Министерство образования Республики Беларусь

ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет «Повышения квалификации и переподготовки кадров»

Кафедра ««Информационные системы и автоматизация производства»»

Допущена к защите

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. Е. Казаков

« » 2020 г.

ДИПЛОМНАЯ РАБОТА

**тема: разработка информационной системы управления аварийно-профилактической группой**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | |  | |
| Специальность | | | 1-40 01 73 «Программное обеспечение информационных систем» | |
| Группа | | | ИС-17 | |
| Исполнитель |  |  |  | Д.Н.Прядкин |
|  |  |  |  | И.О. Фамилия |
| Руководитель |  |  |  | к.т.н., доц. А.С.Дягилев |
|  |  |  |  | учёная степень, должность, И.О. Фамилия |

**Витебск 20****20**

**УО «Витебский государственный технологический университет»**

**Факультет повышения квалификации и переподготовки кадров**

УТВЕРЖДАЮ

Зав. кафедрой

В. Е. Казаков

(ФИО, подпись)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

**ЗАДАНИЕ**

**по подготовке дипломной работы**

1.Тема работы «Разработка информационной системы управления аварийно-профилактической группой»

Утверждена приказом по институту от №

2. Сроки сдачи слушателем выполненной работы

3. Перечень вопросов, которые будут разработаны в дипломной работе или краткое содержание дипломной работы:

* + 1. Понятие об информационной системе управления аварийно-профилактической группой
    2. Понятие информационной системы управления аварийно-профилактической группой и ее применение
    3. Предметная область и требования к информационной системе управления аварийно-профилактической группой
    4. Средства реализации и разработки информационной системы управления аварийно-профилактической группой
    5. Разработка информационной системы управления аварийно-профилактической группой
    6. Проектирование структуры базы данных
    7. Разработка классов
    8. Интерфейс информационной системы управления аварийно-профилактической группой
    9. Тестирование информационной системы управления аварийно-профилактической группой

4. Перечень практических материалов, которые будут разработаны в дипломной работе:

* + 1. информационная система управления аварийно-профилактической группой.

Задание принял для выполнения Прядкин Д.Н.

(фамилия) (дата)

Подпись слушателя

**Реферат**

Дипломная работа 55 c., 38 рис., 11 источников, 5 прил.

**Ключевые слова:** ИНФОРМАЦИОННАЯ СИСТЕМА, СИСТЕМА УПРАВЛЕНИЯ, PHP, БАЗА ДАННЫХ, КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ, UML-ПРОЕКТИРОВАНИЕ, ЛОГИЧЕСКАЯ МОДЕЛЬ, ФИЗИЧЕСКАЯ МОДЕЛЬ.

**Объект исследования** – информационная система управления аварийно-профилактической группой.

**Предмет исследования** – применение современных информационных технологий для создания информационной системы управления аварийно-профилактической группой.

**Цель работы** – проектирование и разработка информационной системы управления аварийно-профилактической группой.

**Методы исследования и технология разработки**: описательно-аналитический, экспериментальный, объектно-ориентированное программирование, построение баз данных. В качестве системы управления базами данных выбрана СУБД MySQL 8.0. Средствами разработки являются PHP 7.4, HTML, CSS, UML. Инструментарий для разработки: IDE PhpStorm 2020.2, Apache 2.4, онлайн ресурс для построения UML-диаграмм Draw.io, Bootstrap 4, MySQL Workbench 8.0 CE.

Теоретическая и практическая значимость заключается в том, что результаты работы можно применять на практике, концепция данного исследования может быть использована в качестве методического пособия при разработке информационных систем.

**Перечень условных обозначений и терминов**

АПГ – аварийно-профилактическая группа.

ИС – информационная система.

ООП – объектно-ориентированное программирование.

СУБД – система управления базами данных.

CSS (англ. Cascading Style Sheets – каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

HTTP – HyperText Transport Protocol (протокол передачи гипертекста).

MVC (Model View Controller, «модель-представление-контроллер», «модель-вид-контроллер») – схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента.

PHP – рекурсивный акроним словосочетания Hypertext Preprocessor – это распространенный язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML.

SQL – Structured Query Language (язык структурированных запросов).

UI – User Interface (пользовательский интерфейс)

UML (англ. Unified Modeling Language – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения.

XHTML – Extensible HyperText Markup Language (расширяемый язык разметки гипертекста).

**Оглавление**

[Введение 7](#_Toc48887440)

[1 Постановка задачи 9](#_Toc48887441)

[1.1 Описание предметной области 9](#_Toc48887442)

[1.2 Определение требований к разрабатываемой программной системе 10](#_Toc48887443)

[1.3 Выбор средства реализации 13](#_Toc48887444)

[2 Разработка программного продукта 19](#_Toc48887445)

[2.1 Проектирование структуры базы данных 19](#_Toc48887446)

[2.2 Разработка классов 27](#_Toc48887447)

[2.3 Описание интерфейса программного продукта 33](#_Toc48887448)

[2.4 Тестирование информационной системы управления аварийно-профилактической группой 46](#_Toc48887449)

[Заключение 53](#_Toc48887450)

[Список использованных источников 55](#_Toc48887451)

[ПРИЛОЖЕНИЕ А 56](#_Toc48887452)

[ПРИЛОЖЕНИЕ Б 65](#_Toc48887453)

[ПРИЛОЖЕНИЕ В 66](#_Toc48887454)

[ПРИЛОЖЕНИЕ Г 67](#_Toc48887455)

[ПРИЛОЖЕНИЕ Д 68](#_Toc48887456)

# 

**Введение**

Современное общество подчиняется концепции современного этапа общественного развития, характеризующего переход к новой форме постиндустриального общества, где доминирующей ценностью, экономической и ресурсной, становится информация. Это формирует так называемое общество знания или информационное общество.

Суть информационного общества заключается в том, что человеческая цивилизация после аграрной и индустриальной стадии развития вступает в новую – информационную, где информация считается наиболее ценным ресурсом, а её доступность является наиболее важной в данной идеологии.

**Целью** нашей работы является проектирование и разработка информационной системы управления аварийно-профилактической группой.

Основные **задачи**:

* изучить основные характеристики систем управления работами, определить их задачи и функции;
* обозначить аудиторию и предметную область разрабатываемой системы управления аварийно-профилактической группой, функции, которые она будет выполнять; определить информационную составляющую;
* выбрать программные средства и разработать систему управления аварийно-профилактической группой.

Конечный **результат** работы будет представлять собой сайт управления производственными задачами и поездками, выполняемыми АПГ. Теоретическая и практическая значимость заключается в том, что результаты работы можно применить для планирования и учёта задач в любой сфере деятельности.

**Методы исследования и технология разработки**: описательно-аналитический, экспериментальный, объектно-ориентированное программирование, процедурное программирование, построение баз данных. В качестве системы управления базами данных выбрана СУБД MySQL 8.0. Средствами разработки являются PHP 7.4, ХHTML, CSS, UML. Инструментарий для разработки: IDE PhpStorm 2020.2, Apache 2.4, онлайн ресурс для построения UML-диаграмм Draw.io, Bootstrap 4, MySQL Workbench 8.0 CE.

**1 Постановка задачи**

**1.1 Описание предметной области**

Для осуществления эфирной трансляции программ радиовещания и цифрового телевидения, РУП «Белтелеком» содержит, обслуживает и постоянно модернизирует сеть автоматизированных радиотелевизионных передающих станций. Такая сеть состоящая из радиотелевизионных передающих устройств подобна сети мобильного оператора предоставляющего абонентам услуги мобильный связи стандартов (2G/3G/4G). При этом имеются и существенные отличия: большие размеры зон покрытия сигналами, односторонний характер связи, наличие разнотипных радиопередающих устройств, а так же оборудования сети передачи и выделения телевизионных транспортных потоков и сигналов радиовещания.

Для осуществления технического обслуживания оборудования, выполнения планово-профилактических, ремонтно-настроечных и аварийно-восстановительных работ на объектах телерадиовещания сформированы аварийно-профилактические группы (АПГ). Они укомплектованы водителями, антенщиками-мачтовиками, высококвалифицированным инженерно-техническим персоналом, а так же оснащены необходимыми средствами измерений, приборами, инструментами и автотранспортом. В процессе осуществления производственной деятельности работники АПГ выполняют множество задач. Эти задачи необходимо своевременно планировать, контролировать в процессе выполнения и учитывать в журналах работ и отчётах. В данное время эти данные регистрируются на бумажных носителях информации и в электронных таблицах.

В связи с тем, что обслуживаемые станции разнесены между собой на довольно большие расстояния, то для доставки специалистов и оборудования используется служебный автотранспорт. Выезды автомашин должны осуществляется на основании предварительно поданных заявок.

Разрабатываемая информационная система управления АПГ предназначена для планирования, контроля и учёта производственных заданий и выездов автотранспорта.

Данная дипломная работа является актуальной в связи с необходимостью внедрения электронного документирования выполняемых задач и минимизации избыточного дублирования данных в различных учётных документах.

**1.2 Определение требований к разрабатываемой программной системе**

Информационная система должна быть независимой от операционных систем, установленных на компьютерах пользователей, многопользовательской (предоставлять возможность одновременного доступа к необходимой информации для пользователей с различными правами).

Для работы с разрабатываемой программной системой были определены три категории (группы) пользователей: администратор, руководитель и обычные пользователи.

Администратор работает со следующими объектами и функциями информационной системы: создание, изменение, удаление пользователей системы, изменять их права доступа к разделам информационной системы; работать со справочниками (добавлять, изменять, удалять данные в них).

Руководитель работает (создаёт, изменяет, удаляет) с заданиями и поездками служебного автотранспорта.

Пользователь имеет возможность просматривать ведомость выездов автотранспорта и список незавершённых заданий (журнал работ).

Диаграммы прецедентов для разрабатываемой информационной системы приведена на рисунках 1.1 и 1.2.

Таким образом, можно сформулировать следующие требования с разрабатываемой программной системе:

* независимость от операционной системы;
* наличие системы авторизации и разграничения прав доступа к системе;
* гибкость и удобство использования, как для пользователей, так и для администратора;
* возможность добавлять, изменять, удалять (для администратора) данные в справочниках: «Локации» (места проведения работ), «Транспорт» (автопарк подразделения), «Персонал» (сотрудники организации);
* возможность добавлять, изменять, удалять (руководителя) и просматривать (для пользователя) заявок на выезд автотранспорта для выполнения производственных задач;
* возможность (для руководителя) изменять статус задачи в зависимости от этапов её реализации.

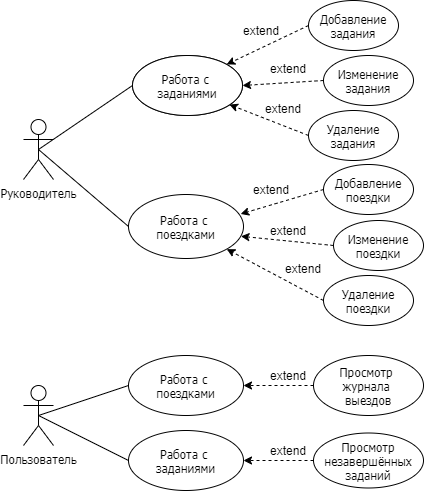
**

Рисунок 1.1 – Диаграмма прецедентов системы управления аварийно-профилактической группой для «Руководителя» и «Пользователя».

