduler для Symfony [22], который запускает задачи через фоновый мессенджер. Пример обработки таких задач из описания документации модуля Scheduler представлен на рисунке 14:

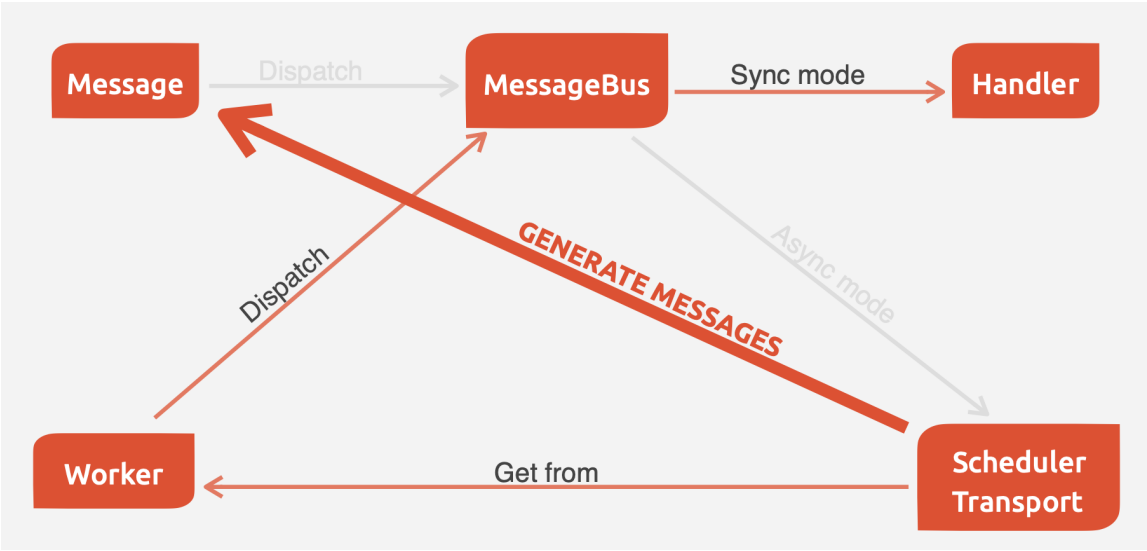


Рисунок 14 – Процесс обработки фоновых задач

Полный пример кода добавлен в Приложении А.

3.3.4 Реализация валидации данных

Для валидации данных были использованы аннотации Doctrine, которые позволяют проверять корректность данных на этапе их отправки на сервер. Были реализованы различные типы проверки данных, такие как обязательность заполнения полей, проверка на уникальность и другие.

Symfony предлагает возможности по созданию собственных типов проверки данных, для примера так задана валидация на максимально допустимую вместимость по объему транспортного средства, пример представлен на рисунке 15:

```php
class VehicleVolumeLimitValidator extends ConstraintValidator  no usages
{
    /**
     * Проверка на основе максимальной допустимой
     * вместимости транспортного средства
     *
     * @param $value
     * @param Constraint $constraint
     *
     * @return void
     */
    public function validate($value, Constraint $constraint)
    {
        // Общая сумма объемов упаковок
        $volumeSum = $this->getCurrentObject()->getVolumeSum();

        if (!$volumeSum) {
            $this->getCurrentObject()->calculateVolumeSum();
            $volumeSum = $this->getCurrentObject()->getVolumeSum();
        }
        $capacity = $this->getCapacity();

        // Если сумма объемов упаковок превышает допустимую
        // вместимость транспортного средства, то ошибка
        if ($volumeSum == 0 || ($capacity && $capacity < $volumeSum)) {
            $this->context->addViolation(
                $constraint->message,
                ['%capacity%' => $capacity, '%volume%' => $volumeSum]
            );
        }
    }

    /**
     * @return RouteSheet
     */
    private function getCurrentObject(): RouteSheet  4 usages
    {
        return $this->context->getObject();
    }

    /**
     * @return float|null
     */
    private function getCapacity(): ?float  1 usage  Vadim Ulyanov
    {
        return $this->getCurrentObject()->getVehicle()?->getCapacity();
    }
}
```

Рисунок 15 – Пример класса проверки данных

3.3.5 Реализация представлений

Для реализации представлений были использованы шаблоны Twig, которые позволяют создавать гибкие и удобные пользовательские интерфейсы. Были реализованы формы и таблицы, которые позволяют отображать данные в удобном для пользователя формате.

Пример реализации twig-шаблона для авторизации пользователя представлен на рисунках 16-17:

```
{% extends 'base.html.twig' %}

{% block title %}Log in!{% endblock %}

{% block body %}
    <form method="post">
        {% if error %}
            <div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
        {% endif %}

        {% if app.user %}
            <div class="mb-3">
                You are logged in as {{ app.user.userIdentifier }}, <a href="{{ path('/logout') }}">Logout</a>
            </div>
        {% endif %}

        <h1 class="h3 mb-3 font-weight-normal">Please sign in</h1>
        <label for="username">Username</label>
        <input type="text" value="{{ last_username }}" name="_username"
               id="username" class="form-control" autocomplete="username" required autofocus>
        <label for="password">Password</label>
        <input type="password" name="_password" id="password"
               class="form-control" autocomplete="current-password" required>

        <input type="hidden" name="_csrf_token"
               value="{{ csrf_token('authenticate') }}"
        >

        <button class="btn btn-lg btn-primary" type="submit">
            Sign in
        </button>
    </form>
{% endblock %}
```

Рисунок 16 – Пример twig-шаблона формы авторизации

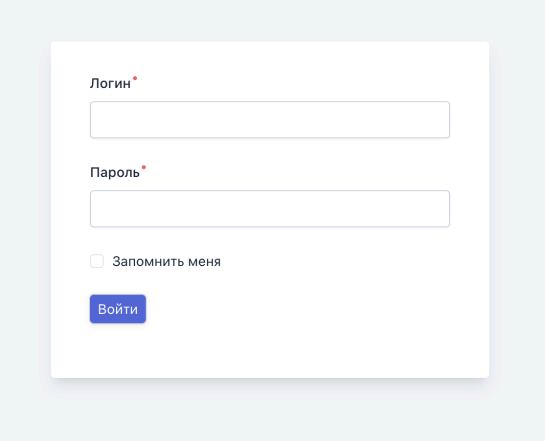


Рисунок 17 – Работа формы авторизации по шаблону

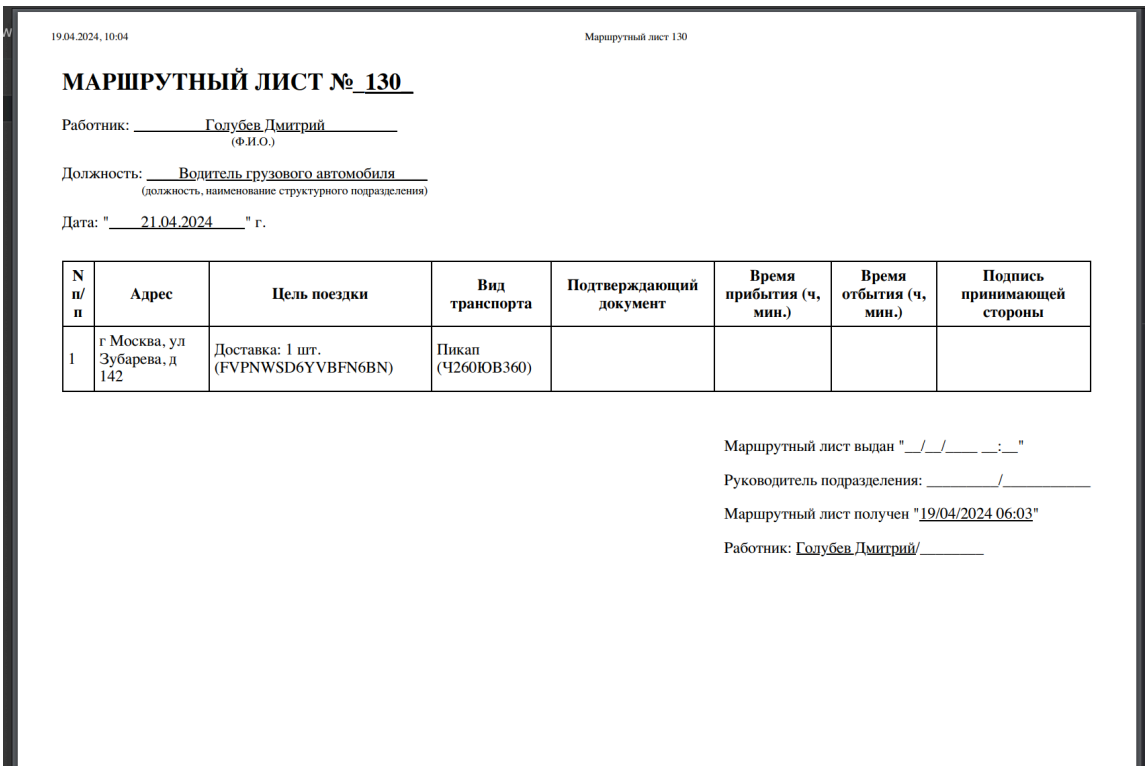
Для генерации pdf-документа маршрутного листа также использован принцип первичной генерации страницы через twig-шаблон и обертывание его отрендеренного html в создание pdf. Пример реализации twig-шаблона для генерации маршрутного листа представлен на рисунках 18-19:

```
<body>
<h2>МАРШРУТНЫЙ ЛИСТ №_<span class="underline">{{ routeSheet.id }}_</span></h2>

<p>Работник: <span class="underline">_________{{ routeSheet.driver.name | default("") }}_________</span></p>
<p class="note" style="...">(Ф.И.О.)</p>
<p>Должность: <span class="underline">____{{ routeSheet.driver.post | default("") }}____</span></p>
<p class="note" style="...">(должность, наименование структурного подразделения)</p>
<p>Дата: "<span class="underline">____{{ routeSheet.planDate|date("d.m.Y") }}____</span>" г.</p>

<table class="route-sheet-table">
    <thead>
    <tr>
        <th>N п/п</th>
        <th>Адрес</th>
        <th>Цель поездки</th>
        <th>Вид транспорта</th>
        <th>Подтверждающий документ</th>
        <th>Время прибытия (ч, мин.)</th>
        <th>Время отбытия (ч, мин.)</th>
        <th>Подпись принимающей стороны</th>
    </tr>
    </thead>
    <tbody>
    {% for routeDestination in routeSheet.destinations %}
        <tr>
            <td>{{ loop.index }}</td>
            <td>{{ routeDestination.destination.address }}</td>
            <td>{{ routeDestination.purpose }}</td>
            <td>{{ routeSheet.vehicle.name }}</td>
            <td></td>
            <td>
                {% if routeDestination.arrivalDateTime is not null %}
                    {{ routeDestination.arrivalDateTime|date("H:i") }}
                {% endif %}
            </td>
            <td>
                {% if routeDestination.leavingDateTime is not null %}
                    {{ routeDestination.leavingDateTime|date("H:i") }}
                {% endif %}
            </td>
            <td></td>
        </tr>
    {% endfor %}
    </tbody>
</table>
<div class="footer">
    <p>Маршрутный лист выдан "__/__/____ __:__"</p>
    <p>Руководитель подразделения: _________/__________</p>
    <p>Маршрутный лист получен "<span class="underline">{{ routeSheet.startDateTime|date("d/m/Y H:i") }}</span>"</p>
```

Рисунок 18 – Пример twig-шаблона генерации маршрутного листа



19.04.2024, 10:04 Маршрутный лист 130

МАРШРУТНЫЙ ЛИСТ № 130

Работник: Голубев Дмитрий
(Ф.И.О.)

Должность: Водитель грузового автомобиля
(должность, наименование структурного подразделения)

Дата: "21.04.2024" г.

N п/п	Адрес	Цель поездки	Вид транспорта	Подтверждающий документ	Время прибытия (ч, мин.)	Время отбытия (ч, мин.)	Подпись принимающей стороны
1	г Москва, ул Зубарева, д 142	Доставка: 1 шт. (FVPNWSD6YVBFN6BN)	Пикап (Ч260ЮВ360)				

Маршрутный лист выдан "__/__/____ __:__"

Руководитель подразделения: _____/_____

Маршрутный лист получен "19/04/2024 06:03"

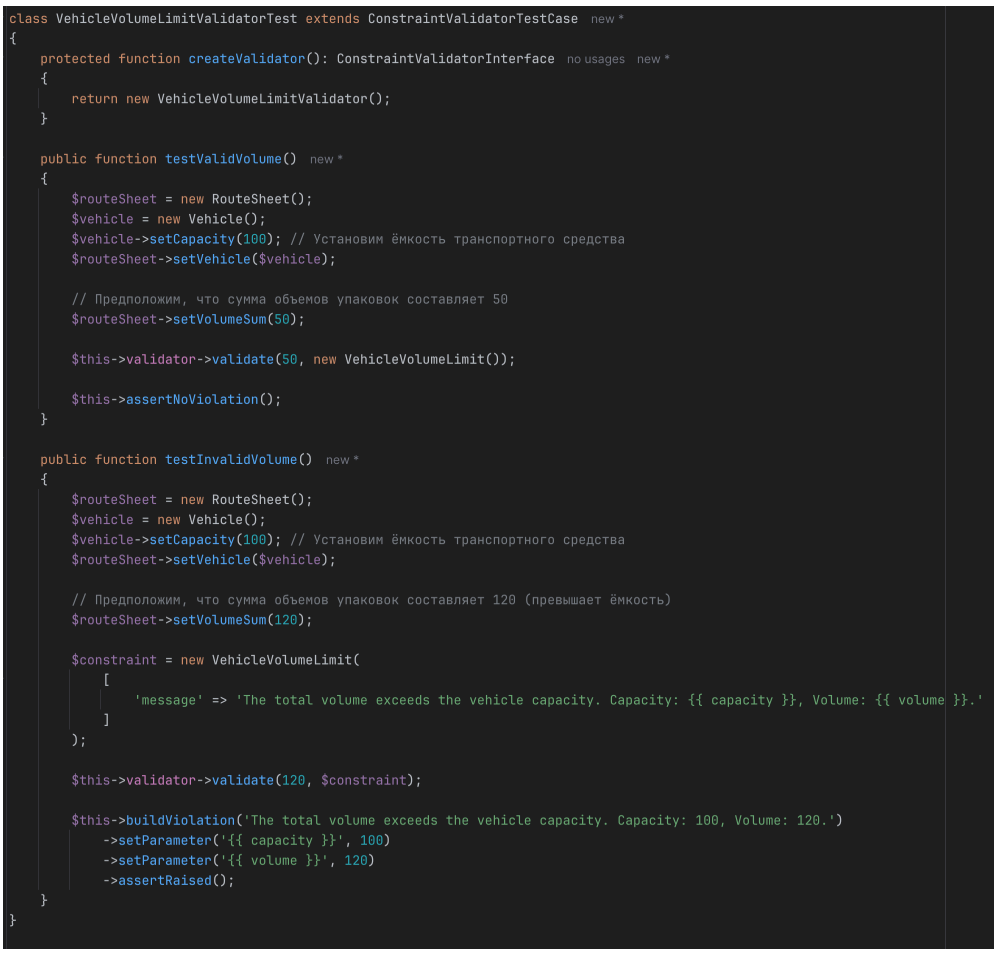
Работник: Голубев Дмитрий/_____

Рисунок 19 – Сгенерированный pdf-документ маршрутного листа на основании шаблона

3.3.6 Реализация тестов

Для проверки корректности работы системы были реализованы автоматические тесты, которые позволяют проверять работу системы на различных уровнях. Были реализованы Unit-тесты на уровне контроллеров, сервисов и представлений.

Пример Unit-теста представлен на рисунке 20:



```php
class VehicleVolumeLimitValidatorTest extends ConstraintValidatorTestCase  new *
{
    protected function createValidator(): ConstraintValidatorInterface  no usages  new *
    {
        return new VehicleVolumeLimitValidator();
    }

    public function testValidVolume()  new *
    {
        $routeSheet = new RouteSheet();
        $vehicle = new Vehicle();
        $vehicle->setCapacity(100); // Установим ёмкость транспортного средства
        $routeSheet->setVehicle($vehicle);

        // Предположим, что сумма объемов упаковок составляет 50
        $routeSheet->setVolumeSum(50);

        $this->validator->validate(50, new VehicleVolumeLimit());

        $this->assertNoViolation();
    }

    public function testInvalidVolume()  new *
    {
        $routeSheet = new RouteSheet();
        $vehicle = new Vehicle();
        $vehicle->setCapacity(100); // Установим ёмкость транспортного средства
        $routeSheet->setVehicle($vehicle);

        // Предположим, что сумма объемов упаковок составляет 120 (превышает ёмкость)
        $routeSheet->setVolumeSum(120);

        $constraint = new VehicleVolumeLimit(
            [
                'message' => 'The total volume exceeds the vehicle capacity. Capacity: {{ capacity }}, Volume: {{ volume }}.'
            ]
        );

        $this->validator->validate(120, $constraint);

        $this->buildViolation('The total volume exceeds the vehicle capacity. Capacity: 100, Volume: 120.')
            ->setParameter('{{ capacity }}', 100)
            ->setParameter('{{ volume }}', 120)
            ->assertRaised();
    }
}
```

Рисунок 20 – Пример Unit-теста класса проверки данных

Также для более ускоренного тестирования был задействован механизм генерации данных на основе модуля DoctrineFixturesBundle [18] для создания «имитационных» данных для тестирования. С помощью него можно создать необходимый количественный и качественный набор валидных данных для тестирования и проверить работу алгоритмов на всем объеме, а также выявить отклонения при тестировании. Список разработанных классов и пример реализации класса для генерации имитационных данных представлены на рисунках 21-22:

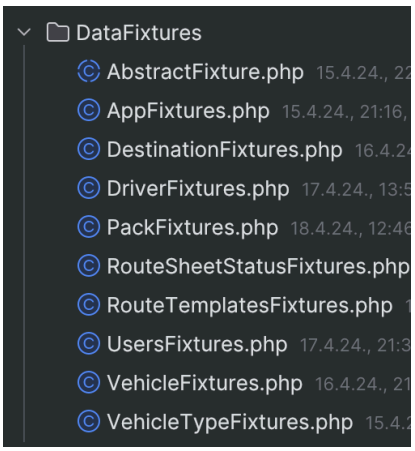


Рисунок 21 – Список классов для генерации имитационных данных

```php
class DestinationFixtures extends AbstractFixture  2 usages  Vadim Ulyanov *
{
    public function load(ObjectManager $manager): void  Vadim Ulyanov
    {
        $items = [];
        $items = [
            ...$items,
            ...array_map(fn() => call_user_func([$this, 'makeEntity'], DestinationType::Warehouse, true), range(0, 0))
        ];
        $items = [
            ...$items,
            ...array_map(fn() => call_user_func([$this, 'makeEntity'], DestinationType::Shop), range(0, 50))
        ];

        $this->saveItems($manager, $items);
    }

    /**
     * @param DestinationType $type
     * @param bool|null $active
     *
     * @return Destination
     */
    private function makeEntity(DestinationType $type, ?bool $active = null): Destination  2 usages  Vadim Ulyanov *
    {
        return (new Destination())
            ->setType($type)
            ->setAddress($this->getRandomAddress())
            ->setLatitude($this->getRandomCord(53, 58))
            ->setLongitude($this->getRandomCord(34, 38))
            ->setActive($active !== null ? $active : $this->getRandomActive());
    }

    /**
     * @return string
     */
    private function getRandomAddress(): string  1 usage  Vadim Ulyanov
    {
        return sprintf(
            'г Москва, %s, д %d',
            $this->getRandomFromArr($this->getAddresses()),
            rand(1, 200)
        );
    }

    private function getAddresses(): array  1 usage  Vadim Ulyanov
    {
        return [
            'пр-кт Мира',
            'ул Верхняя Красносельская',
            'Ленинский пр-кт',
            'ул Мартынова',
            'ул Зубарева',
            'пр-кт Вернадского',
            'ул Шишкина',
        ];
    }
```

Рисунок 22 – Пример класса генерации данных для сущности Destination

3.4 Обеспечение безопасности

При разработке системы необходимо предусмотреть механизмы аутентификации и авторизации пользователей. Аутентификация – это процесс проверки подлинности пользователя, то есть установления того, что пользователь действительно является тем, кто он заявляет о себе. Авторизация – это процесс проверки разрешений пользователя на доступ к определенным ресурсам или функциям системы.

Существуют различные методы аутентификации, такие с помощью ввода логина и пароля, двухфакторная аутентификация, аутентификация по биометрическим данным и т.д. Выбор метода аутентификации зависит от требований безопасности системы и удобства использования для пользователей.

Symfony предоставляет доступ к библиотеке Symfony SecurityBundle [23], которая позволяет добавлять конфигурации безопасности, управления уровнями доступа и др. В нашей системе мы воспользовались ей для создания конфигурации аутентификации и авторизации. Как для пользователей административной панели, так и для API, выбран способ аутентификации по логину и паролю.

Для пользователей административной системы реализована страница авторизации в веб-приложении. А для пользователей, которые обращаются извне к системе, будет использован способ общения по REST-API, таким образом любая система, которая интегрируется с системой, будет общаться с ней в режиме запрос-ответ. API разработанной системы создано с помощью библиотеки The API Platform Core Library [26], которая предоставляет широкие возможности по созданию API любой сложности. Swagger/OpenAPI поддерживает поддержку CRUD-модели и позволяет автоматически генерировать техническую документацию для внешних систем.

API для доступа к ресурсам системы при вызове того или иного REST-запроса ограничено аутентификацией, требуется заголовок Authorization с типом авторизации Basic, пример:

Authorization: Basic YWRtaW46U3Ryb25nUGFzc3dvcmQxMjMh

Для управления доступом к ресурсам в реализуемой системе созданы роли пользователей, обозначенные ниже.

Для пользователей группы водителей определена роль – ROLE_DRIVER, функциональные возможности:

а) просмотр списка маршрутных листов;

б) частичное редактирование маршрутных листов:

1) статус маршрутного листа;

2) задание даты начала выполнения рейса;

3) задание даты прибытия и убытия с точек доставки;

в) печать маршрутного листа.

Для пользователей группы менеджеров-логистов определена роль ROLE_LOGISTICS, функциональные возможности:

а) планирование маршрутов перевозок:

1) формирование шаблонов маршрутов;

2) формирование маршрутных листов – ручная и автоматическая;

б) ведение справочников системы:

1) справочник складов/пунктов назначения;

2) справочник транспортных средств;

3) справочник водителей.

Для пользователей внешних систем определена роль ROLE_INTEGRATION, функциональные возможности:

а) прием информации о грузах на перевозку;

б) передача информации по маршрутным листам (для аналитических и отчетных систем).

Для администраторов системы определена роль ROLE_SUPER_ADMIN, функциональные возможности:

а) все возможности всех других ролей:

б) дополнительно:

1) генерация данных для тестирования системы;
2) генерация пользователей для водителей на основании списка сущности Водитель.

Код реализации API-аутентификатора ApiAuthAuthenticator добавлен в Приложении Б.

3.5 Разработка пользовательского интерфейса

Для разработки интерфейса пользователя также используется подход к созданию сущностей через CRUD-модели и CRUD-контроллеры. Также, как упоминалось ранее, формы редактирования для разных групп пользователей могут ограничиваться ролями доступа, которые задаются у каждого пользователя системы, в предыдущем подразделе были описаны заданные роли.

Пример кода реализованного CRUD-контроллера представлен в Приложении В.

Далее будут описаны интерфейсные формы административной панели.

3.5.1 Форма авторизации

На рисунке 23 изображена форма авторизации, при помощи которой пользователи могут получить доступ к пользовательскому интерфейсу для взаимодействия с системой.

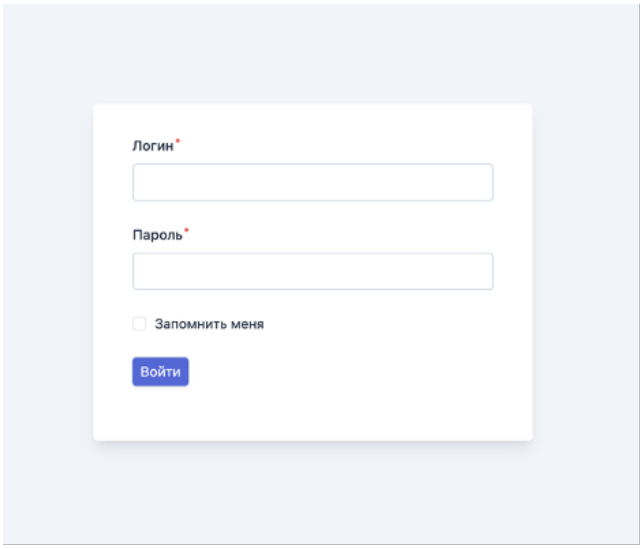


Рисунок 23 – Форма авторизации

3.5.2 Страница «Маршрутные листы»

Данная страница является базовой, после авторизации. Форма отображена на рисунке 24.

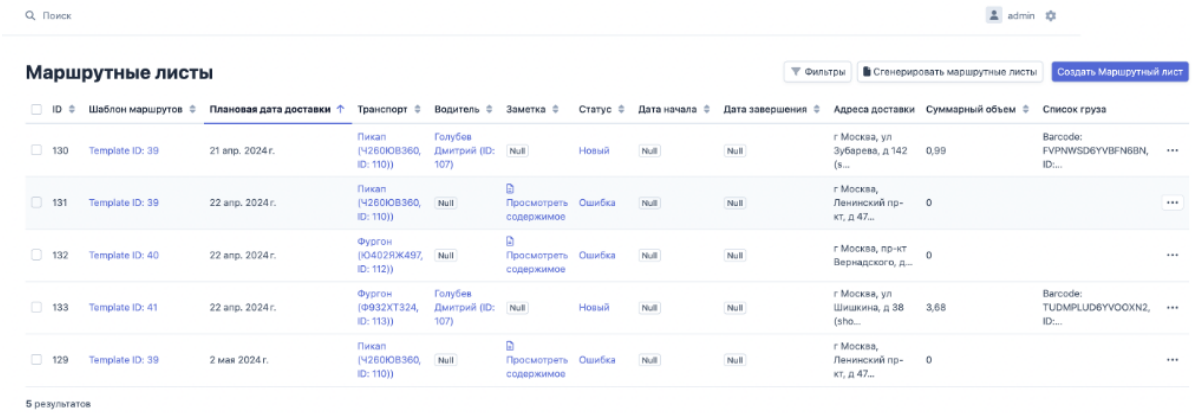


Рисунок 24 – Страница «Маршрутные листы»

На данной странице выводится список сгенерированных маршрутных листов, а также детальная информация по ним.

Другие функциональные возможности страницы:

– фильтрация списка;

– сортировка по полям;

– генерация новых маршрутных листов на основе существующего шаблона для определенных дат;
– создание нового маршрутного листа с ручным заполнением всех полей;
– редактирование записи;
– удаление записи;
– пагинация.

3.5.3 Страница редактирования маршрутного листа

Страница открывается при редактировании маршрутного листа. Формы отображены на рисунках 25-26.