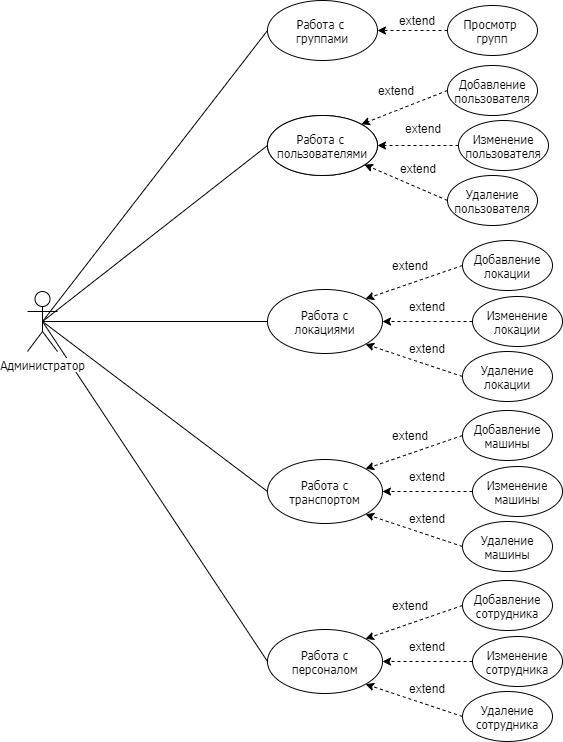


Рисунок 1.2 – Диаграмма прецедентов системы управления аварийно-профилактической группой для «Администратора».

**1.3 Выбор средства реализации**

Учитывая поставленные выше задачи, в качестве средств реализации и разработки системы управления аварийно-профилактической группой были выбраны PHP, ХHTML, CSS. В качестве системы управления базами данных выбрана СУБД MySQL, для разработки диаграмм прецедентов и классов использовался язык UML.

Язык программирования РНР 7 — серверный язык, при помощи которого можно создавать веб-сайты, причем как небольшие лендинги, состоящие из одной страницы, так и гигантские системы, использующие сотни и тысячи серверов. Элек­тронная энциклопедия Wikipedia, социальные сети Facebook, ’’ВКонтакте”, электронная площадка объявлений Avito созданы с использованием РНР.

Основными преимуществами языка программирования PHP являются:

* Ориентация на веб-разработку — РНР создавался, развивался и поддерживается как язык для создания веб-сайтов. Многие конструкции и решения в нем созданы для удобства работы в веб-среде.
* Кроссплатформенность — РНР перенесен на все основные операционные системы: можно разрабатывать сайт в Windows, Mac OS X, а эксплуатировать на Linux-сервере. Сложности переноса будут минимальны и нивелироваться языком.
* Бесплатность — РНР является разработкой из мира свободного программного обеспечения, не потребуется платить ни за сам язык, ни за большинство сопутствующих программ (редакторы, веб-серверы, базы данных). Вдобавок большинство программных продуктов, с которыми придется иметь дело, будут иметь доступный для изучения и модификации исходный код. Вложения могут потребоваться при аренде доменного имени и сервера для публикации сайта в Интернете. Однако изучать РНР можно, не вкладывая ни копейки.
* Низкий порог входа — изучить РНР и начать создавать на нем готовые приложения много проще, чем с использованием конкурирующих технологий (.NET, Python, Ruby, Go). Изучение PHP не закрывает для разработчика другие технологии, в Web сам язык — значительная, но меньшая часть используемых технологий. Знания, приемы работы, сопутствующие технологии (веб-серверы, базы данных, библиотеки, вспомогательные языки) пригодятся и в любой другой экосистеме, отличной от РНР. [1]

XHTML – это основанный на [XML](https://htmlweb.ru/xml/xml1.php) язык разметки гипертекста, максимально приближенный к текущим стандартам [HTML](https://htmlweb.ru/html/bhtml.php). XHTML отличается от HTML строгостью написания кода. Если HTML позволял писать практически любые конструкции и браузер их корректно распознавал, то теперь, с появлением XHTML, это стало невозможным. Последний требует строгого соблюдения всех правил, предъявляемых W3C. Строгие требования к оформлению XHTML- кода позволяют избежать многих ошибок ещё на стадии написания и отладки. XHTML-документ состоит из текста, представляющего собой содержание документа, и тегов, определяющих его структуру и внешний вид при отображении браузером, например, теги сообщают браузеру, какая часть текста является заголовком, где начинается новый абзац, что нужно подчеркнуть и где расположить изображение. XHTML-данные правильно отдавать клиенту именно в формате <application/xhtml+xml>, так как все преимущества, помимо кросс-браузерности (увеличение скорости анализа кода процессором XML, сообщение об ошибках самим браузером и пр.), могут достигаться только в случае, если пользовательский агент поддерживает XHTML и ему сообщается о том, что входящие данные - XHTML-код. Единственное, что надо помнить при отправке XHTML-кода: если браузер понимает XHTML, то только тогда информацию можно отправить как <application/xhtml+xml>; если же нет, то только как <text/html> [2].

CSS (англ. Cascading Style Sheets – каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL. CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля [3].

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle. MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов OpenServer, XAMPP, VertrigoServ. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц, благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц. Преимуществами СУБД MySQL можно считать следующие:

- Высокая скорость обработки данных;

- Поддержка SQL, что обеспечивает высокий уровень кроссплатформенности данных и кода, созданных с помощью MySQL. Благодаря этому можно спокойно перенести БД в любую другую современную СУБД, также поддерживающую язык структурированных запросов;

- Система привилегий, что позволяет наделять каждую учетную запись сервера правами на осуществление определенных действий с данными. Причем не только на уровне сервера, БД, но и на уровне отдельных таблиц.

- Хэширование паролей – обеспечивает высокий уровень «противовзломности». Именно поэтому в MySQL восстановить пароль root очень сложно.[4].

UML (англ. Unified Modeling Language – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода [5].

Для создания веб-сайта на локальном компьютере будет использоваться сборка на «домашней» (локальной) Windows-машине без необходимости выхода в Интернет. Сборка выглядит следующим образом:

* Open Server (Apache 2.4, MySQL 8.0.19, PHP 7.4.05, PHPMyAdmin 5.0.2),
* IDE PhpStorm 2020.2,
* Браузер Google Chrome или любой другой.

Open Server Panel — это портативная серверная платформа и программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий.

Программный комплекс имеет богатый набор серверного программного обеспечения, удобный, многофункциональный продуманный интерфейс, обладает мощными возможностями по администрированию и настройке компонентов. Платформа широко используется с целью разработки, отладки и тестирования веб-проектов, а так же для предоставления веб-сервисов в локальных сетях. [6]

Хотя изначально программные продукты, входящие в состав комплекса, не разрабатывались специально для работы друг с другом, такая связка стала весьма популярной среди пользователей Windows, в первую очередь из-за того, что они получали бесплатный комплекс программ с надежностью на уровне Linux серверов.

Apache HTTP Сервер –  веб-[сервер](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) с открытым исходным кодом, поддерживается и развивается компанией Apache Software Foundation. Самой главной задачей веб-сервера является одновременное обслуживание различных запросов от многих пользователей. он выполняет роль посредника между компьютером сервера и компьютером клиента. Веб-сервер обрабатывает файлы написанные на разных языках программирования, таких как PHP, Python, Java и другие. Он превращает их в статический файл HTML, которые отдаёт на обслуживание в браузер пользователя.

Для создания UML-диаграмм использовался инструмент Draw.io [2]. Назначение программы - создание схем и диаграмм. Приложение имеет различные панели инструментов для форм, соединителей, соединение объектов создается одним щелчком, изменение размерных стрелок является естественным и быстрым. Возможно выравнивание, установка размеров, группировка, расположение в слоях, использование умных форм для макетов UI, UML и больше. Изменение ключевых визуальных особенностей интуитивно в нескольких щелчках.

MySQL Workbench 8.0 CE – это бесплатный и профессиональный инструмент для разработчиков БД и пользователей MySQL. Он автоматизирует рутинные задачи по разработке и администрированию СУБД MySQL, а также открывает новые возможности для получения лучшего результата. В нем можно разрабатывать SQL скрипты, хранимые процедуры и функции, составлять и выполнять запросы, редактировать данные, осуществлять их экспорт и импорт, управлять пользователями, редактировать объекты БД, работать с проектами БД и многое другое.

Для работы над кодом использовалась IDE PhpStorm — это среда разработки, которая обеспечивает первоклассную поддержку PHP всех версий, мгновенно предотвращает ошибки, предоставляет точное автодополнение и безопасные рефакторинги, а также возможность редактирования кода на HTML, CSS и JavaScript. Основными преимуществами являются: умное автодополнение, подсветка синтаксиса, расширенные настройки форматирования, проверка ошибок на лету, сворачивание фрагментов кода, поддержку языковых комбинаций и много других возможностей для продуктивной разработки. Сотни встроенных инспекций проверяют код прямо в режиме редактирования, а также позволяют проанализировать весь проект на наличие ошибок и проблем в структуре кода. Для большинства инспекций доступны быстрые исправления, которые позволяют откорректировать код мгновенно. PhpStorm помогает быстро перемещаться по коду и экономить время при работе над большими проектами. [7]

**Выводы:**

Таким образом, система управления аварийно-профилактической группой должна, способствовать повышению эффективности и качества планирования и учёта производственных задач и выездов автотранспорта.

Таким образом, была выделена предметная область и основные требования к разрабатываемому программному продукту:

* система разграничения прав пользователей;
* гибкость и простота загрузки и получения информации в многомользовательском режиме;

В соответствии с поставленными задачами были выбраны также средства и инструментарий для проектирования и разработки веб-сайта, с учетом современных тенденций сайтостроения: PHP, XHTML, CSS, MySQL, а также паттерн проектирования MVC, набор инструментов Bootstrap, среда разработки IDE PhpStorm и другие необходимые программные продукты.

**2 Разработка программного продукта**

**2.1** **Проектирование структуры базы данных**

Для правильной работы приложения чрезвычайно важно качественное проектирование базы данных.

MySQL – это система управления реляционными базами данных. Важная особенность реляционных систем, их отличие от одноуровневых баз данных – возможность располагать данные в нескольких таблицах. Взаимосвязанные данные можно хранить в отдельных таблицах и объединять по ключу, общему для обеих таблиц. Ключ – это отношение (relation) между таблицами. Выбор первичного ключа (primary key) – наиболее важное решение, принимаемое при разработке новой базы данных. Самое главное – необходимо гарантировать уникальность выбранного ключа. Если есть вероятность того, что значение некоторого атрибута может совпасть у двух записей, то его нельзя использовать в качестве первичного ключа. Если таблица содержит ключевые поля из другой таблицы, то между ними образуется связь – взаимоотношением внешнего ключа (foreign key), например, «начальник-подчиненный» или «покупатель-покупка». Существуют следующие разновидности отношений (связей) между таблицами:

один-ко-многим. Самый распространенный вид связи. Одной записи таблицы соответствует любое количество (и 0, и 1, и даже еще больше) записей другой таблицы;

многие-ко-многим. Связь "многие-ко-многим" возникает между двумя таблицами, когда в каждой из них может присутствовать несколько ключей другой таблицы;

один-к-одному. При связи "один-к-одному" каждому элементу соответствует один и только один другой элемент.