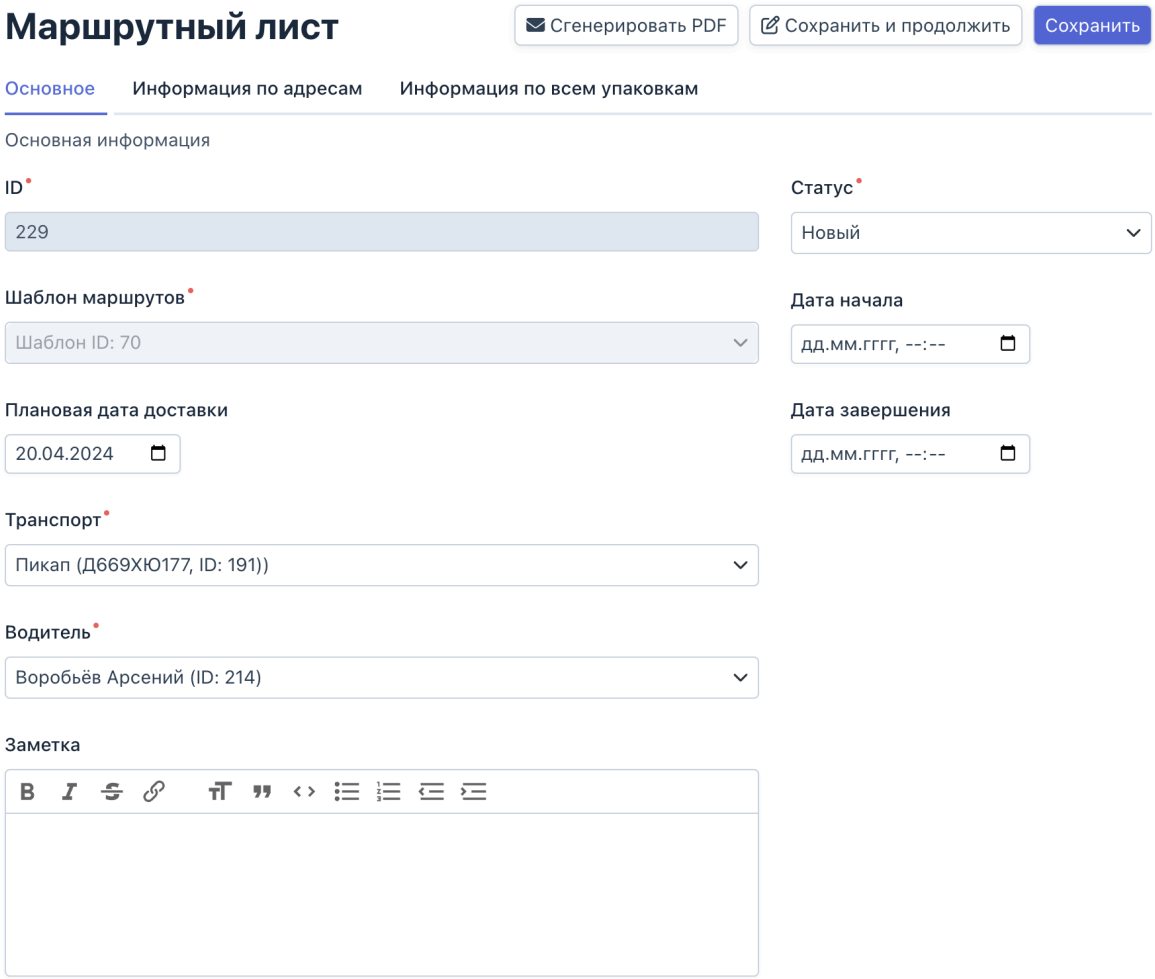


Рисунок 25 – Страница редактирования маршрутного листа, раздел «Основное»

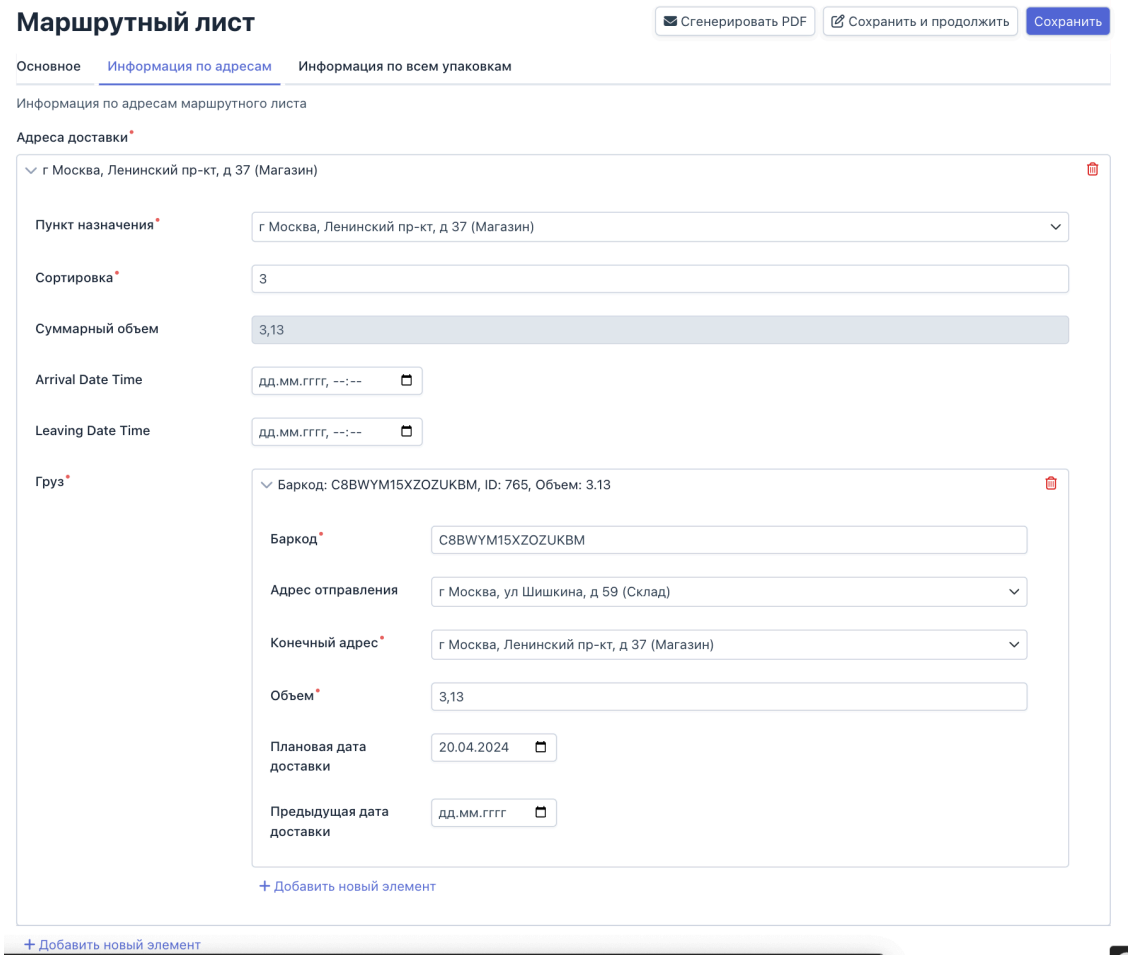


Рисунок 26 – Страница редактирования маршрутного листа, раздел «Информация по адресам»

Функциональные возможности страницы:

– задание шаблона маршрута;
– управление статусом;
– задание плановой даты листа;
– задание даты начала и завершения листа;
– задание транспортного средства и водителя;
– добавление информации по адресам и грузам;
– генерация pdf документа для печати.

3.5.4 Шапка страниц

Шапка для каждой страницы используется одна и та же. Форма изображена на рисунке 27.

Рисунок 27 – Шапка страниц

Функциональные возможности страницы:

– возможность поиска по ключевым словам на любой доступной странице;
– выход из системы;
– изменение оформления интерфейса.

3.5.5 Страница «Груз»

На данной странице выводится список посылок на перемещение, полученных из внешних систем через интеграцию по API. Страница изображена на рисунке 28.

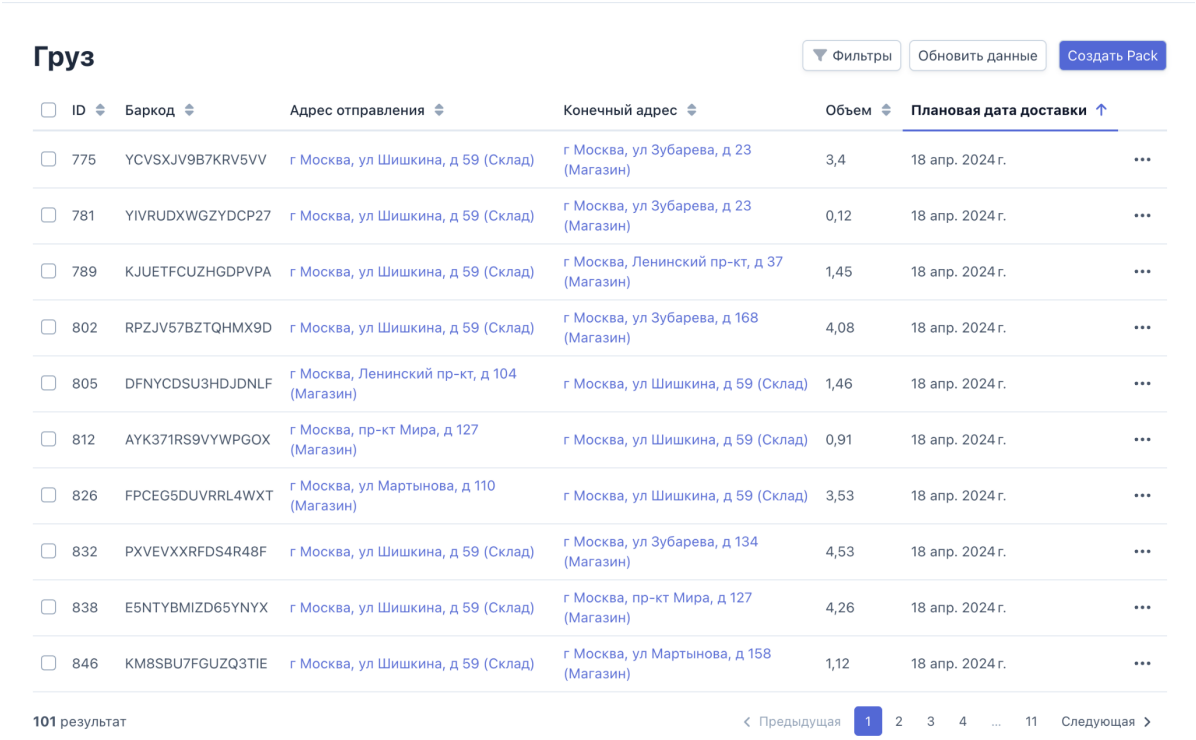


Рисунок 28 – Страница «Груз»

Функциональные возможности страницы:

– фильтрация списка;
– сортировка по полям;

– обновление данных (функция предназначена для полного сброса и создания новых тестовых данных в целях тестирования);
– создание нового груза;
– редактирование записи;
– удаление записи;
– пагинация.

3.5.6 Страница «Шаблоны маршрутов»

На этой странице выводится список ранее созданных шаблонов маршрутов. Страница изображена на рисунке 29.

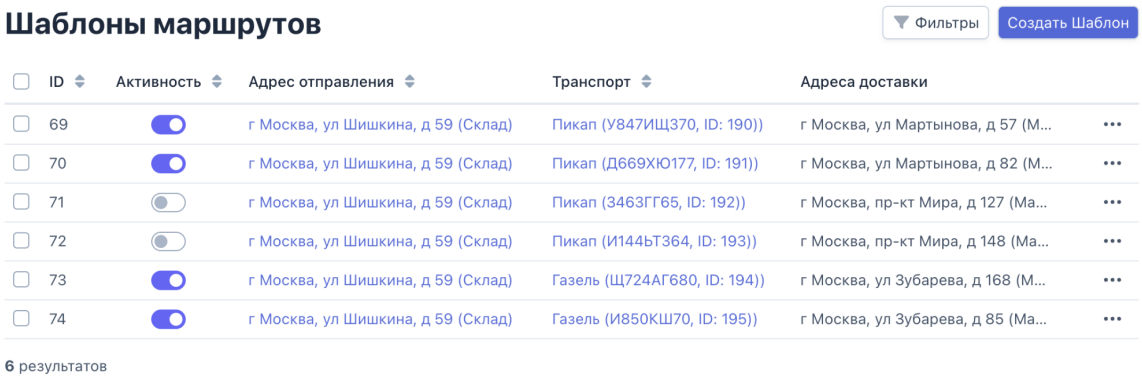


Рисунок 29 – Страница «Шаблоны маршрутов»

Функциональные возможности страницы:
– фильтрация списка;
– создание нового шаблона;
– сортировка по полям;
– управление активностью шаблона;
– редактирование шаблона;
– удаление записи;
– пагинация.

3.5.7 Страница «Статусы маршрутных листов»

На странице «Статусы маршрутных листов» ведется список статусов для маршрутных листов. Страница изображена на рисунке 30:

Рисунок 30 – Страница «Статусы маршрутных листов»

Функциональные возможности страницы:

– создание нового статуса;

– сортировка по полям;

– удаление записи.

3.5.8 Страница «Адреса доставки»

На данной странице выведен список и детальная информация по адресам доставки, на которые доступно осуществление логистики. Страница изображена на рисунке 31:

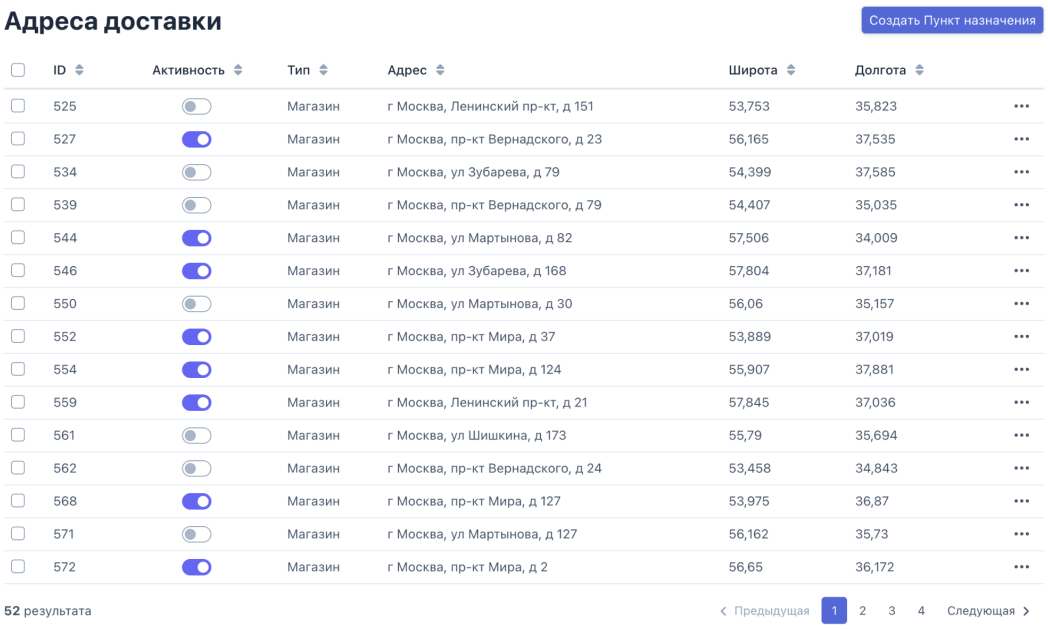


Рисунок 31 – Страница «Адреса доставки»

Функциональные возможности страницы:

– создание нового пункта назначения;
– сортировка по полям;
– редактирование существующего адреса;
– управление активностью пункта назначения;
– удаление записи;
– пагинация.