Представление о взаимоотношениях данных и наиболее эффективном способе их организации называется нормализацией. Нормализация заключается в разделении данных на основе логических взаимоотношений с целью минимизировать дублирование данных.

Проектирование базы данных складывается из нескольких этапов и включает в себя концептуальное (инфологическое), логическое (даталогическое) и физическое проектирование.

Концептуальное (инфологическое) проектирование – это построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Чаще всего концептуальная модель базы данных включает в себя: описание информационных объектов или понятий предметной области и связей между ними и описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними [10].

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком.

Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

Сущность – любой различимый объект (объект, который мы можем отличить от другого), информацию о котором необходимо хранить в базе данных. Сущностями могут быть люди, места, самолеты, рейсы, вкус, цвет и т.д.

Атрибут – поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

Ключ – минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся.

Связь – ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся сотни или даже тысячи сущностей, то теоретически между ними может быть установлено более миллиона связей. Наличие такого множества связей и определяет сложность инфологических моделей.

Рассмотрим структурную модель базы данных, полученную в результате исследования предметной области и построения функциональной диаграммы.

Опишем предметную область проектируемой нами базы данных. Требуется разработать информационную систему управления аварийно-профилактической группой, для внесения, хранения и предоставления информации о всех выполняемых работах и выездах автотранспорта. Количество таблиц и связи между ними определяются задачами, которым должен отвечать разрабатываемый проект.

При анализе предметной области выявлены следующие сущности: «Пользователи», «Группы пользователей», «Персонал», «Автотранспорт», «Задача», «Локации», «Поездки».

Определим типы связей существующих между выделенными нами сущностями. Для этого снова анализируем требования к БД. Тип связи представляет собой название связи, ее координальность в этой связи. Результат анализа представлен в таблице 2.1.

Таблица 2.1 Типы связей между сущностями

|  |  |  |  |
| --- | --- | --- | --- |
| Тип  сущности | Тип связи | Тип сущности | Координальность |
| tasks | содержит | workers |  |
| places | имеет | tasks |  |
| taskstates | имеет | tasks |  |
| workers | имеет | people |  |
| workers | имеет | tasks |  |

Продолжение таблицы 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| Тип  сущности | Тип связи | Тип сущности | Координальность |
| trips | имеет | cars |  |
| trips | имеет | people |  |
| groups | содержит | users |  |

Выявленные атрибуты сущностей приведены в таблице 2.2.

Таблица 2.2 Атрибуты сущностей и связей

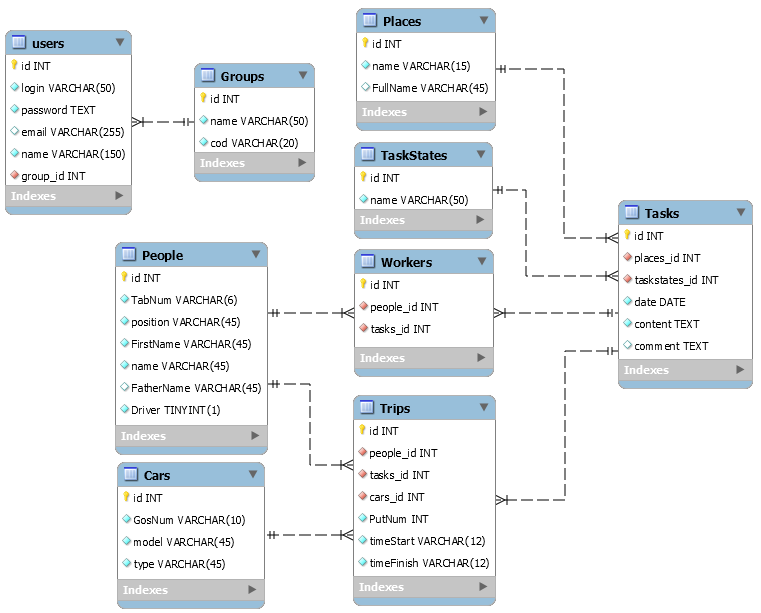
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип сущности (связи) | Атрибут | Описание | Домен | Обязательность |
| cars | id | код машины | целое | да |
| model | модель машины | символьный(45) | да |
| GosNum | номер машины | символьный(10) | да |
| type | тип машины | символьный(45) | да |
| places | id | код места | целое | да |
| name | название места | символьный(15) | да |
| FullName | полное название места | символьный(45) | нет |
| workers | id | код | целое | да |
| people\_id | код сотрудника | целое | да |
| tasks\_id | код задачи | целое | да |
| taskStates | id | код статуса | целое | да |
| name | статус задачи | символьный(50) | да |
| groups | id | код группы | целое | да |
| name | название | символьный(50) | да |
| cod | кодовое название | символьный(20) | да |
| users | id | код пользователя | целое | да |
| login | логин | символьный(50) | да |
| password | пароль | текстовый | да |
| email | почта | символьный(255) | нет |
| name | ФИО | символьный(150) | да |
| group\_id | код группы | целое | да |

Продолжение таблицы 2.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип сущности (связи) | Атрибут | Описание | Домен | Обязательность |
| people | id | код сотрудника | целое | да |
| TabNum | табельный номер | символьный(6) | да |
| position | должность | символьный(45) | да |
| FirstName | фамилия | символьный(45) | да |
| name | имя | символьный(45) | да |
| FatherName | отчество | символьный(45) |  |
| driver | право на вождение служебной машины | целое(1) | да |
| trips | id | код поездки | целое | да |
| people\_id | код водителя | целое | да |
| tasks\_id | код задачи | целое | да |
| cars\_id | код машины | целое | да |
| PutNum | номер путёвки | целое | да |
| timeStart | время выезда | символьный(12) | нет |
| timeFinish | время возвращения | символьный(12) | нет |
| tasks | id | код задачи | целое | да |
| places\_id | код места | целое | да |
| taskstates\_id | код статуса задачи | целое | да |
| date | дата выполнения задачи | дата | да |
| content | содержание задачи | текст | да |
| comment | описание задачи | текст | нет |

Логическое (даталогическое) проектирование – это создание схемы базы данных на основе конкретной модели данных. Для реляционной модели данных это набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

В результате проектирования получена даталогическая модель базы данных «wb\_db», представленная на рисунке 2.1.

Рисунок 2.1 - Даталогическая модель базы данных

Таким образом, веб-приложение содержит базу данных, состоящую из девяти таблиц.

Таблица «tasks» – производственные задачи (работы).

На рисунке 2.2 представлена структура таблицы «tasks».



Рисунок 2.2 – Структура таблицы «tasks»

Таблица «taskstates» – варианты статуса задачи. Каждая задача может иметь разный статус в зависимости от её состояния: запланировано, выполняется, приостановлено, выполнено.

На рисунке 2.3 представлена структура таблицы «taskstates».

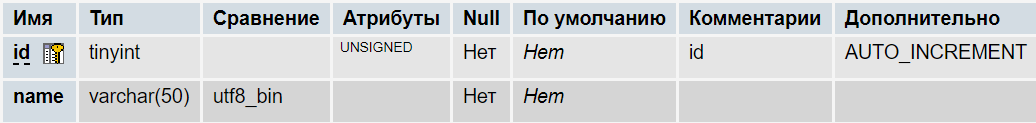


Рисунок 2.3 – Структура таблицы «taskstates»

Таблица «people» содержит данные сотрудников АПГ.

На рисунке 2.4 представлена структура таблицы «people».



Рисунок 2.4 – Структура таблицы «people»

Таблица «workers» – связывает между собой таблицы «people» и «tasks». В ней содержится информация об исполнителях работ. Одну работу(задание) могут выполнять от одного до нескольких человек.

На рисунке 2.5 представлена структура таблицы «workers».

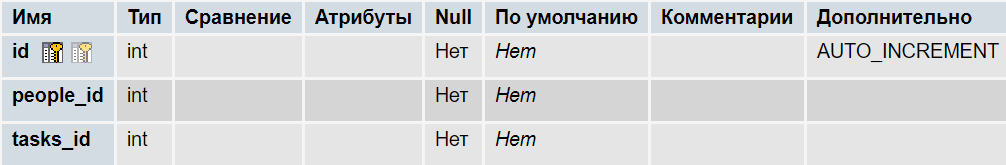


Рисунок 2.5 – Структура таблицы «workers»

Таблица «places» – содержит информацию о местах, где выполняются работы.

На рисунке 2.6 представлена структура таблицы «places».

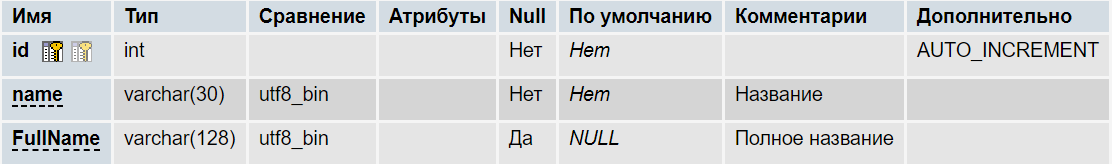


Рисунок 2.6 – Структура таблицы «places»

Таблица «cars» – содержит информацию об автотранспорте.

На рисунке 2.7 представлена структура таблицы «cars».



Рисунок 2.7 – Структура таблицы «places»

Таблица «trips» – содержит информацию о выездах автотранспорта.

На рисунке 2.8 представлена структура таблицы «trips».



Рисунок 2.8 – Структура таблицы «places»

Таблица «users» используется в процессе авторизации, содержит номер, логин и пароль.

На рисунке 2.9 представлена структура таблицы «users».



Рисунок 2.9 – Структура таблицы «users»

Таблица «groups» содержит роли пользователей.

На рисунке 2.10 представлена структура таблицы «groups».

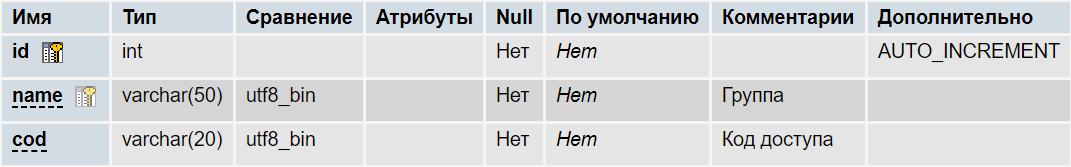


Рисунок 2.10 – Структура таблицы «groups»

Программный код создания базы данных представлен в Приложении А.

**2.2 Разработка классов**

При разработке нашего программного продукта мы используем объектно-ориентированное программирование на языке PHP. Объектно-ориентированные программы более просты и мобильны, их легче модифицировать и сопровождать.

Стратегию ООП лучше всего описать как смещение приоритетов в процессе программирования от функциональности приложения к структурам данных. Это позволяет программисту моделировать в создаваемых приложениях реальные объекты и ситуации.

Специфика ООП заметно повышает эффективность труда программистов и позволяет им создавать более мощные, масштабируемые и эффективные приложения. Объектно-ориентированное программирование основано на следующих принципах:

Инкапсуляция – это механизм, объединяющий данные и обрабатывающий их код как единое целое. Инкапсуляцией называется включение различных мелких элементов в более крупный объект, в результате чего программист работает непосредственно с этим объектом. Это приводит к упрощению программы, поскольку из нее исключаются второстепенные детали.

Полиморфизм позволяет использовать одни и те же имена для похожих, но технически разных задач. Главным в полиморфизме является то, что он позволяет манипулировать объектами путем создания стандартных интерфейсов для схожих действий. Полиморфизм значительно облегчает написание сложных программ.

Наследование позволяет одному объекту приобретать свойства другого объекта, при наследовании точная копия дополняется уникальными свойствами, которые характерны только для производного объекта [10].

Кроме того, при разработке электронной библиотеки была использована концепция Model-view-controller (MVC, «модель-представление-контроллер). Это схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные.

Основная цель применения этой концепции состоит в разделении бизнес-логики (модели) от её визуализации (представления, вида). За счет такого разделения повышается возможность повторного использования. К одной модели можно присоединить несколько видов, не затрагивая при этом реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы. Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), для этого достаточно использовать другой контроллер.

Модель (англ. Model) предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.

Представление, вид (англ. View) отвечает за отображение информации (визуализацию). Часто в качестве представления выступает форма (окно) с графическими элементами.

Контроллер (англ. Controller) обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

Важно отметить, что как представление, так и контроллер зависят от модели. Однако модель не зависит ни от представления, ни от контроллера. Тем самым достигается назначение такого разделения: оно позволяет строить модель независимо от визуального представления, а также создавать несколько различных представлений для одной модели [8].

Опишем процесс разработки проекта. С самого начала необходимо создать в папке проекта соответствующую структуру папок, которая изображена на рисунке 2.11

В папке public содержатся все публичные файлы и подпапки:

.htaccess – файл настройки веб-сервера Apache;

index.html – стартовая страница проекта;

favicon.ico – иконка сайта проекта;

asset – в папке содержатся шрифты иконок для пунктов главного меню.

Создаём файл composer.json и указываем в нём зависимости, необходимые для дальнейшей работы.

Листинг указанных файлов .htaccess, index.html и composer.json приведены в приложении В.

Используя composer, загружаем и подключаем все необходимые библиотеки.

В проекте использованы две новых библиотеки:

- texlab/mydb – для работы с БД MySQL.

- texlab/html – для динамического создания визуальных элементов на страницах сайта.

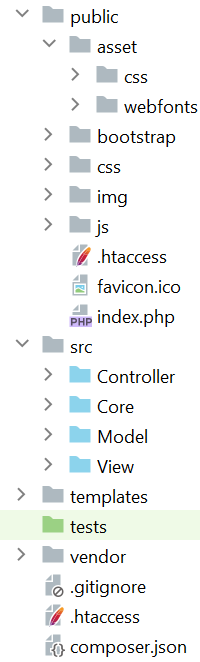


Рисунок 2.11 – Структура папок проекта

В папке src/core создаём два класса: Config и Dispatcher. Первый содержит в своих полях константы для подключения к БД и некоторые статичные данные необходимые для корректной работы системы. Второй необходим для диспетчеризации поступающих от пользователей запросов. Эти классы представлены на рисунке 2.12.

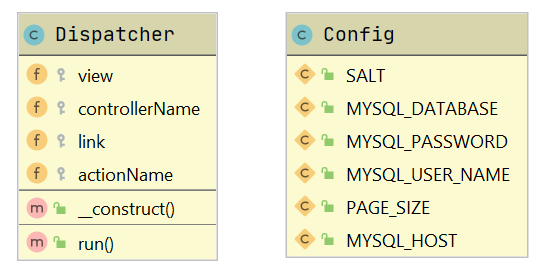


Рисунок 2.12 – Классы Dispatcher и Config

Листинг этих файлов приведены также в приложении В.

Далее приступаем к разработке классов-контроллеров. На этом этапе создается абстрактный уровень описания объектов. Диаграмма абстрактных классов-контроллеров представлена на рисунке 2.13.

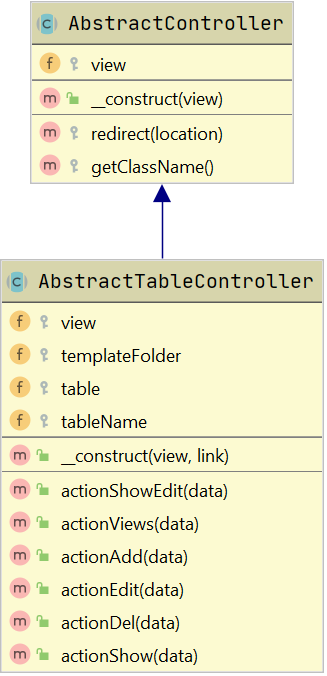


Рисунок 2.13 – Абстрактные классы-контроллеры

После этого создаём классы-контроллеры для всех сущностей. Диаграмма этих классов-контроллеров представлена в приложении В.

На основе библиотеки texlab/mydb создаём классы-модели для взаимодействия с БД. Диаграмма этих классов приведена на рисунке 2.14.

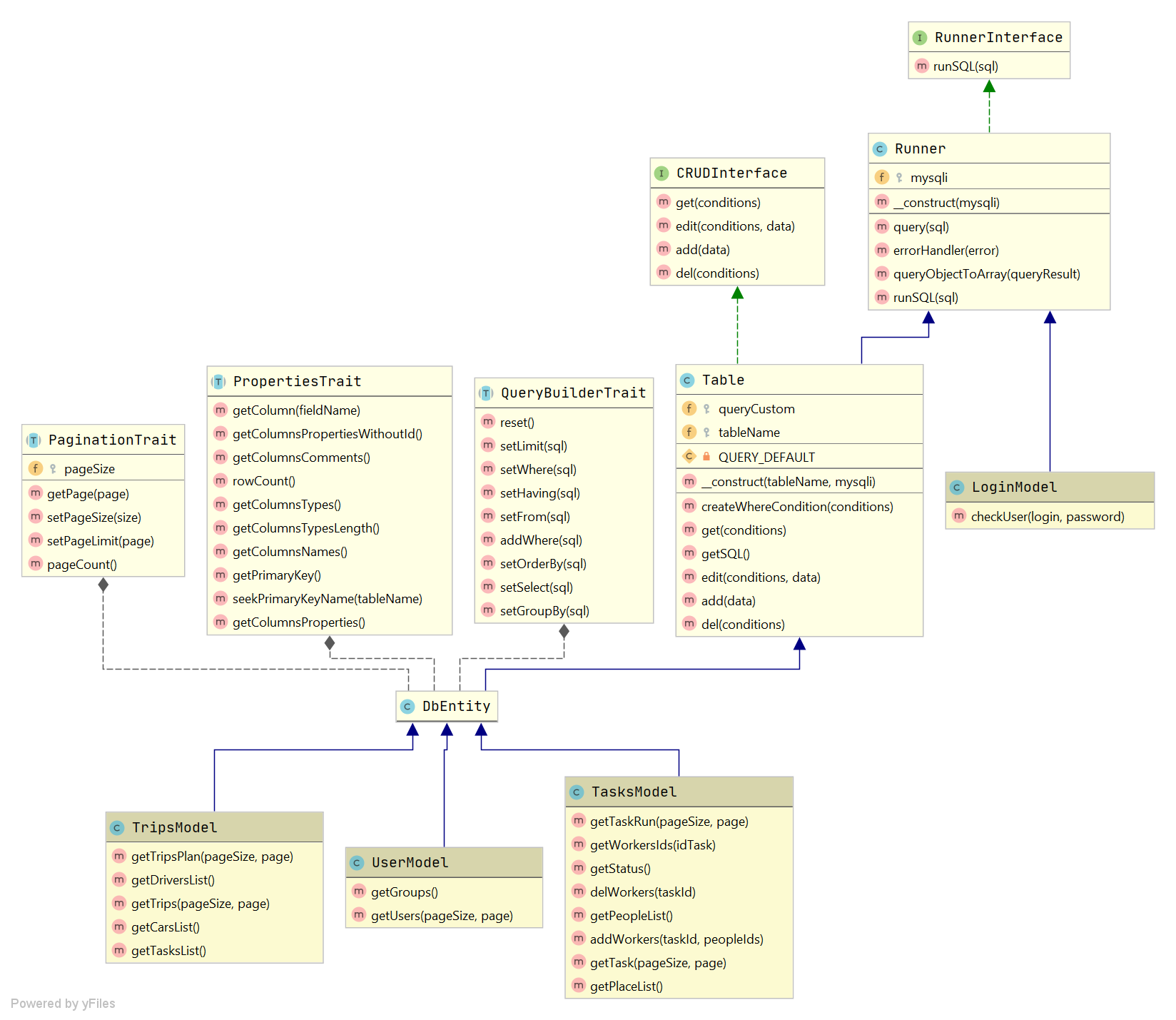


Рисунок 2.14 – Диаграмма классов-моделей на основе библиотеки texlab/mydb.

Далее создается универсальный класс для отображения страниц – View.

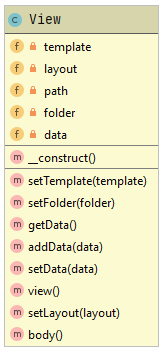


Рисунок 2.15 – Класс View.

Программный код перечисленных классов приводится в приложении Д.

**2.3 Описание интерфейса программного продукта**

В разрабатываемом проекте оформление и отображение UI основано на фреймворке Bootstrap 4, который поддерживает адаптивность. Bootstrap (также известен как Twitter Bootstrap) — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. [9]

Основные инструменты Bootstrap:

Сетки — заранее заданные размеры колонок, которые можно сразу же использовать, например, ширина колонки 140 px относится к классу .span2 (.col-md-2 в третьей версии фреймворка), который можно использовать в CSS-описании документа.

Шаблоны — фиксированный или резиновый шаблон документа.

Типографика — описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. п.

Медиа — предоставляет некоторое управление изображениями и видео.

Таблицы — средства оформления таблиц, вплоть до добавления функциональности сортировки.

Формы — классы для оформления форм и некоторых событий, происходящих с ними.

Навигация — классы оформления для панелей, вкладок, перехода по страницам, меню и панели инструментов.

Алерты — оформление диалоговых окон, подсказок и всплывающих окон.

Для отображения иконок в пунктах главного меню в проект внедрены шрифты значков. Подпапки css и webfonts в папке public/assets.

Создание веб-страниц начинается с создания макетов (layout), стилей (css), шаблонов (templates), меню (menu). Файлы разметки (макеты), шаблоны и меню, созданные в процессе работы над проектом показаны на рисунке 2.16.