3.5.9 Страница «Транспорт».

На странице «Транспорт» выводится список заведенных транспортов и детальная информация по ним. Страница изображена на рисунке 32:

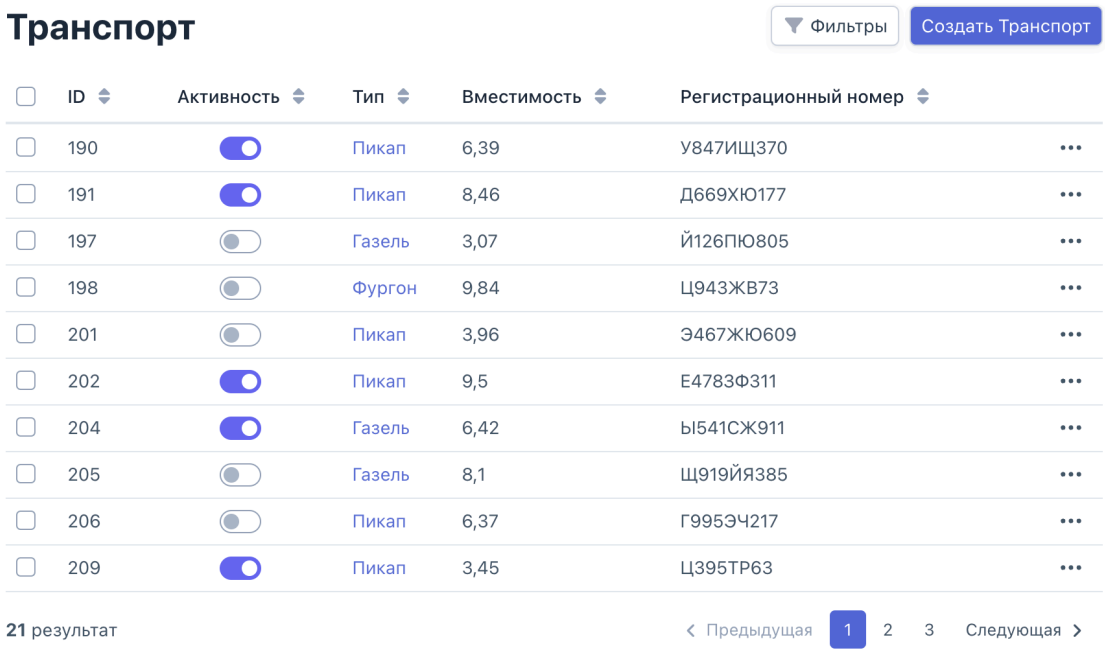


Рисунок 32 – Страница «Транспорт»

Функциональные возможности страницы:

– создание нового транспорта;
– сортировка по полям;
– фильтрация;
– редактирование существующего транспорта;
– управление активностью транспорта;

– удаление записи;

– пагинация.

3.5.10 Страница «Тип транспорта».

На данной странице ведется список доступных типов транспорта. Страница изображена на рисунке 33:

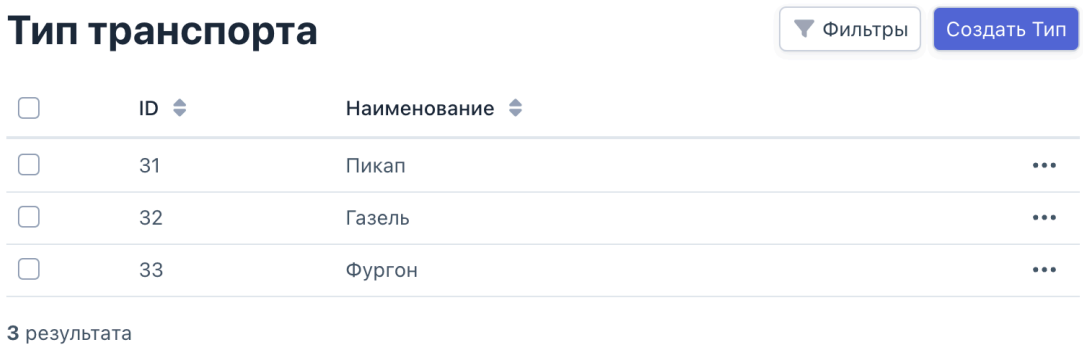


Рисунок 33 – Страница «Тип транспорта»

Функциональные возможности страницы:

– создание нового типа транспорта;

– сортировка по полям;

– фильтрация;

– редактирование существующего типа;

– удаление записи.

3.5.11 Страница «Водители».

На странице «Водители» ведется справочник, созданных в системе водителей. Страница изображена на рисунке 34:

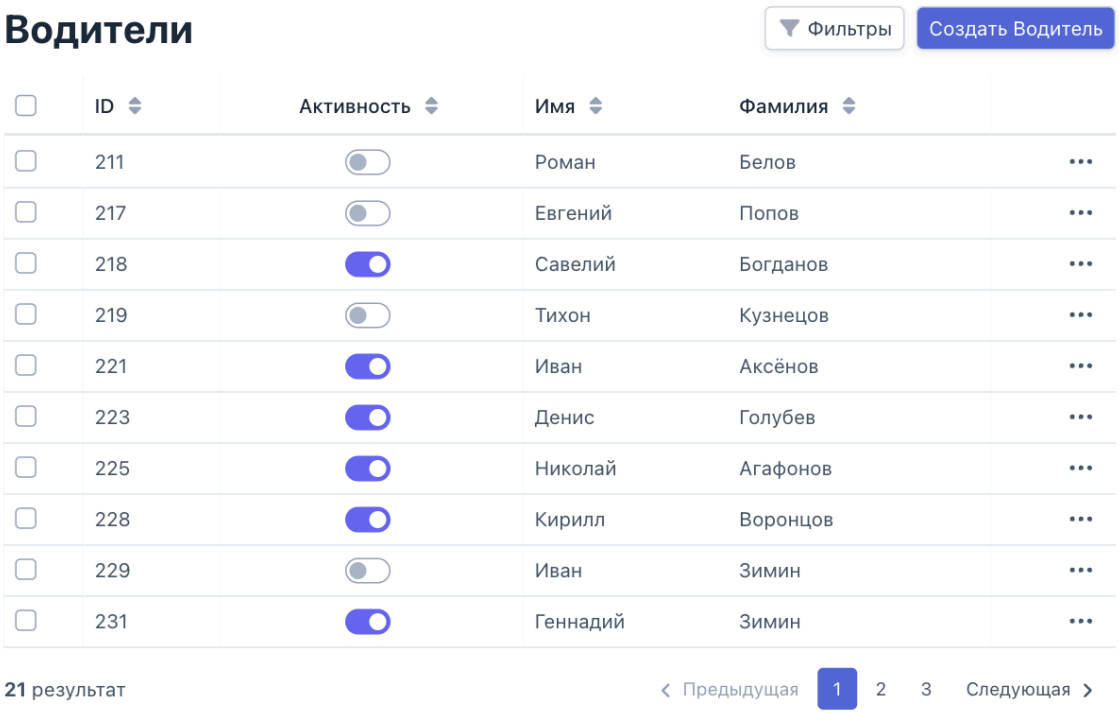


Рисунок 34 – Страница «Водители»

Функциональные возможности страницы:

– создание нового водителя;
– сортировка по полям;
– фильтрация;
– редактирование существующего водителя;
– управление активностью водителя;
– удаление записи;
– пагинация.

3.5.12 Страница «Пользователи»

На странице ведется список пользователей, у которых имеется доступ к системе. Страница изображена на рисунке 35:

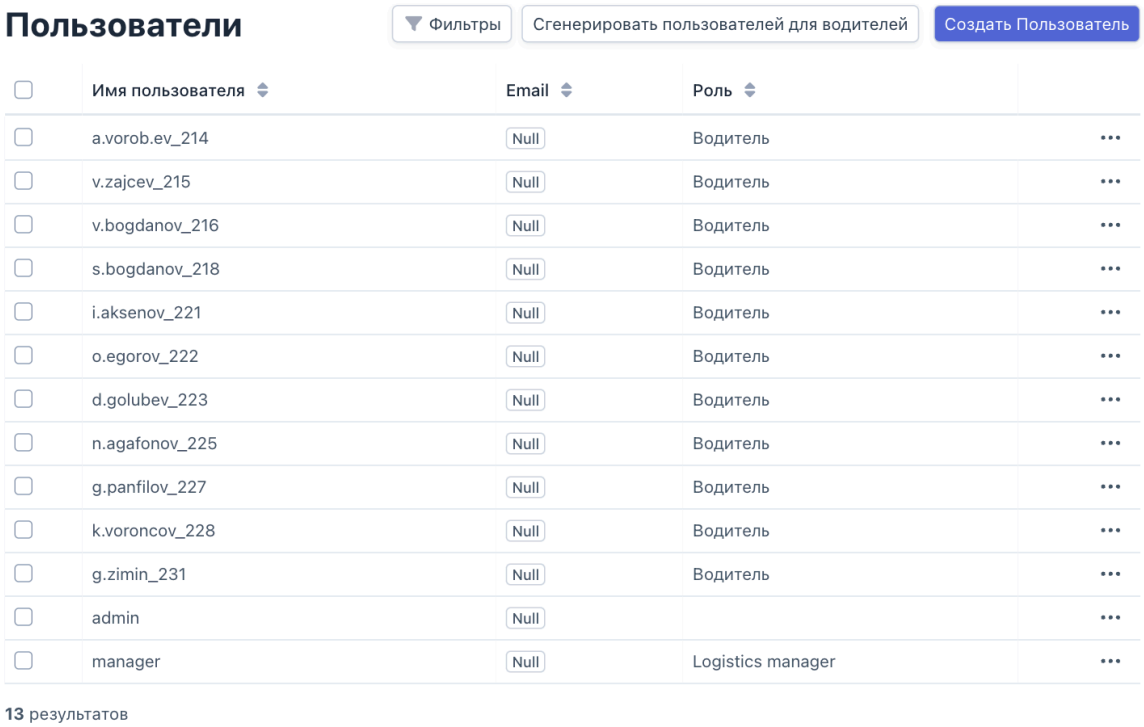


Рисунок 35 – Страница «Пользователи»

Функциональные возможности страницы:

– создание нового пользователя;
– генерация новых пользователей по созданным водителям (для возможности авторизации водителям в системе и просмотру маршрутных листов);
– сортировка по полям;
– фильтрация;
– редактирование существующего пользователя;
– удаление записи;
– пагинация.

3.6 Разработка интерфейса для взаимодействия с внешними системами

3.6.1 Реализация API для взаимодействия с внешними системами

Как упоминалось ранее, для Symfony есть возможность использования открытой библиотеки создания API - The API Platform Core Library [26]. Библиотека предоставляет гибкие возможности автоматического создания API методов для CRUD-моделей.

Для добавления возможности создания управления сущностей по API нам необходимо добавить определенные аннотации для сущности, такие как ApiResource или конкретные для необходимых REST-методов Get, GetCollection, Post и др. Пример добавленных аннотаций приведен на рисунке 36:

```
#[ORM\Entity(repositoryClass: PackRepository::class)]
#[UniqueEntity('barcode')]
#[GetCollection(
    normalizationContext: ['groups' => 'integration:list'],
    security: "is_granted('ROLE_INTEGRATION')",
    securityMessage: 'Only integration users can get list of packs.'
)]
#[Get(
    uriTemplate: '/pack',
    normalizationContext: ['groups' => 'integration:get'],
    security: "is_granted('ROLE_INTEGRATION')",
    securityMessage: 'Only integration users can get packs.'
)]
#[Post(
    uriTemplate: '/pack',
    denormalizationContext: ['groups' => 'integration:create'],
    security: "is_granted('ROLE_INTEGRATION')",
    securityMessage: 'Only integration users can create packs.'
)]
#[ApiFilter(DateFilter::class, properties: ['planDate', 'factDate'])]
#[ApiFilter(RangeFilter::class, properties: ['volume'])]
class Pack implements \Stringable
{
    #[ORM\Id] 1 usage
    #[ORM\GeneratedValue]
    #[ORM\Column]
    #[Groups(['integration:list', 'integration:get'])]
    private ?int $id = null;
```

Рисунок 36 – Атрибуты для конфигурации сущности для API

При передаче информации по API есть возможность ограничивать те или иные атрибуты сущности, которые используются в схеме создания/получения данных, используя группы, которые задаются для каждого поля с помощью аннотации Groups. Также при обработке сущности немаловажно использовать валидацию данных, здесь мы тоже воспользуемся возможностью аннотаций и добавим валидацию при создании сущности, пример добавленных аннотаций на рисунке 37:

Рисунок 37 – Аннотации ограничения и валидации по API

В результате описанных действий система дает нам возможность схему API запросов, валидацию данных на этапе обработки данных по API, а также,

что немаловажно, сгенерировать техническое описание API для внешних систем, которым смогут воспользоваться разработчики, которые будут интегрироваться с системой извне. Техническая документация доступна с помощью Swagger UI, пример доступа к автоматически сгенерированной документации сервиса на рисунке 38:

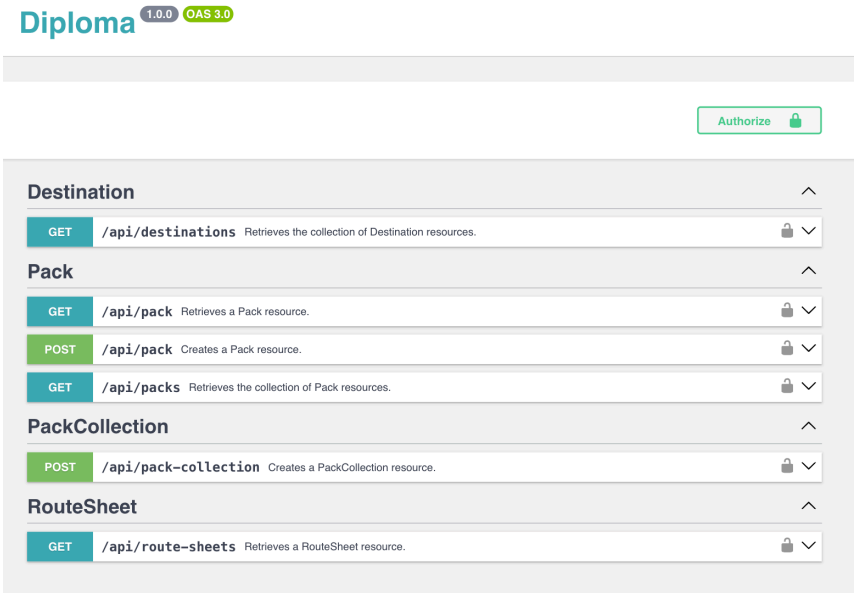


Рисунок 38 – Техническая документация, список доступных методов

3.6.2 Описание запросов

Далее опишем адреса конечных точек API, которые мы разработали для нашего проекта.