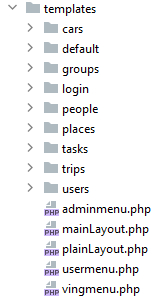


Рисунок 2.16 – Макеты, шаблоны и меню.

При генерации HTML-кода страницы авторизации используется макет plainLayout.php, а для всех остальных страниц – mainLayout.php.

Для разных категорий (групп) пользователей структура главного меню различна. Были созданы три варианта главного меню:

* adminmenu.php – для администратора;
* vingmenu.php – для руководителя;
* usermenu.php – для обычных пользователей.

В папке templates сложены подпапки, содержащие шаблоны страниц для отображения (show) и редактирования (edit) сущностей, выделенных при изучении предметной области. Кроме этого, там созданы шаблоны для страницы авторизации (loginform.php) и страницы отображаемой по умолчанию (default.php).

Первоначальная загрузка проекта начинается с файла index.html, затем в диспетчер классов – Dispatcher.php, в котором происходит перенаправление в файл, в котором реализован контроллер страницы авторизации. Алгоритм работы информационной системы приведён в приложении Б. Внешний вид страницы авторизации показан на рисунке 2.17.

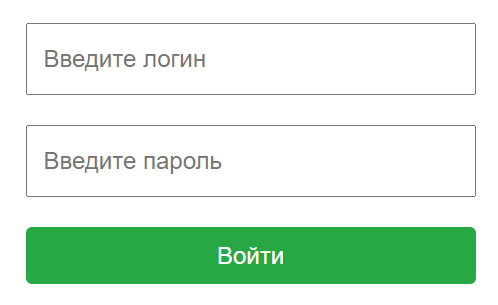


Рисунок 2.17 – Страница авторизации

После прохождения авторизации пользователь попадает на главную страницу. Главная страница различается в зависимости от группы пользователей.

Если пользователь является администратором, то у него появляется возможность добавлять, редактировать, удалять данные из базы. Администратор имеет полный доступ ко всем функциям системы управления АПГ. Для администраторов главной страницей является «Страница добавления задач». С этой страницы легко можно перейти на страницу «Редактирования задачи» или, воспользовавшись пунктами главного меню, перейти к любой другой странице сайта.

Только пользователи из группы «Администраторы» имеют возможность управления доступом к сайту. Для добавления пользователей и их групп администратору необходимо перейти в раздел «Доступ», нажав на одноименную кнопку, расположенную в верхнем меню страницы. Внешний вид меню «Доступ» представлен на рисунке 2.18.

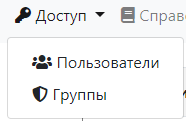


Рисунок 2.18 – Пункт «Доступ» главного меню.

Страницы добавления и редактирования пользователей приведены на рисунках 2.19 и 2.20.

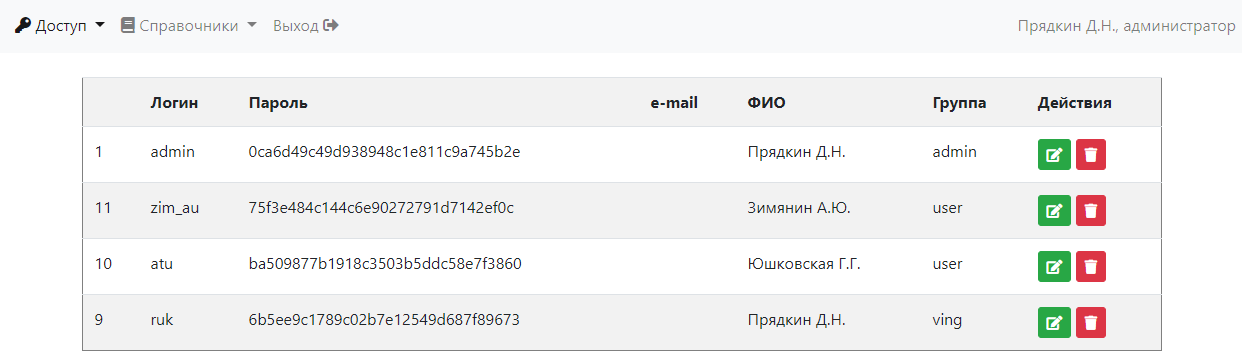




Рисунок 2.19 – Страница работы с учётными записями пользователей программной системы.

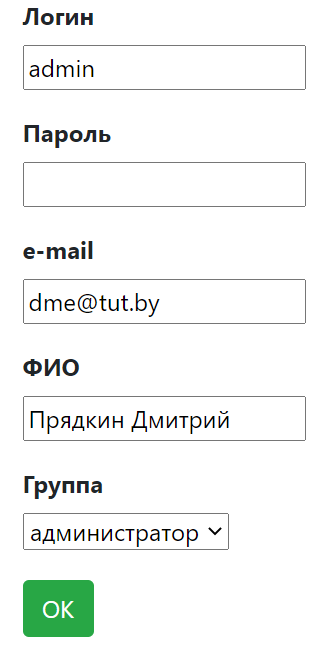


Рисунок 2.20 – Страница редактирования данных о пользователе.

Страницы просмотра групп пользователей сайта приведена на рисунке 2.21.

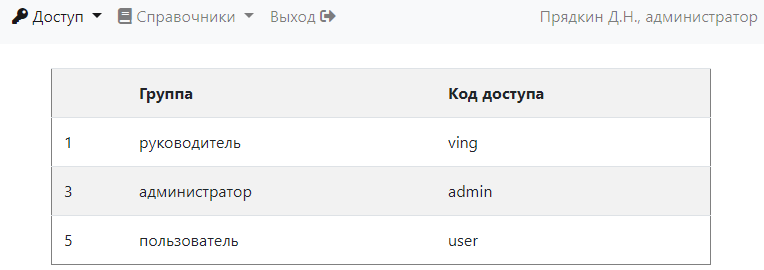


Рисунок 2.21 – Страница добавления групп пользователей.

Администратор также имеет возможность добавлять локации (места проведения работ), транспорт, персонал, которые в последствии отображаются в списках при добавлении и редактировании задач и выездов.

Для добавления локаций, транспорта, персонала администратору необходимо перейти в раздел «Справочники», нажав на одноименную кнопку, расположенную в верхнем меню главной страницы. Внешний вид меню «Справочники» представлен на рисунке 2.22.

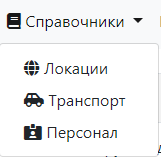


Рисунок 2.22 – Пункт «Справочники» главного меню.

Страницы добавления и редактирования локаций приведены на рисунках 2.23 и 2.24.

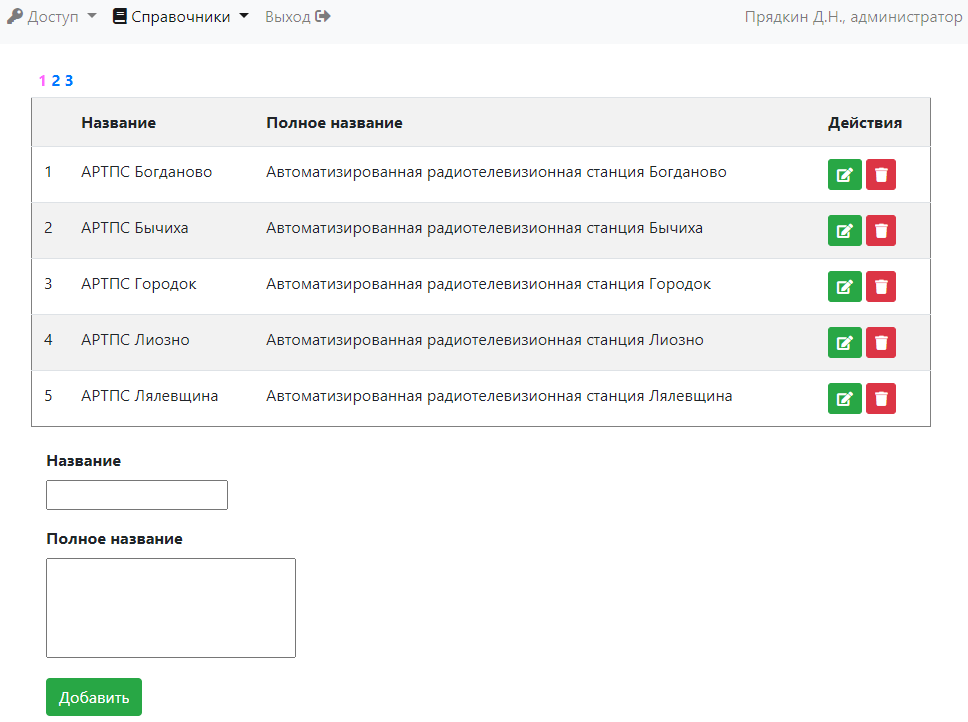


Рисунок 2.23 – Страница добавления локаций.

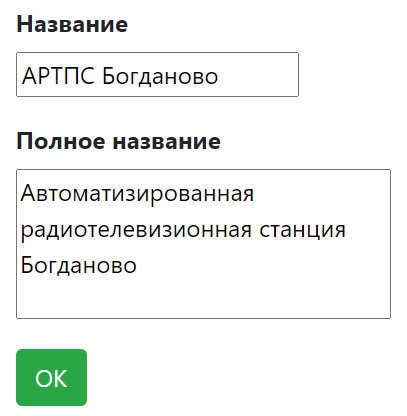


Рисунок 2.24 – Страница редактирования локации.

Страницы добавления и редактирования транспорта приведены на рисунках 2.25 и 2.26.

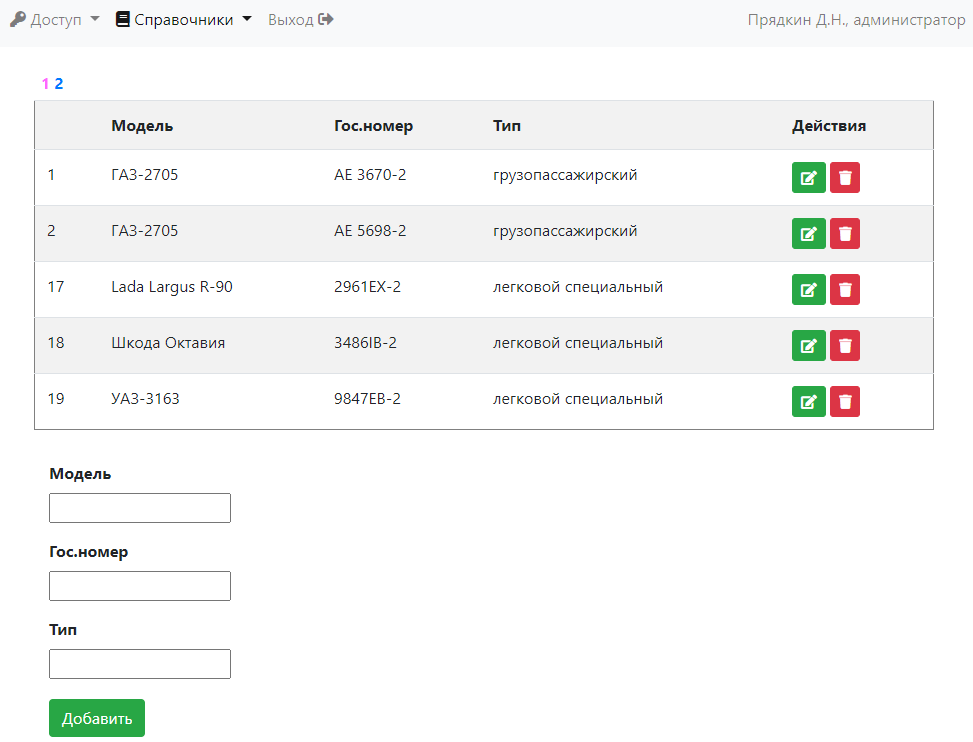


Рисунок 2.25 – Страница добавления транспорта.

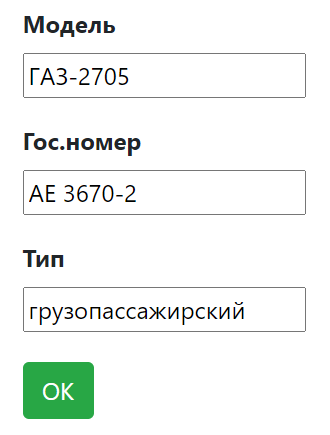
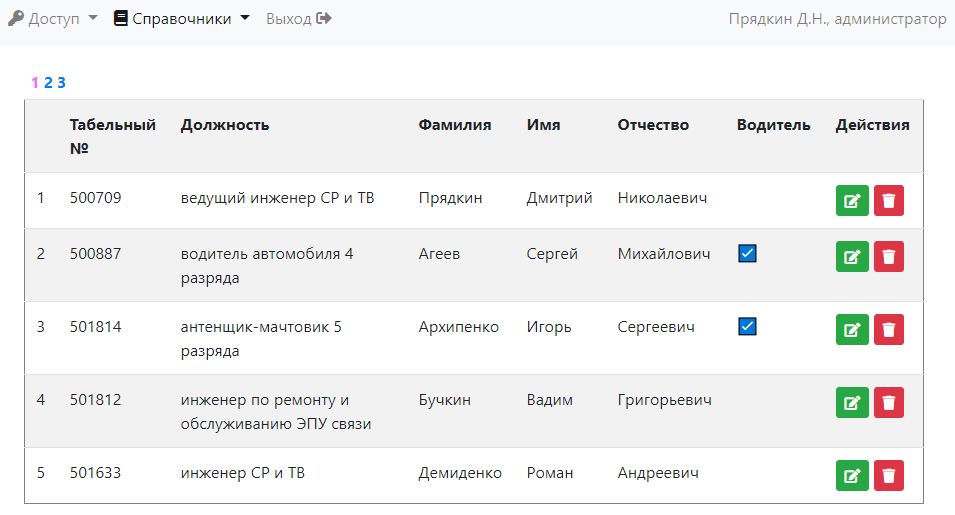


Рисунок 2.26 – Страница редактирования транспорта.

Страницы добавления и редактирования данных о сотрудниках приведены на рисунках 2.27 и 2.28.



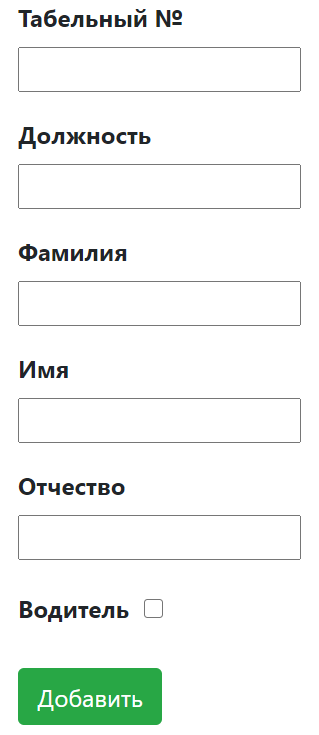


Рисунок 2.27 – Страница добавления данных о сотрудниках.

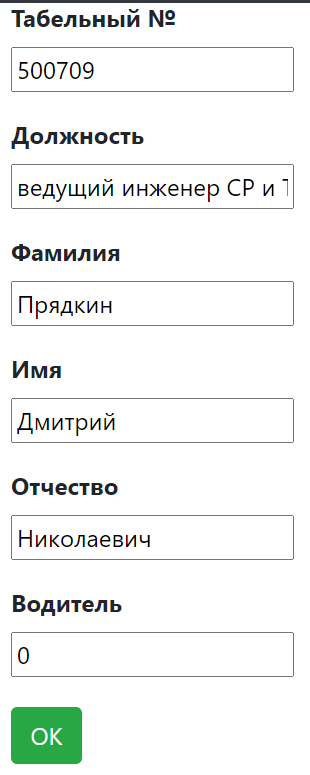


Рисунок 2.28– Страница редактирования данных сотрудника.

Для добавления заданий и поездок автотранспорта администратору необходимо перейти в раздел «Работы», нажав на одноименную кнопку, расположенную в верхнем меню главной страницы. Внешний вид меню «Работы» представлен на рисунке 2.29.

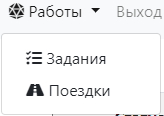
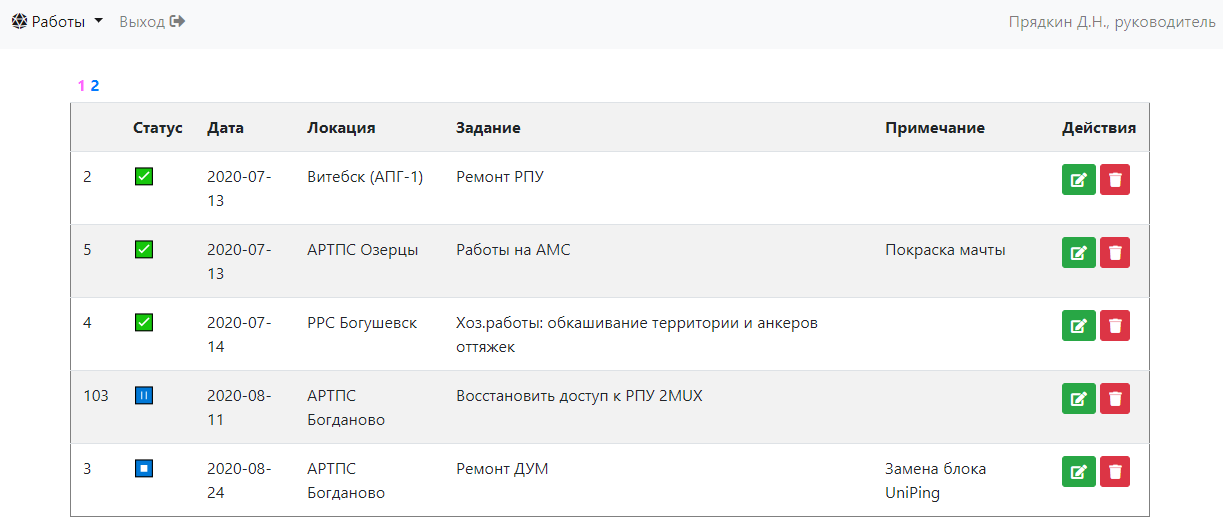
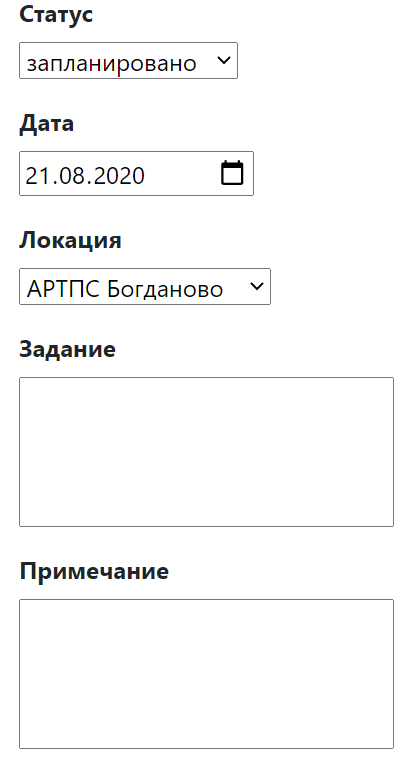


Рисунок 2.29 – Пункт «Работы» главного меню.

Внешний вид окон добавления и редактирования заданий, представлен на рисунках 2.30 и 2.31.





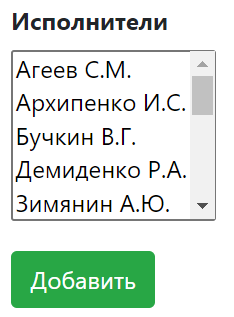
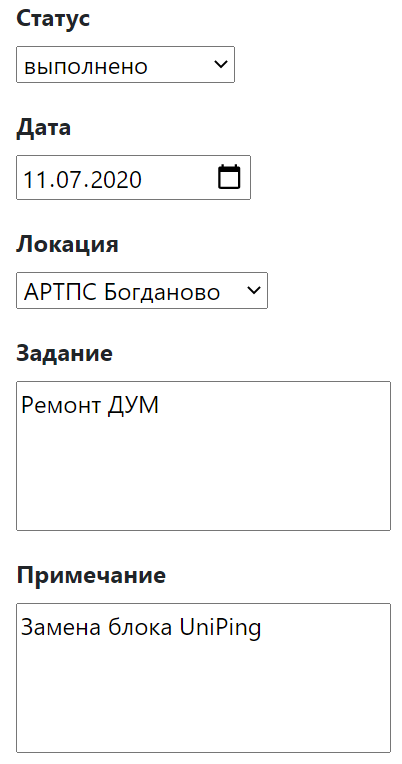


Рисунок 2.30 – Страница работы с заданиями.



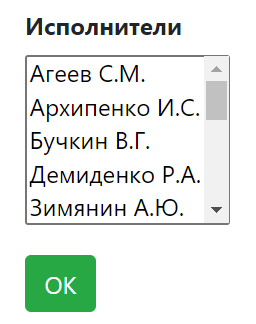


Рисунок 2.31 – Страница редактирования задачи.

Внешний вид окон добавления и редактирования поездок, представлен на рисунках 2.32 и 2.33.