а) Создание грузов на перевозку:

1) POST /api/pack – добавление груза
2) POST /api/pack-collection – добавление коллекции грузов, для возможности передачи нескольких сущностей за раз

б) Получение упаковок:

1) GET /api/packs – получение коллекции упаковок
2) GET /api/pack/{id} – получение упаковки по id

в) Получение маршрутных листов:

1) GET /api/route-sheets – получение коллекции упаковок

2) GET /api/route-sheet/{id} – получение конкретного маршрутного листа по id

3.6.3 Примеры выполнения запросов

Среди разработчиков для тестирования запросов широко используется программа Postman. Воспользуемся ей для тестирования всех наших точек доступа API.

Ниже представлены примеры выполнения успешных запросов на рисунках 39-41.

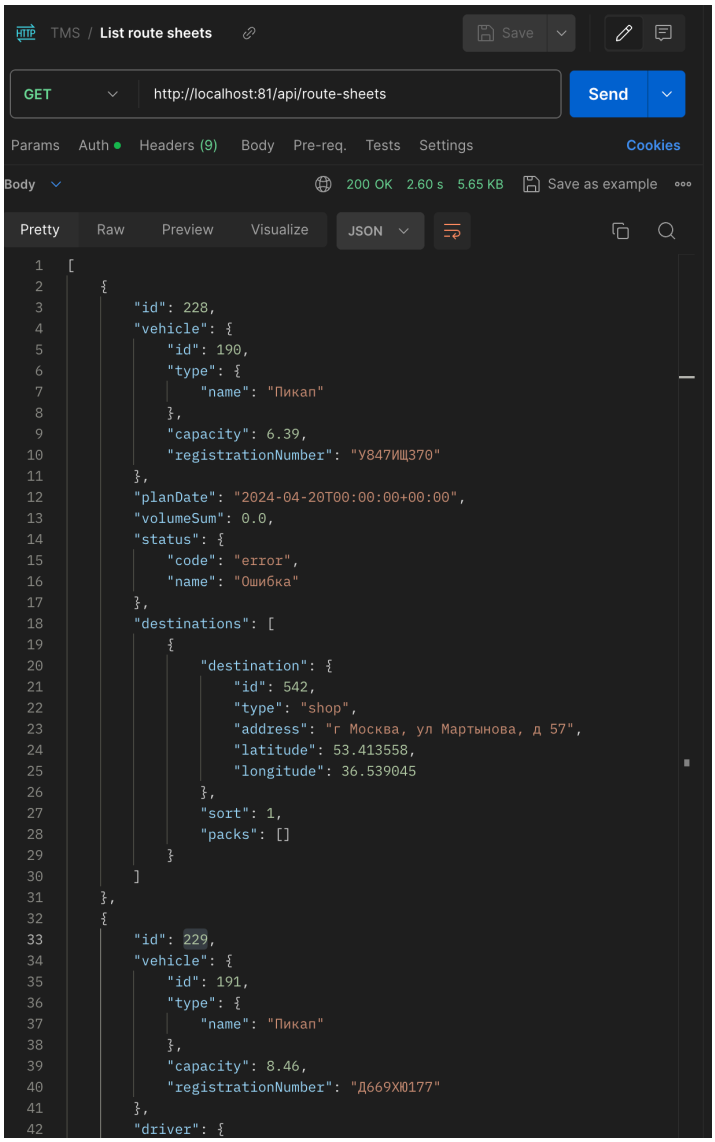


Рисунок 39 – Пример получения списка маршрутных листов списка

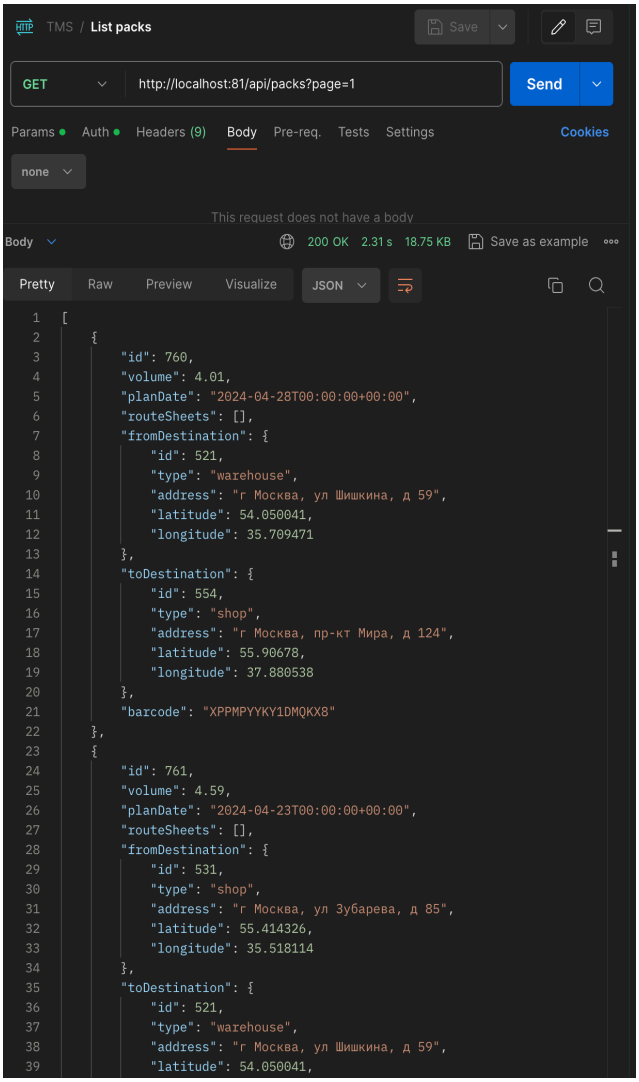


Рисунок 40 – Пример получения списка упаковок

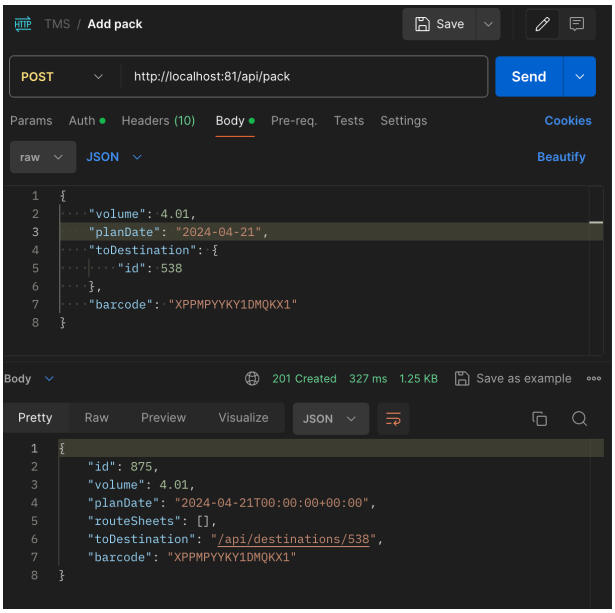


Рисунок 41 – Пример создания упаковок

Также при отправке некорректных данных выдается список ошибок, пример на рисунке 42:

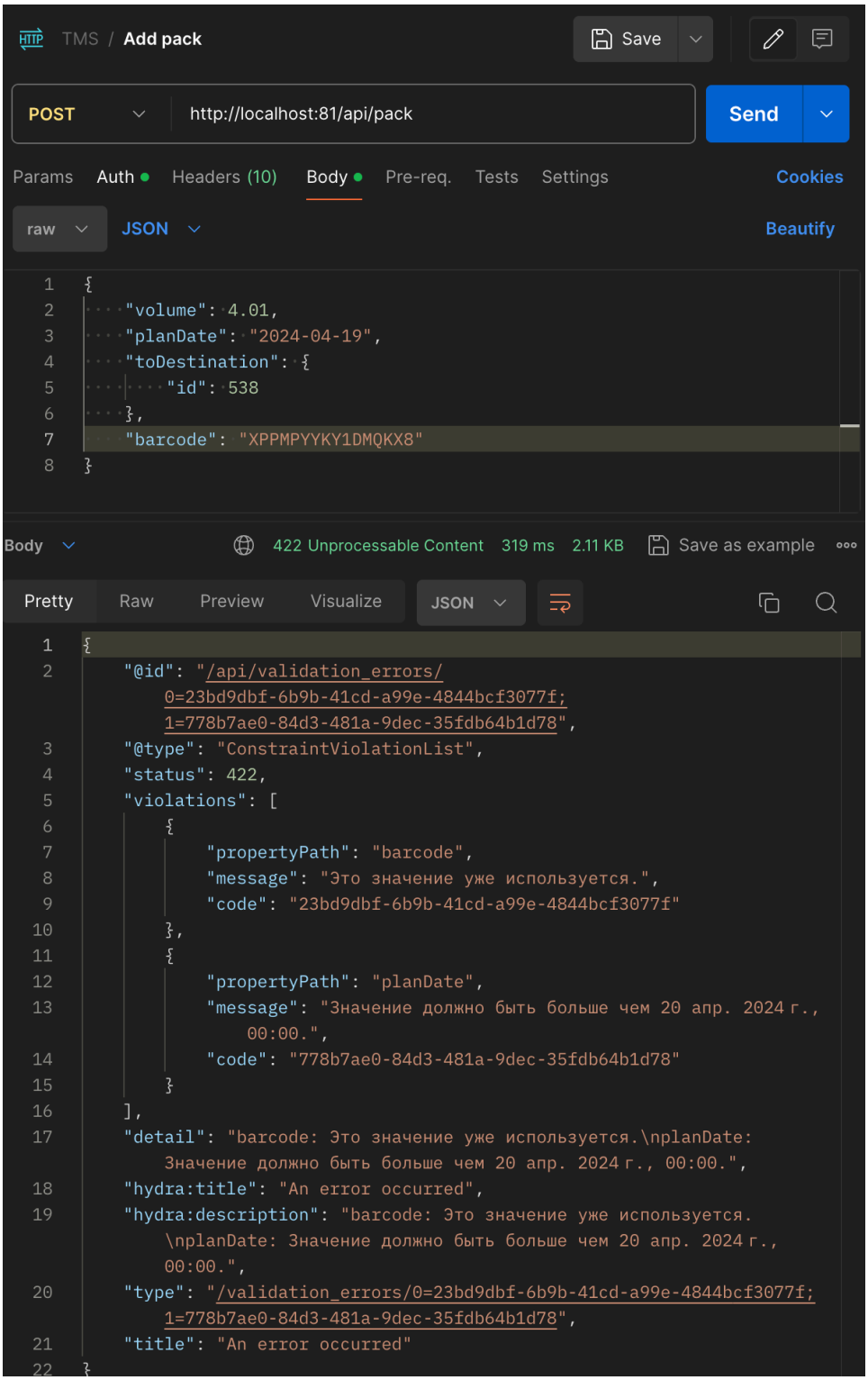


Рисунок 42 – Пример ошибки при проверке данных при создании упаковок

3.7 Разработка контрольного примера тестирования системы

При разработке, как указывалось ранее, использован подход генерации имитационных данных с помощью модуля DoctrineFixturesBundle.

Контрольным примером при тестировании можно выделить создание всех необходимых сущностей для основного процесса системы – формирования маршрутного листа.

Контрольный пример данных для тестирования представлен в таблицах 2-6.

Таблица 2 – Пример данных сущности «Упаковка (Pack)»

	Идентификатор упаковки	Объем упаковки, м³	Планируемая дата доставки	Баркод	Пункт назначения
Упаковка 1	1648	0.74	2024-04-22	YA2LP8CGZ VQYETWU	г Москва, пр-кт Мира, д 5
Упаковка 1	1683	1.8	2024-04-22	MRTSNDIPD UQB3T6Y	г Москва, ул Шишкина, д 82

Таблица 3 – Пример данных сущности «Транспортное средство (Vehicle)»

	Идентификатор транспортного средства	Тип транспортного средства	Вместимость, м³	Регистрационный номер	Признак активности
Транспортное средство 1	358	Пикап	6.69	Д699ЩЙ811	Активно
Транспортное средство 2	363	Пикап	4.89	У39ЖШ498	Активно
Транспортное средство 3	113	Фургон	9.84	Ф932ХТ	Неактивно

Таблица 4 – Пример данных сущности «Тип транспортного средства (VehicleType)»

	Идентификатор типа	Название типа
Тип 1	1	Грузовик
Тип 2	2	Фургон
Тип 3	3	Пикап

Таблица 5 – Пример данных сущности «Водитель (Driver)»

	Идентификатор водителя	Имя водителя	Телефон водителя	Признак активности
Водитель 1	379	Шубин Борис	+79123456789	Активен
Водитель 2	382	Ширяев Роман	+79234567890	Активен
Водитель 3	290	Алексей Михайлов	+79345678901	Неактивен

Таблица 6 – Пример данных сущности «Пункт назначения (Destination)»

	Идентификатор пункта назначения	Тип пункта назначения	Адрес пункта назначения	Признак активности
Пункт назначения 1	937	Склад	г Москва, ул Зубарева, д 197	Активен
Пункт назначения 2	977	Магазин	г Москва, пр-кт Мира, д 5	Активен
Пункт назначения 3	979	Магазин	г Москва, ул Шишкина, д 82	Активен
Пункт назначения 4	941	Магазин	г Москва, пр-кт Мира, д 196	Активен
Пункт назначения 5	949	Магазин	г Москва, ул Верхняя Красносельская, д 6	Активен

Таблица 7 – Пример данных сущности «Шаблон маршрута (RouteTemplate)»

	Идентификатор шаблона маршрута	Пункт отправления (склад)	Назначенное транспортное средство	Список пунктов назначения	Признак активности
Шаблон 1	128	Склад (id: 937)	358	941, 977 (id адресов магазинов)	Активен
Шаблон 2	133	Склад (id: 937)	363	949, 979 (id адресов магазинов)	Активен

При запуске генерации маршрутных листов созданы два маршрутных листа. Обратим внимание, что в шаблонах были заданы по 2 пункта назначения, но в итоговом маршрутном листе добавлены только по одному пункту назначения, так как доставка в этот день предполагается только по адресам, на которые нужно доставить упаковки. Маршрутные листы сформированы посредством автогенерации через алгоритм, данные представлены в таблице 3.