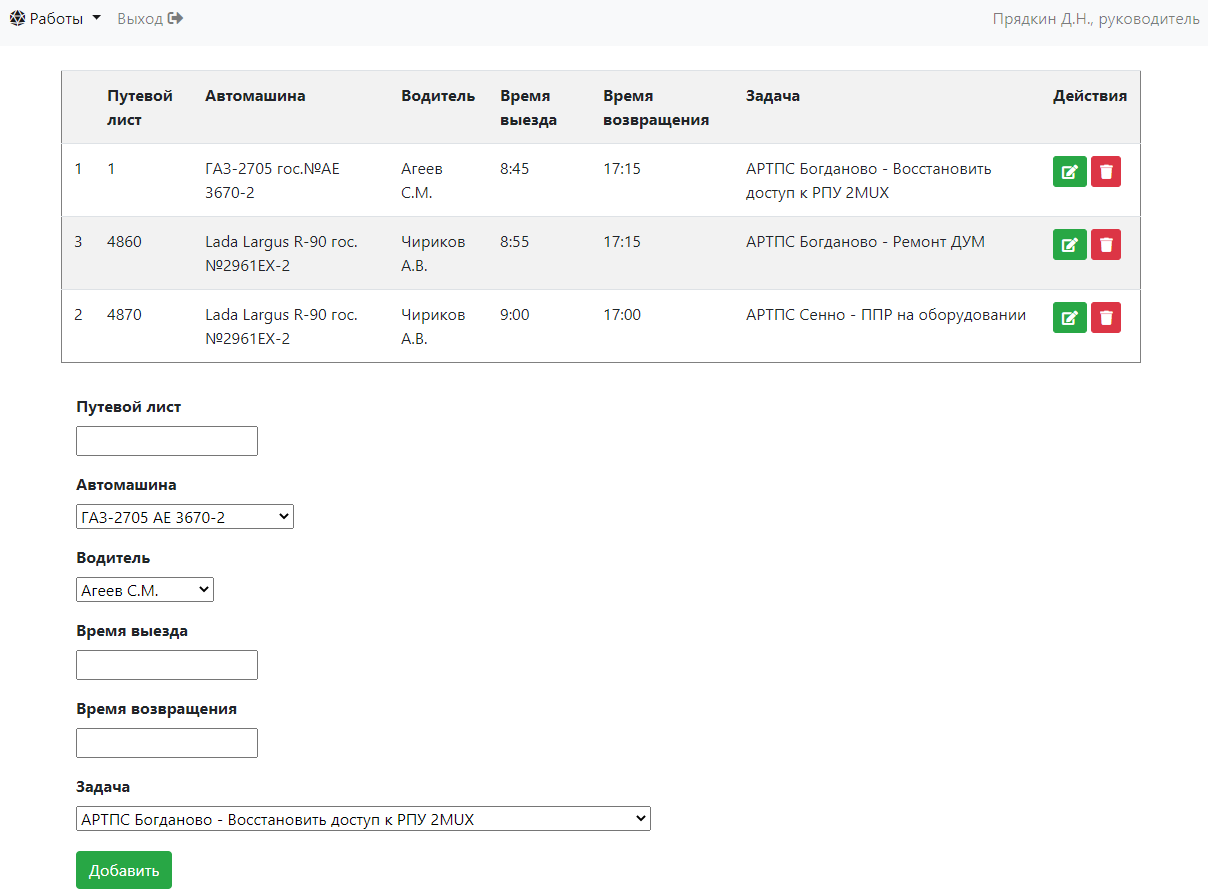


Рисунок 2.32 – Страница добавления данных о поездках.

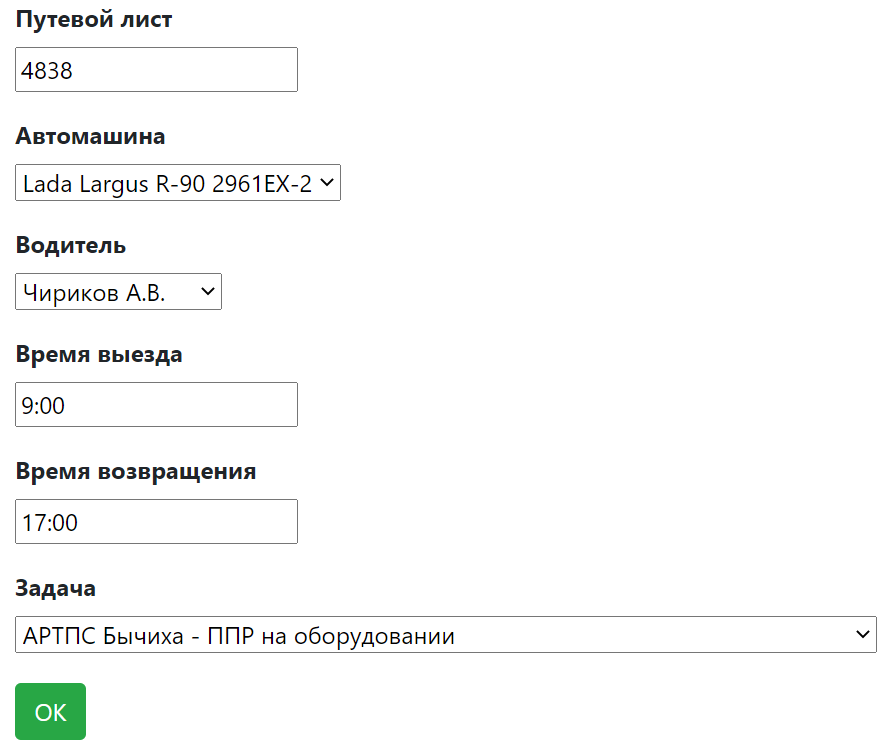


Рисунок 2.33– Страница редактирования данных о поездке.

Для обычных пользователей функции системы ограничены, так как они могут только просматривать информацию о незавершённых заданиях и поездках. Внешний вид страниц просмотра заданий и выездов показаны на рисунке 2.34 и 2.35.

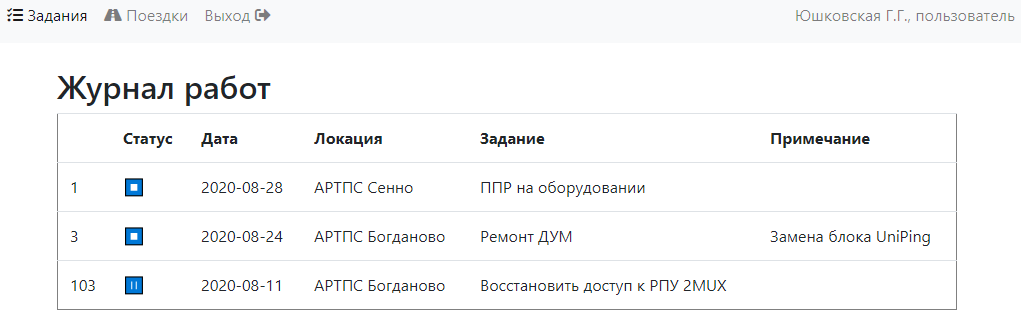


Рисунок 2.34 – Страница просмотра незавершённых заданиях.

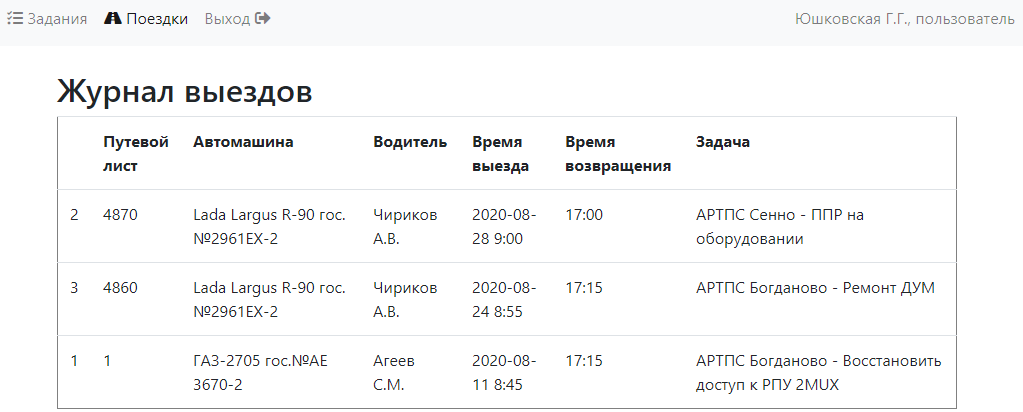


Рисунок 2.35 – Страница просмотра данных о поездках.

Таким образом, были описаны основные страницы информационной системы управления аварийно-профилактической группой, доступные как администратору, так и обычным пользователям. Достоинствами проекта, является простота администрирования: добавления, удаления и редактирования данных о работах и поездках и приятный дизайн UI.

# 2.4 Тестирование информационной системы управления аварийно-профилактической группой

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий две различные цели:

* продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
* выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации [11].

Комплексное решение задач обеспечения качества программных средств предполагает разработку и внедрение той или иной системы управления качеством. В мировой практике наибольшее распространение получила система, основанная на международных стандартах серии ISO 9000, включающей, в том числе стандарт, регламентирующий обеспечение качества ПО (ISO 9000/3). Эти стандарты должны служить руководством для ведущих специалистов компаний, разрабатывающих ПО на заказ.

Определения характеристик и субхарактеристик качества (ISO 9126-1):

**Функциональные** возможности – способность программного средства обеспечивать решение задач, удовлетворяющих сформулированные потребности заказчиков и пользователей при применении комплекса программ в заданных условиях.

Задачей, было создать информационную систему электронной библиотеки соответствующий следующим техническим требованиям:

* наличие системы авторизации и разграничения прав доступа к содержимому проекта;
* гибкость и удобство использования;
* возможность добавлять, редактировать, удалять книги и другую информацию (для администратора) и искать, читать, скачивать, распечатывать (для пользователей) необходимые книги и справочные материалы.

Разработанная информационная система электронной библиотеки соответствует всем этим требованиям: она представляет собой веб-проект, оснащенный системами авторизации, а также возможностью добавлять, редактировать, удалять книги и другую информацию (для администратора) и искать, читать, скачивать, распечатывать (для пользователей) необходимые книги и справочные материалы. Проект простой в использовании и интуитивно понятен как для пользователя, так и для администратора.

**Правильность** (корректность) – способность программного средства обеспечивать правильные или приемлемые для пользователя результаты и внешние эффекты.

Правильность функционирования информационной системы электронной библиотеки проверялась как в процессе её разработки (промежуточное тестирование корректности функционирования), так и на заключительном этапе на локальной машине. По ходу тестирования в код вводились некоторые правки.

**Защищенность** – способность компонентов программного средства защищать программы и информацию от любых негативных воздействий.

Благодаря функционированию системы разграничения доступа, у пользователя нет возможности каким-либо образом менять содержание библиотеки, препятствуя правильному его функционированию.

**Практичность** (применимость) – свойства программного средства, обусловливающие сложность его понимания, изучения и использования, а также привлекательность для квалифицированных пользователей при применении в указанных условиях.

Разработанная информационная система электронной библиотеки достаточно проста в освоении, интуитивно понятна как в пользовательской, так и в административной части.

**Сопровождаемость** – приспособленность программного средства к модификации и изменению конфигурации и функций.

Благодаря использованию MVC модели наш программный продукт является масштабируемым, можно легко добавить новые функции или модифицировать старые, не меняя логику работы приложения. Возможно также повторное использование кода.

Большая трудоемкость тестирования и ограниченные ресурсы приводят к необходимости систематизации процесса и методов тестирования. Тестирование должен проходить любой программный продукт тем более такой, как прикладные программы информационной системы. Помимо обнаружения имеющихся в прикладных программах и, возможно, в структурах базы данных ошибок, сбор статистических данных на стадии тестирования позволяет установить показатели надежности и качества созданного программного обеспечения. Журнал тестирования веб-приложения представлен в таблице 2.3.