Таблица 8 – Контрольный пример информации сгенерированного маршрутного листа

	Идентификатор маршрутного листа	Шаблон маршрута, id	Транспортное средство, id	Водитель, id	Общий объем груза, м³	Список упаковок, шт.	Список пунктов назначения, id	Статус
Маршрутный лист 1	294	128	358	379	0.74	1	977	Новый
Маршрутный лист 2	296	133	363	382	1.8	1	979	Новый

Также ниже пример маршрутного листа на рисунках 44-46:

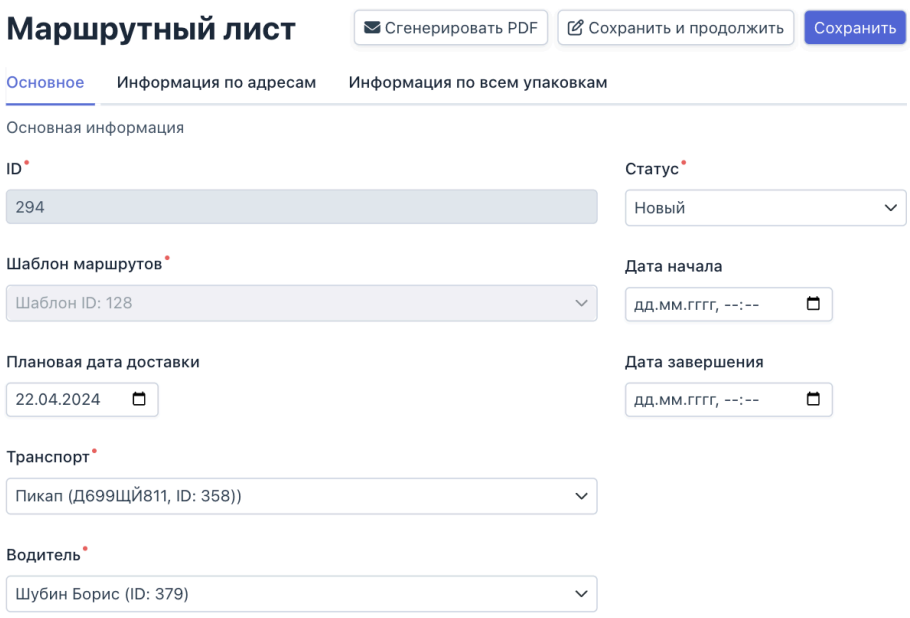


Рисунок 43 – Маршрутный лист id 294, основная информация

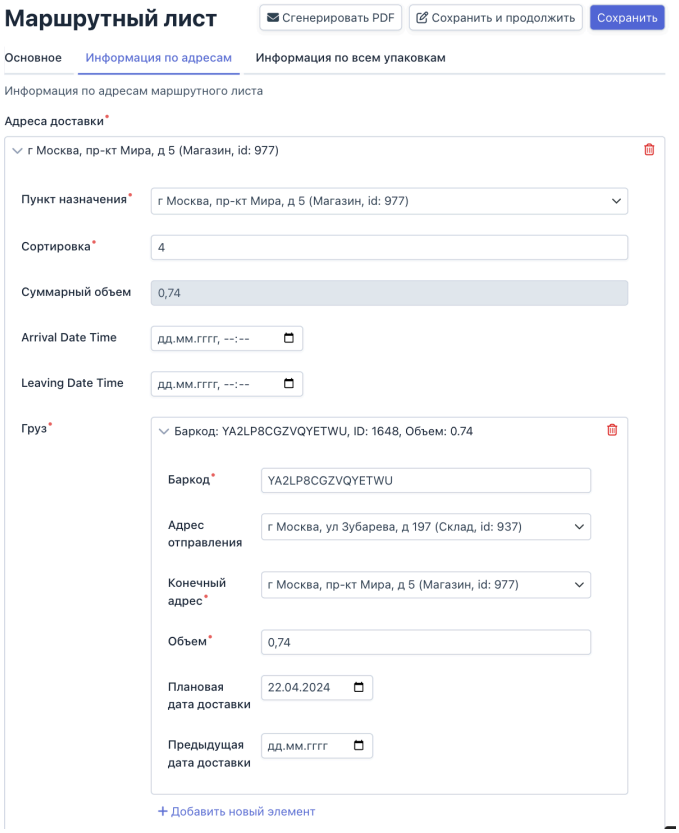
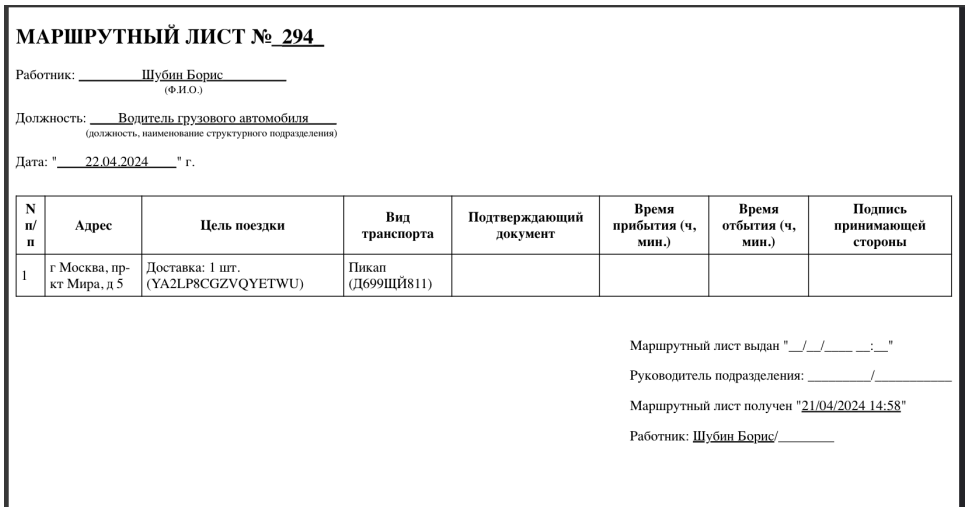


Рисунок 44 – Маршрутный лист id 294, адреса и упаковки



МАРШРУТНЫЙ ЛИСТ № 294

Работник: Шубин Борис
(Ф.И.О.)

Должность: Водитель грузового автомобиля
(должность, наименование структурного подразделения)

Дата: "22.04.2024" г.

N п/п	Адрес	Цель поездки	Вид транспорта	Подтверждающий документ	Время прибытия (ч, мин.)	Время отбытия (ч, мин.)	Подпись принимающей стороны
1	г Москва, пр-кт Мира, д 5	Доставка: 1 шт. (YA2LP8CGZVQYETWU)	Пикап (Д699ЩЙ811)				

Маршрутный лист выдан "__/__/____ __:__"

Руководитель подразделения: _____/_____

Маршрутный лист получен "21/04/2024 14:58"

Работник: Шубин Борис/_____

Рисунок 45 – PDF-документ маршрутного листа id 294

На основании приведенных результатов обработки данных мы видим, что маршрутные листы были сгенерированы корректно, с использованием только тех данных, которые были активны и валидны. Данный способ генерации подтверждает и оправдывает необходимость автоматизации

управления логистических процессов, так как при увеличении количества информации, количества процессов на доставку, увеличивается емкость и сложность логистических процессов, которыми необходимо эффективно управлять и оптимизировать.

Разработанная система с высокой производительностью обрабатывает данные, генерирует данные без ошибок, оптимизирует маршрутные листы на основании необходимости доставки нужного количества упаковок в тот или иной день, оптимизирует порядок доставки и др. Все это позволяет сделать вывод о том, что система пригодна для использования теми ролями пользователей, которые определены в системе.

В качестве улучшений системы в будущем можно привести следующие доработки:

- отображение маршрутов на карте;
- отслеживание транспорта на карте в реальном времени;
- оптимизация маршрутов в зависимости от ситуации на дорогах по информации из внешних источников (например, Яндекс.Карты).

Вывод к главе 3

В третьей главе была определена архитектура для автоматизированной системы транспортной логистики, определены ключевые функциональные возможности системы и создана диаграмма бизнес-процессов.

После выявления бизнес-процессов была спроектирована модель данных и определены ключевые сущности системы. На основании данных реализованы физические модели сущностей и создана необходимая структура физической базы данных PostgreSQL.

На основании всей информации далее была разработана бизнес-логика обработки данных сущностей, генерации данных, а также разработан пользовательский интерфейс для пользователей системы.

Важным этапом было обеспечение безопасности, а именно закрытие доступа к системе на основании аутентификации пользователей, авторизации и управлении уровнями доступа на основании ролей.

Также для взаимодействия с внешними системами был определен способ общения – REST-API, наиболее используемый способ интеграции между системами, с использованием Basic-авторизации.

В конце главы были предоставлены примеры входящей информации в систему в виде данных необходимых сущностей и примеры обработки этой информации, а именно создания маршрутных листов на основании процесса генерации.

Заключение

В ходе выполнения дипломной работы был проведен анализ предметной области транспортной логистики, сравнительный анализ инструментов разработки программного обеспечения, а также разработка автоматизированной системы управления транспортной логистикой.

В результате анализа предметной области было выявлено, что транспортная логистика играет ключевую роль в обеспечении эффективного перемещения материальных потоков в цепочках поставок. Существующие программные решения для управления транспортной логистикой ориентированы на крупные предприятия, что создает барьеры для использования их малым и средним бизнесом. Таким образом, выявлена потребность в разработке системы, которая сможет предоставить определенные преимущества по сравнению с упомянутыми в тексте более сложными системами для данного бизнес-сектора. Требования к разрабатываемой системе включают в себя планирование маршрутов, управление транспортными средствами, отслеживание грузов, интеграцию с другими системами, а также масштабируемость, надежность, безопасность и удобство использования.

После проведения сравнительного анализа инструментов разработки было принято решение использовать веб-разработку для обеспечения кроссплатформенного доступа к системе через веб-браузер на различных устройствах. Выбор стека технологий (PHP, Symfony, PostgreSQL) обоснован современными тенденциями веб-разработки, обеспечивая необходимую производительность, масштабируемость, безопасность и кроссплатформенность.

В ходе разработки автоматизированной системы управления транспортной логистикой была определена архитектура системы, спроектирована модель данных, разработана бизнес-логика обработки данных, обеспечена безопасность системы, а также создан пользовательский

интерфейс и программный API-интерфейс для интеграции с внешними системами.

Таким образом, разработанная автоматизированная система управления транспортной логистикой позволит преодолеть ограничения существующих решений, повысить эффективность логистических процессов, сократить издержки и улучшить качество обслуживания клиентов. Внедрение такой системы обеспечит предприятию значительные операционные и финансовые выгоды, а также повысит его конкурентоспособность.

В заключении хочу отметить, что разработка автоматизированной системы управления транспортной логистикой является важным шагом в современном бизнесе, позволяя оптимизировать логистические процессы, улучшить обслуживание клиентов и повысить конкурентоспособность предприятия.

Список используемой литературы и используемых источников

1. Введение в Flutter [Электронный ресурс] // Метанит. - URL: https://metanit.com/dart/flutter/1.1.php (дата обращения: 12.03.2024).

2. Веселова А.О. Логистика: учеб. пособие для студ. экон. направлений подготовки (бакалавриат) очной и заочной форм обучения / А.О. Веселова, Е.А. Антинескул. - Пермь : Перм. гос. нац. исслед. ун-т., 2014. - 154 с.

3. Гвоздева Т.В. Проектирование информационных систем. Стандартизация: учебное пособие для вузов / Т.В. Гвоздева, Б.А. Баллод. - 2-е изд., стер. - СПб : Лань, 2021. - 252 с.

4. Заботина Н.Н. Проектирование информационных систем: учебное пособие / Н.Н. Заботина. - Москва : ИНФРА-М, 2022. - 331 с.

5. Иванов С. CI/CD - что это: основы непрерывной интеграции и доставки в DevOps [Электронный ресурс] / С. Иванов // Блог Скиллфактори. - 2024. - URL: https://blog.skillfactory.ru/glossary/ci-cd/ (дата обращения: 04.04.2024).

6. Как кратно увеличить продажи с помощью логистики [Электронный ресурс] // Ростов-Логист. - URL: https://rostov-logist.ru/o-logistike-obuchenii-i-konsaltinge/kak-kratno-uvelichit-prodazhi-s-pomoschyu-logistiki/
(дата обращения: 13.04.2024).

7. Ковалев В.В. Секреты успешных предприятий: бизнес процессы и организационная структура / В.В. Ковалев, С.В. Ковалев - М. : БИТЕК, 2012. – 498 с.

8. Методы проектирования организационной структуры и бизнес-процессов предприятия при внедрении ERP-систем (часть 1) [Электронный ресурс] // Корпоративные информационные системы. - URL: https://corpinfosys.ru/archive/2018/issue-4/134-2018-4-processes
(дата обращения: 15.03.2024).

9. Разработка нативных и кроссплатформенных приложений - что выбрать? [Электронный ресурс] // Tproger. - URL:

https://serptop.ru/blog/razrabotka-nativnykh-i-krossplatformennykh-prilozhenii-chto-vybrat/ (дата обращения: 15.03.2024).

10. Реляционные и нереляционные базы данных: какие выбрать? [Электронный ресурс] // Хабр. - URL: https://habr.com/ru/companies/sberbank/articles/672022/ (дата обращения: 14.03.2024).

11. Степанов Д.Ю. Анализ, проектирование и разработка корпоративных информационных систем: теория и практика [Электронный ресурс] / Д.Ю. Степанов // Российский технологический журнал. - 2015. - Т. 8, № 3. - С. 227-238. - URL: https://stepanovd.com/science/31-article-2015-2-erpthpr (дата обращения: 20.03.2024).

12. Степанов Д.Ю. Анализ, проектирование и разработка корпоративных информационных систем: уровень процессов [Электронный ресурс] / Д.Ю. Степанов // МГТУ МИРЭА. - М., 2017. - URL: https://stepanovd.com/training/12-erp/51-erp-7-processlevel
(дата обращения: 20.03.2024).

13. Сулейменов Т.Б. Транспортная логистика (I часть) / Т.Б. Сулейменов, М. И. Арпабеков. - Астана, 2012. - 211 с.

14. Фреймворки для веб-разработки Python [Электронный ресурс] // Бобдей. - URL: https://bobday.ru/freymvorki-dlya-veb-razrabotki-python (дата обращения: 25.02.2024).

15. Шумаев В.А. Основы логистики : учеб. пособие / В.А. Шумаев. - Москва : Юридический институт МИИТ, 2016. - 314 с.

16. Яценков А.В. Разработка веб-приложений: учебное пособие / А.В. Яценков. - СПб : НИУ ИТМО, 2014. - 88 с.

17. Doctrine ORM Documentation [Электронный ресурс] // Doctrine Project. - URL: https://www.doctrine-project.org/projects/doctrine-orm/en/current/index.html (дата обращения: 27.03.2024).

18. DoctrineFixturesBundle Documentation [Электронный ресурс] // Symfony, 2024. URL:

https://symfony.com/bundles/DoctrineFixturesBundle/current/index.html (дата обращения: 5.04.2024).

19. Lerman J. Code First: A New Approach to Database Development // MSDN Magazine. - May 2011.

20. Migrations in Symfony Applications [Электронный ресурс] // Symfony Documentation. - URL: https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html (дата обращения: 27.03.2024).

21. React Native [Электронный ресурс] // Блог Скиллфактори. - URL: https://blog.skillfactory.ru/glossary/react-native/ (дата обращения: 12.03.2024).

22. Scheduler [Электронный ресурс] // Symfony. - URL: https://symfony.com/doc/current/scheduler.html (дата обращения: 5.04.2024).

23. Security Bundle Component [Электронный ресурс] // Symfony. - URL: https://symfony.com/components/Security%20Bundle (дата обращения: 5.04.2024).

24. Spring Framework: что это и зачем нужен - принципы и структура фреймворка, как начать пользоваться Spring для начинающих [Электронный ресурс] // Яндекс Практикум. - URL: https://practicum.yandex.ru/blog/framework-spring-java/ (дата обращения: 25.02.2024).

25. Symfony Documentation [Электронный ресурс] // Symfony. - URL: https://symfony.com/doc/current/index.html (дата обращения: 25.02.2024).

26. The API Platform Core Library [Электронный ресурс] // API Platform. - URL: https://api-platform.com/docs/core/ (дата обращения: 5.04.2024).

27. Xamarin: кроссплатформенная разработка мобильных приложений [Электронный ресурс] // Microsoft Docs. - URL: https://docs.microsoft.com/ru-ru/xamarin/ (дата обращения: 14.03.2024).

Приложение А

Код генерации маршрутного листа

```
<?php
declare(strict_types=1);

namespace App\Service;

use App\Entity\Driver;
use App\Entity\Pack;
use App\Entity\RouteSheet;
use App\Entity\RouteSheetDestination;
use App\Entity\RouteSheetStatus;
use App\Entity\RouteTemplate;
use App\Entity\RouteTemplateDestination;
use App\Entity\Vehicle;
use App\Exception\NotFoundAvailableDriver;
use App\Exception\NotFoundAvailablePacks;
use App\Exception\NotFoundAvailableVehicle;
use App\Repository\DriverRepository;
use App\Repository\PackRepository;
use App\Repository\RouteSheetRepository;
use App\Repository\RouteSheetStatusRepository;
use App\Repository\VehicleRepository;
use App\Service\GenerateRouteSheet\RouteSheetStepCollection;
use Doctrine\ORM\EntityManagerInterface;
use Doctrine\ORM\EntityNotFoundException;
use Symfony\Component\Validator\ConstraintViolationInterface;
use Symfony\Component\Validator\Exception\ValidatorException;
use Symfony\Component\Validator\Validator\ValidatorInterface;
use Symfony\Contracts\Translation\TranslatorInterface;

/**
 * Сервис генерации маршрутного листа на основании шаблона и плановой даты
 */
class GenerateRouteSheet implements GenerateRouteSheetInterface
{
    /**
     * Конструктор класса
     *
     * @param RouteSheetRepository $routeSheetRepository
     * @param PackRepository $packRepository
     * @param VehicleRepository $vehicleRepository
     * @param DriverRepository $driverRepository
     * @param RouteSheetStatusRepository $routeSheetStatusRepository
     * @param ValidatorInterface $validator
     * @param CalculateRouteSheetStepsInterface $calculateRouteSheetSteps
     * @param EntityManagerInterface $entityManager
     * @param TranslatorInterface $translator
     */
    public function __construct(
```

```
        private RouteSheetRepository $routeSheetRepository,
        private PackRepository $packRepository,
        private VehicleRepository $vehicleRepository,
        private DriverRepository $driverRepository,
        private RouteSheetStatusRepository $routeSheetStatusRepository,
        private ValidatorInterface $validator,
        private CalculateRouteSheetStepsInterface $calculateRouteSheetSteps,
        private EntityManagerInterface $entityManager,
        private TranslatorInterface $translator
    ) {
    }

    /**
     * Главный метод выполнения операции
     *
     * @param RouteTemplate $routeTemplate
     * @param \DateTimeInterface|null $planDate
     *
     * @return RouteSheet
     */
    public function execute(RouteTemplate $routeTemplate, ?\DateTimeInterface $planDate =
null): RouteSheet
    {
        // Если установлена плановая дата, то проверяем ее на минимально возможную
        // Минимально возможная дата - завтра
        if ($planDate) {
            $minDate = (new \DateTime('tomorrow'))->setTime(0, 0);
            if ($planDate < $minDate) {
                throw new ValidatorException(
                    $this->translator->trans("Min date for generating is '%date%'",
                        ['%date%' => $minDate->format('d.m.Y')]
                    )
                );
            }
        }
        // Если шаблон маршрутна не активен - выдает исключение
        if (!$routeTemplate->isActive()) {
            throw new ValidatorException('Route template is not active');
        }

        // Получаем период на основании плановой даты
        $period = $this->getDatePeriod($planDate);
        // Пытаемся наути уже существующий маршрутный лист
        $routeSheet = $this->routeSheetRepository->findOneByRouteTemplate(
            $routeTemplate,
            $period
        );

        // Если маршрутный лист найден и его статус не в числе допустимых,
        // то прерываем генерацию по выбранным параметрам марщрутного листа
        if ($routeSheet && !in_array($routeSheet->getStatus()->getCode(), $this-
>getWrongStatuses())) {
```

```
            return $routeSheet;
        }

        // Создаем объект маршрутного листа, если его нет и назначаем плановую дату
        $routeSheet ??= $this->makeRouteSheet($routeTemplate);
        $routeSheet->setPlanDate($period->getStartDate());
        // Обнуляем комментарии к листу
        $routeSheet->setNote(null);

        try {
            // Если прикрепленное Транспортное средство неактивно - устанавливаем из
шаблона
            if ($routeTemplate->getVehicle()?->isActive()) {
                $routeSheet->setVehicle($routeTemplate->getVehicle());
            }

            // Получаем список упаковок, которые наобходимо доставить
            // в нужный период по адресам, прикрепленным к маршрутному шаблону
            $packs = $this->getNewPacks($routeTemplate, $period);

            // Добавляем список упаковок к маршрутному листу
            $routeSheet->setPacks($packs);

            // Считаем шаги маршрутного листа с помощью отдельного сервисного класса
            $steps = $this->calculateRouteSheetSteps->execute($routeSheet);
            if ($steps->count() == 0) {
                // Если не сосчитаны шаги, выдаем исключение, что маршрутный лист не
требуется к созданию
                throw new NotFoundAvailablePacks('No routes needed');
            }

            // Если не установлено транспортное средство, то находим доступное и подходящее
            if (!$routeSheet->getVehicle()) {
                $this->setAvailableVehicle($routeSheet, $steps, $period);
            }

            // Проверяем и подготавливаем список упаковок, которые поместятся с учетом
выбранного транспорта
            $packs = $this->preparePacksForVehicle($routeSheet, $packs);

            // Добавляем список пересчитанных упаковок к маршрутному листу с учетом
подобранного транспорта
            $routeSheet->setPacks($packs);

            // Устанавливаем доступного водителя
            $this->setAvailableDriver($routeSheet, $period);
            // Проверка данных маршрутного листа
            $this->validate($routeSheet);

            // Устанавливаем необходимый статус
            $routeSheet->setStatus($this->getDefaultStatus());
        } catch (EntityNotFoundException|ValidatorException $exception) {
```

```
            $routeSheet->setStatus($this->getErrorStatus());
            $routeSheet->addNote($exception->getMessage());
        }

        // Создаем или обновляем маршрутный лист через класс репозитория
        $this->routeSheetRepository->add($routeSheet);

        return $routeSheet;
    }

    /**
     * Создать сущность маршрутного листа на основе шаблона
     *
     * @param RouteTemplate $routeTemplate
     *
     * @return RouteSheet
     */
    private function makeRouteSheet(RouteTemplate $routeTemplate): RouteSheet
    {
        $entity = new RouteSheet();
        $entity->setRouteTemplate($routeTemplate);

        foreach ($routeTemplate->getDestinations() as $templateDestination) {
            $routeSheetDestination = new RouteSheetDestination();
            $routeSheetDestination->setDestination($templateDestination->getDestination());
            $routeSheetDestination->setSort($templateDestination->getSort());
            $entity->addDestination($routeSheetDestination);
        }

        return $entity;
    }

    /**
     * Получить упаковки для доставки на запланированную дату
     *
     * @param RouteTemplate $routeTemplate
     * @param \DatePeriod $period
     *
     * @return Pack[]
     * @throws NotFoundAvailablePacks
     */
    private function getNewPacks(RouteTemplate $routeTemplate, \DatePeriod $period): array
    {
        $newPacks = $this->packRepository->findNewByDestinations(
            array_map(fn(RouteTemplateDestination $d) => $d->getDestination(),
                $routeTemplate->getDestinations()->getValues()),
            $period
        );

        if (empty($newPacks)) {
            throw new NotFoundAvailablePacks('Packs not found for route template');
        }
```

```
        return $newPacks;
    }

    /**
     * Получить доступного водителя на дату
     *
     * @param float $minVolume
     * @param \DatePeriod $period
     *
     * @return Vehicle
     */
    private function getAvailableVehicle(float $minVolume, \DatePeriod $period): Vehicle
    {
        $availableVehicle = $this->vehicleRepository->findAvailable(
            $minVolume,
            $period
        );

        if (empty($availableVehicle)) {
            throw new NotFoundAvailableVehicle('Not found available vehicle');
        }

        return $availableVehicle;
    }

    /**
     * Получить доступное транспортное средство на дату
     *
     * @param \DatePeriod $period
     *
     * @return Driver
     */
    private function getAvailableDriver(\DatePeriod $period): Driver
    {
        $availableDriver = $this->driverRepository->findAvailable($period);

        if (empty($availableDriver)) {
            throw new NotFoundAvailableDriver('Not found available driver');
        }

        return $availableDriver;
    }

    /**
     * Валидация маршрутного листа
     *
     * @param RouteSheet $routeSheet
     *
     * @return void
     */
    private function validate(RouteSheet $routeSheet): void
```

```
{
    $errors = $this->validator->validate($routeSheet);
    if ($errors->count() > 0) {
        throw new ValidatorException(
            implode(
                ', ',
                array_map(fn(ConstraintViolationInterface $error) => $error->getMessage(),
                    iterator_to_array($errors))
            )
        );
    }
}

/**
 * Установить транспортное средство маршрутного листа
 *
 * @param RouteSheet $routeSheet
 * @param RouteSheetStepCollection $steps
 * @param \DatePeriod $period
 *
 * @return void
 */
private function setAvailableVehicle(
    RouteSheet $routeSheet,
    RouteSheetStepCollection $steps,
    \DatePeriod $period
): void {
    $routeSheet->setVehicle(
        $this->getAvailableVehicle(
            $steps->getMaxVolume(),
            $period
        )
    );
}

/**
 * Установить водителя маршрутного листа
 *
 * @param RouteSheet $routeSheet
 * @param \DatePeriod $period
 *
 * @return void
 */
private function setAvailableDriver(RouteSheet $routeSheet, \DatePeriod $period): void
{
    $routeSheet->setDriver($this->getAvailableDriver($period));
}

/**
 * Получить статуса по умолчанию для создаваемого маршрутного листа
 *
 * @return RouteSheetStatus
```

```
     */
    private function getDefaultStatus(): RouteSheetStatus
    {
        return $this->routeSheetStatusRepository->getDefaultStatus();
    }

    /**
     * Получить ошибочный статус
     * @return RouteSheetStatus
     */
    private function getErrorStatus(): RouteSheetStatus
    {
        return $this->routeSheetStatusRepository-
>findOneByCode(RouteSheetStatus::CODE_ERROR);
    }

    /**
     * Получить запланированную дату
     *
     * @param \DateTimeImmutable|null $planDate
     *
     * @return \DatePeriod
     */
    private function getDatePeriod(\DateTimeImmutable|null $planDate): \DatePeriod
    {
        $planDate ??= (new \DateTimeImmutable('now'))->setTime(8, 0);
        $interval = new \DateInterval('PT12H');

        return new \DatePeriod(
            $planDate,
            $interval,
            $planDate->add($interval)
        );
    }

    /**
     * Статусы, в которых маршрутный лист можно создать заново
     *
     * @return array
     */
    private function getWrongStatuses(): array
    {
        return [
            RouteSheetStatus::CODE_NEW,
            RouteSheetStatus::CODE_ERROR
        ];
    }

    /**
     * Если в шаблоне маршрута задано транспортное средство,
     * то необходимо оставить упаковки, которые поместятся в общую вместимость
транспортного средства
```

```
     *
     * @param RouteSheet $routeSheet
     * @param array $packs
     *
     * @return array
     */
    private function preparePacksForVehicle(RouteSheet $routeSheet, array $packs): array
    {
        $routeSheet->calculateVolumeSum();

        while ($routeSheet->getVehicle()->getCapacity()
            && $routeSheet->getVehicle()->getCapacity() < $routeSheet->getVolumeSum()
        ) {
            usort($packs, function (Pack $a, Pack $b) {
                return $a->getVolume() > $b->getVolume();
            });

            /** @var Pack $packToRemove */
            $packToRemove = array_pop($packs);
            $packToRemove->setOldPlanDate(clone $packToRemove->getPlanDate());
            $newDate = clone $packToRemove->getPlanDate();
            $newDate = $newDate->modify('+ 1 day');
            $packToRemove->setPlanDate($newDate);
            $routeSheet->removePack($packToRemove);
            $routeSheet->calculateVolumeSum();
            $routeSheet->addNote(
                sprintf(
                    'Moved pack ID: %d to new date: %s',
                    $packToRemove->getId(),
                    $packToRemove->getPlanDate()->format('Y-m-d')
                )
            );

            $this->entityManager->persist($packToRemove);
            $this->entityManager->flush($packToRemove);
        }

        if (empty($packs)) {
            throw new NotFoundAvailablePacks('All packs were planned for another date');
        }

        return $packs;
    }
}
```

Приложение Б

Код API-аутентификатора

```
<?php

namespace App\Security;

use App\Repository\UserRepository;
use Symfony\Component\HttpFoundation\InputBag;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;
use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;
use Symfony\Component\Security\Core\Exception\AuthenticationException;
use
Symfony\Component\Security\Core\Exception\CustomUserMessageAuthenticationException;
use Symfony\Component\Security\Core\Exception\RuntimeException;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Security\Core\User\UserProviderInterface;
use Symfony\Component\Security\Http\Authenticator\AbstractLoginFormAuthenticator;
use Symfony\Component\Security\Http\Authenticator\Passport\Badge\UserBadge;
use
Symfony\Component\Security\Http\Authenticator\Passport\Credentials\PasswordCredentials;
use Symfony\Component\Security\Http\Authenticator\Passport\Passport;
use Symfony\Component\Security\Http\SecurityRequestAttributes;
use Symfony\Component\Security\Http\Util\TargetPathTrait;

use function Symfony\Component\Translation\t;

class ApiAuthAuthenticator extends AbstractLoginFormAuthenticator
{
    use TargetPathTrait;

    /**
     * Методы, которые содержат данные в теле запроса
     *
     * @var array<string>
     */
    private static array $bodyMethods = ['POST', 'PUT', 'PATCH', 'DELETE'];

    /**
     * Название роута для авторизации
     */
    public const LOGIN_ROUTE = 'app_login';

    private UrlGeneratorInterface $urlGenerator;
    private UserRepository $userRepository;

    /**
```

```
     * Конструктор класса
     *
     * @param UrlGeneratorInterface $urlGenerator
     * @param UserRepository $userRepository
     */
    public function __construct(
        UrlGeneratorInterface $urlGenerator,
        UserRepository $userRepository
    ) {
        $this->urlGenerator = $urlGenerator;
        $this->userRepository = $userRepository;
    }

    /**
     * Метод позволяет проверить доступен ли данный аутентификатор для запроса
     *
     * @param Request $request
     *
     * @return bool
     */
    public function supports(Request $request): bool
    {
        // do your work when we're POSTing to the login page
        return (bool)preg_match('#^/api/(?!doc)#', $request->getPathInfo());
    }

    /**
     * Получение данных для авторизации из запроса
     *
     * @param Request $request
     *
     * @return array
     */
    public function getCredentials(Request $request)
    {
        return [
            'username' => $request->headers->get('php-auth-user'),
            'password' => $request->headers->get('php-auth-pw'),
        ];
    }

    /**
     * Получение объекта пользователя по данным авторизации
     *
     * @param $credentials
     * @param UserProviderInterface|null $userProvider
     *
     * @return \App\Entity\User|null
     */
    public function getUser($credentials, UserProviderInterface $userProvider = null)
    {
        return $this->userRepository->findOneBy(['username' => $credentials['username']]);
```

```
    }

    /**
     * Метод для входной точки
     *
     * @param Request $request
     * @param AuthenticationException|null $authException
     *
     * @return Response
     */
    public function start(Request $request, AuthenticationException $authException = null):
Response
    {
        $data = [
            'message' => Response::$statusTexts[Response::HTTP_UNAUTHORIZED]
        ];

        return new JsonResponse($data, Response::HTTP_UNAUTHORIZED);
    }

    /**
     * Метод, выполняющийся для проверки авторизации
     * @param Request $request
     *
     * @return Passport
     */
    public function authenticate(Request $request): Passport
    {
        $this->prepareRequestData($request);
        $credentials = $this->getCredentials($request);
        if (empty($credentials['username'])) {
            throw new CustomUserMessageAuthenticationException(t('Invalid credentials.'));
        }
        $request->getSession()->set(
SecurityRequestAttributes::LAST_USERNAME,
$credentials['username']
);
        return new Passport(
            new UserBadge($credentials['username']),
            new PasswordCredentials($credentials['password'])
        );
    }
    /**
     * Метод, выполняющийся при неуспешной автризации
     *
     * @param Request $request
     * @param AuthenticationException $exception
     *
     * @return Response
     */
    public function onAuthenticationFailure(Request $request, AuthenticationException
$exception): Response
```

```
    {
        $data = [
            'message' => strtr(t($exception->getMessageKey(), [], 'messages'), $exception-
>getMessageData())
        ];

        return new JsonResponse($data, Response::HTTP_UNAUTHORIZED);
    }
    /**
     * Метод, выполняющийся при успешной автризации
     *
     * @param Request $request
     * @param TokenInterface $token
     * @param string $firewallName
     *
     * @return Response|null
     */
    public function onAuthenticationSuccess(Request $request, TokenInterface $token, string
$firewallName): ?Response
    {
        return null;
    }
    /**
     * Получение ссылки для авторизации
     *
     * @param Request $request
     *
     * @return string
     */
    protected function getLoginUrl(Request $request): string
    {
        return $this->urlGenerator->generate(self::LOGIN_ROUTE);
    }
}
```

Приложение В

Код CRUD-контроллера для маршрутных листов

```
<?php

namespace App\Controller\Admin;

use App\Entity\RouteSheet;
use App\Entity\User;
use App\Form\Type\MapType;
use App\Scheduler\Task\GenerateRouteSheets;
use DateTimeImmutable;
use Doctrine\ORM\QueryBuilder;
use EasyCorp\Bundle\EasyAdminBundle\Collection\FieldCollection;
use EasyCorp\Bundle\EasyAdminBundle\Collection\FilterCollection;
use EasyCorp\Bundle\EasyAdminBundle\Config\Action;
use EasyCorp\Bundle\EasyAdminBundle\Config\Actions;
use EasyCorp\Bundle\EasyAdminBundle\Config\Assets;
use EasyCorp\Bundle\EasyAdminBundle\Config\Crud;
use EasyCorp\Bundle\EasyAdminBundle\Config\Filters;
use EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractCrudController;
use EasyCorp\Bundle\EasyAdminBundle\Dto\EntityDto;
use EasyCorp\Bundle\EasyAdminBundle\Dto\SearchDto;
use EasyCorp\Bundle\EasyAdminBundle\Field\AssociationField;
use EasyCorp\Bundle\EasyAdminBundle\Field\CollectionField;
use EasyCorp\Bundle\EasyAdminBundle\Field\DateField;
use EasyCorp\Bundle\EasyAdminBundle\Field\DateTimeField;
use EasyCorp\Bundle\EasyAdminBundle\Field\FormField;
use EasyCorp\Bundle\EasyAdminBundle\Field\IdField;
use EasyCorp\Bundle\EasyAdminBundle\Field\NumberField;
use EasyCorp\Bundle\EasyAdminBundle\Field\TextareaField;
use EasyCorp\Bundle\EasyAdminBundle\Field\TextEditorField;
use EasyCorp\Bundle\EasyAdminBundle\Orm\EntityRepository;
use Exception;
use Psr\Container\ContainerExceptionInterface;
use Psr\Container\NotFoundExceptionInterface;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\Validator\Exception\ValidatorException;

use function Symfony\Component\Translation\t;

/**
 * Класс управления сущностью RouteSheet
 */
class RouteSheetCrudController extends AbstractCrudController
{
    /**
     * Конструктор класса
```

```
     *
     * @param GenerateRouteSheets $generateRouteSheets
     */
    public function __construct(
        private readonly GenerateRouteSheets $generateRouteSheets
    ) {
    }

    /**
     * Указывается сущность, которой управляет контроллер
     *
     * @return string
     */
    public static function getEntityFqcn(): string
    {
        return RouteSheet::class;
    }

    /**
     * Конфигурация подключения ассетов (стили, скрипты)
     *
     * @param Assets $assets
     *
     * @return Assets
     */
    public function configureAssets(Assets $assets): Assets
    {
        $assets->addJsFile('assets/js/confirm-modal.js');

        return parent::configureAssets($assets);
    }

    /**
     * Конфигурация контроллера для добавления дополнительных действий
     *
     * @param Actions $actions
     *
     * @return Actions
     */
    public function configureActions(Actions $actions): Actions
    {
        $generatePdfAction = Action::new('generatePdf', 'Generate PDF', 'fa fa-envelope')
            ->linkToRoute('route_sheet_generate_pdf', function (RouteSheet $entity): array {
                return [
                    'id' => $entity->getId()
                ];
            });

        if ($this->isGranted('ROLE_SUPER_ADMIN')) {
            $generateListsAction = Action::new('generateLists', 'Generate lists for date', 'fa fa-file')
                ->displayAsLink()
                ->linkToCrudAction('generateLists')
```

```
            ->addCssClass('btn confirm-action')
            ->createAsGlobalAction()
            ->setHtmlAttributes(
                [
                    'data-bs-toggle' => 'modal',
                    'data-bs-target' => '#modal-confirm',
                ]
            );

        $actions->add(Crud::PAGE_INDEX, $generateListsAction);
    }

    return $actions
        ->add(Crud::PAGE_EDIT, $generatePdfAction)
        ->setPermission(Action::DELETE, 'ROLE_MANAGER')
        ->setPermission(Action::NEW, 'ROLE_MANAGER');
}

/**
 * Конфигурация crud-модели
 *
 * @param Crud $crud
 *
 * @return Crud
 */
public function configureCrud(Crud $crud): Crud
{
    return parent::configureCrud($crud)
        ->setPageTitle(Crud::PAGE_INDEX, 'Route sheets')
        ->setDefaultSort(['planDate' => 'ASC', 'id' => 'ASC'])
        ->setEntityLabelInSingular('Route sheet')
        ->setPaginatorPageSize(7)
        ->setEntityPermission('ROLE_ROUTES');
}

/**
 * Конфигурация фильтров
 *
 * @param Filters $filters
 *
 * @return Filters
 */
public function configureFilters(Filters $filters): Filters
{
    return $filters
        ->add('id')
        ->add('status')
        ->add('routeTemplate')
        ->add('planDate')
        ->add('startDateTime')
        ->add('finishDateTime')
        ->add('vehicle')
```

```
            ->add('driver')
            ->add('volumeSum')
            ->add('note');
    }

    /**
     * Конфигурация полей, которыми можно управлять при просмотре, редактировании
сущности
     *
     * @param string $pageName
     *
     * @return iterable
     */
    public function configureFields(string $pageName): iterable
    {
        $editEnabled = $this->isGranted('ROLE_MANAGER');

        yield FormField::addTab('General')
            ->setHelp('General information');

        yield FormField::addColumn(8);
        yield IdField::new('id')->hideWhenCreating()->setDisabled();
        yield AssociationField::new('routeTemplate')
            ->onlyWhenCreating()
            ->setDisabled(false);
        yield AssociationField::new('routeTemplate')
            ->onlyWhenUpdating()
            ->setDisabled();
        yield DateField::new('planDate')->setDisabled(!$editEnabled);
        yield AssociationField::new('vehicle')
            ->setRequired(true)
            ->setDisabled(!$editEnabled);
        yield AssociationField::new('driver')
            ->setRequired(true)
            ->setDisabled(!$editEnabled);
        yield TextEditorField::new('note')->setDisabled(!$editEnabled);

        yield FormField::addColumn(4);
        yield AssociationField::new('status')
            ->setRequired(true);
        yield DateTimeField::new('startDateTime');
        yield DateTimeField::new('finishDateTime');

        yield FormField::addTab('Destination info')
            ->setHelp('Information about all route destinations');

        yield FormField::addColumn(12);
        yield CollectionField::new('destinations', 'Destinations')
            ->setRequired(true)
            ->setEntryIsComplex()
            ->useEntryCrudForm(
                RouteSheetDestinationCrudController::class
```

```
        )
        ->allowAdd($editEnabled)
        ->allowDelete($editEnabled);

    yield FormField::addTab('All packs info')
        ->setHelp('Packs info from all destinations');

    yield NumberField::new('volumeSum')->setDisabled();
    yield CollectionField::new('packs', 'All packs')
        ->setDisabled()
        ->setEntryIsComplex()
        ->useEntryCrudForm(
            PackCrudController::class
        );

    yield TextareaField::new('destinations')
        ->setLabel('Points on the map')
        ->setFormType(MapType::class);
}

/**
 * Создает и конфигурирует объект QueryBuilder для выполнения запроса поиска
 *
 * @param SearchDto $searchDto
 * @param EntityDto $entityDto
 * @param FieldCollection $fields
 * @param FilterCollection $filters
 *
 * @return QueryBuilder
 * @throws ContainerExceptionInterface
 * @throws NotFoundExceptionInterface
 */
public function createIndexQueryBuilder(
    SearchDto $searchDto,
    EntityDto $entityDto,
    FieldCollection $fields,
    FilterCollection $filters
): QueryBuilder {
    /** @var User $user */
    $user = $this->getUser();

    /** @var QueryBuilder $qb */
    $qb = $this->container->get(EntityRepository::class)
        ->createQueryBuilder($searchDto, $entityDto, $fields, $filters);

    if ($user?->getDriver() && in_array('ROLE_DRIVER', $user?->getRoles())) {
        $qb->andWhere($qb->expr()->in('entity.driver', $user->getDriver()->getId()));
    }

    return $qb;
}
```

```
    /**
     * Метод генерации списка листов
     *
     * @param Request $request
     *
     * @return RedirectResponse
     */
    public function generateLists(Request $request)
    {
        try {
            $value = $request->get('date');
            $date = new DateTimeImmutable($value);
            $this->generateRouteSheets->generateRouteSheets($date);

            $this->addFlash(
                'success',
                t(
                    "Route lists for date '%date%' are generated successfully",
                    ['%date%' => $date->format('d.m.Y')]
                )
            );
        } catch (ValidatorException $exception) {
            $this->addFlash('error', t($exception->getMessage()));
        } catch (Exception $exception) {
            $this->addFlash('error', t($exception->getMessage()));
        }

        return $this->redirectToRoute('admin');
    }
}
```