Таблица 2.3 – Журнал тестирования веб-приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название  модуля/экрана | Описание тестового случая | Ожидаемые результаты | Тестовый случай пройден? | Комментарии |
| 1 | 2 | 3 | 4 | 5 |
| Страница авторизации – Кнопка «Войти» | Ввод логина и пароля в текстовое поле | Авторизация пользователя и вход в веб-приложения | Да | Работает корректно |
| Главная страница – Выбор пункта меню | Выбор одинарным щелчком мыши | Отображение данных, соответствующих пунктам меню | Да | Работает корректно |
| Меню «Пользователи» – Кнопка «Добавить» | Выбор группы одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Пользователи» – Кнопка «Редактировать» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Пользователи» – Кнопка «Удалить» | Выбор одинарным щелчком мыши | Удаление данных о записи | Да | Работает корректно |

Продолжение таблицы 2.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Меню «Пользователи» – Кнопка «ОК» | Выбор одинарным щелчком мыши | Добавление данных в базу | Да | Работает корректно |
| Меню «Группы» – Кнопка «Добавить» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Группы» – Кнопка «Редактировать» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Группы» – Кнопка «Удалить» | Выбор одинарным щелчком мыши | Удаление данных о записи | Да | Работает корректно |
| Меню «Группы» – Кнопка «ОК» | Выбор одинарным щелчком мыши | Добавление данных в базу | Да | Работает корректно |
| Меню «Локации» – Кнопка «Добавить» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Локации» – Кнопка «Редактировать» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Локации» – Кнопка «Удалить» | Выбор одинарным щелчком мыши | Удаление данных о записи | Да | Работает корректно |
| Меню «Локации» – Кнопка «ОК» | Выбор одинарным щелчком мыши | Добавление данных в базу | Да | Работает корректно |
| Меню «Транспорт» – Кнопка «Добавить» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |

Продолжение таблицы 2.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Меню «Транспорт» – Кнопка «Редактировать» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Транспорт» – Кнопка «Удалить» | Выбор одинарным щелчком мыши | Удаление данных о записи | Да | Работает корректно |
| Меню «Транспорт» – Кнопка «ОК» | Выбор одинарным щелчком мыши | Добавление данных в базу | Да | Работает корректно |
| Меню «Персонал» – Кнопка «Добавить» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Персонал» – Кнопка «Редактировать» | Выбор одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Персонал» – Кнопка «Удалить» | Выбор одинарным щелчком мыши | Удаление данных о записи | Да | Работает корректно |
| Меню «Персонал» – Кнопка «ОК» | Выбор одинарным щелчком мыши | Добавление данных в базу | Да | Работает корректно |
| Меню «Задачи» – Кнопка «Добавить» | Выбор статуса, локации, даты одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Задачи» – Кнопка «Редактировать» | Выбор статуса, локации, даты одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |

Продолжение таблицы 2.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Меню «Задачи» – Кнопка «Удалить» | Выбор одинарным щелчком мыши | Удаление данных о записи | Да | Работает корректно |
| Меню «Задачи» – Кнопка «ОК» | Выбор одинарным щелчком мыши | Добавление данных в базу | Да | Работает корректно |
| Меню «Поездки» – Кнопка «Добавить» | Выбор автомашины, водителя, задачи одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Поездки» – Кнопка «Редактировать» | Выбор автомашины, водителя, задачи одинарным щелчком мыши | Заполнение полей | Да | Работает корректно |
| Меню «Поездки» – Кнопка «Удалить» | Выбор одинарным щелчком мыши | Удаление данных о записи | Да | Работает корректно |
| Меню «Поездки» – Кнопка «ОК» | Выбор одинарным щелчком мыши | Добавление данных в базу | Да | Работает корректно |

После тестирования основных частей веб-приложения, было выявлено, что приложение отвечает заданным требованиям, что является положительным результатом.

Все выявленные ошибки были устранены.

**Выводы:**

Таким образом, были успешно пройдены все этапы разработки информационной системы управления аварийно-профилактической группой:

* спроектирована структура базы данных и реализована на практике;
* разработаны и реализованы основные классы в рамках MVC модели;
* описаны особенности интерфейса информационной системы и представлены основные типы страниц;
* проведен анализ функциональности образовательно-информационного ресурса на основе его соответствия характеристикам и субхарактеристикам качества (ISO 9126-1).

# Заключение

В процессе разработки дипломного проекта были пройдены все основные этапы разработки информационной системы: описание предметной области, проектирование программного комплекса, программная реализация и тестирование программного обеспечения. Результатом стало создание системы управления аварийно-профилактической группой, оснащенной системой авторизации, а также возможностью добавлять, редактировать, удалять задачи и другую информацию (для администратора) и просматривать (для пользователей) ведомости выездов автотранспорта.

В процессе выполнения дипломной работы были решены следующие задачи:

* анализ средств разработки показал, что для решения поставленных задач целесообразно использование PHP, а так же СУБД My SQL;
* обозначена предметная область разрабатываемой системы управления аварийно-профилактической группой, определены функции, которые она выполняет и информационная составляющая;
* проведено концептуальное, логическое моделирование базы данных;
* разработаны диаграммы UML.

В работе были проведены описание предметной области, анализ систем тестирования, разработана концепция сайта, проведено информационное проектирование, разработаны алгоритмы функционирования программных модулей, разработана программная часть информационной системы управления аварийно-профилактической группой.

Веб-приложение обладает следующей функциональностью:

* ввод данных в базу и их изменение;
* удаление записей из базы данных;
* извлечение информации из базы данных.

Разработан дружественный интерфейс, который удобен и прост, что делает данный программный продукт лёгким в использовании.

Разработанное веб-приложение не требовательно к ресурсам и может работать под любой операционной системой с веб-сервером.

Таким образом, в дипломной работе удалось реализовать поставленные цели и задачи. В качестве дальнейшего совершенствования программного средства при необходимости возможно увеличение его функциональности.

# Список использованных источников

1. Википедия. PHP [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/PHP>.– Дата доступа: 20.07.2020.

2. Википедия. XHTML [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/XHTML>. – Дата доступа: 01.07.2019.

3. Википедия. CSS [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/CSS. – Дата доступа: 01.05.2020.

4. Википедия. MySQL. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/MySQL> – Дата доступа: 01.05.2020.

5. Википедия. UML [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/UML. – Дата доступа: 18.07.2019.

6. Open Server Panel [Электронный ресурс]. – Режим доступа: <https://ospanel.io/> – Дата доступа: 20.07.2020.

7. PhpStorm [Электронный ресурс] – Режим доступа: <https://www.jetbrains.com/ru-ru/phpstorm/> – Дата доступа: 15.06.2020.

8. Википедия. Model-view-controller [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Model-view-controller> – Дата доступа: 02.04.2020.

9. Википедия. Bootstrap (фреймворк) [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Bootstrap_(фреймворк)> – Дата доступа: 02.04.2020.

10. Учебно-методический комплекс дисциплины «Технологии баз данных и знаний». Этапы проектирования базы данных и их процедуры [Электронный ресурс]. – Режим доступа <http://www.bseu.by/it/tohod/lekcii4_6.htm> – Дата доступа: 20.07.2020.

11. Википедия. Тестирование программного обеспечения [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Тестирование программного обеспечения](https://ru.wikipedia.org/wiki/Тестирование%20программного%20обеспечения) – Дата доступа: 10.08.2020.

# ПРИЛОЖЕНИЕ А

**Листинг скрипта создания базы данных «wb\_db»**

--

-- Структура таблицы `cars`

--

CREATE TABLE `cars` (

`id` int NOT NULL,

`model` varchar(45) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Модель',

`GosNum` varchar(10) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Гос.номер',

`type` varchar(45) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Тип'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Структура таблицы `groups`

--

CREATE TABLE `groups` (

`id` int NOT NULL,

`name` varchar(50) COLLATE utf8\_bin NOT NULL COMMENT 'Группа',

`cod` varchar(20) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Код доступа'

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

--

-- Дамп данных таблицы `groups`

--

INSERT INTO `groups` (`id`, `name`, `cod`) VALUES

(1, 'пользователь', 'user'),

(2, 'руководитель', 'ving'),

(3, 'администратор', 'admin');

--

-- Структура таблицы `people`

--

CREATE TABLE `people` (

`id` int NOT NULL,

`tabNum` varchar(6) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Табельный №',

`position` varchar(45) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Должность',

`firstName` varchar(45) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Фамилия',

`name` varchar(45) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Имя',

`fatherName` varchar(45) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL COMMENT 'Отчество',

`driver` tinyint(1) NOT NULL COMMENT 'Водитель'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Структура таблицы `places`

--

CREATE TABLE `places` (

`id` int NOT NULL,

`name` varchar(30) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Название',

`FullName` varchar(128) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL COMMENT 'Полное название'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Структура таблицы `tasks`

--

CREATE TABLE `tasks` (

`id` int NOT NULL,

`status` tinyint UNSIGNED NOT NULL COMMENT 'Статус',

`date` date NOT NULL COMMENT 'Дата',

`places\_id` int NOT NULL COMMENT 'Локация',

`content` text CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Задание',

`comment` text CHARACTER SET utf8 COLLATE utf8\_bin COMMENT 'Примечание'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Структура таблицы `taskstates`

--

CREATE TABLE `taskstates` (

`id` tinyint UNSIGNED NOT NULL COMMENT 'id',

`name` varchar(50) COLLATE utf8\_bin NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

--

-- Дамп данных таблицы `taskstates`

--

INSERT INTO `taskstates` (`id`, `name`) VALUES

(1, 'запланировано'),

(2, 'выполняется'),

(3, 'приостановлено'),

(4, 'выполнено');

-- --------------------------------------------------------

--

-- Структура таблицы `trips`

--

CREATE TABLE `trips` (

`id` int NOT NULL,

`PutNum` int NOT NULL COMMENT 'Путевой лист',

`cars\_id` int NOT NULL COMMENT 'Автомашина',

`people\_id` int NOT NULL COMMENT 'Водитель',

`timeStart` varchar(12) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL COMMENT 'Время выезда',

`timeFinish` varchar(12) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL COMMENT 'Время возвращения',

`tasks\_id` int NOT NULL COMMENT 'Задача'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Структура таблицы `users`

--

CREATE TABLE `users` (

`id` int NOT NULL,

`login` varchar(50) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Логин',

`password` text CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'Пароль',

`email` varchar(255) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL COMMENT 'e-mail',

`name` varchar(150) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL COMMENT 'ФИО',

`group\_id` int NOT NULL COMMENT 'Группа'

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;

--

-- Дамп данных таблицы `users`

--

INSERT INTO `users` (`id`, `login`, `password`, `email`, `name`, `group\_id`) VALUES

(1, 'admin', '2ba833da9cae94b164a7dbb31d4c8866', '', 'Прядкин Дмитрий', 3),

(2, 'user', 'ba509877b1918c3503b5ddc58e7f3860', '', 'Диспетчер', 1),

(3, '', '72eb55db4c1b2a17c2196dc3a56497e0', '', 'Гость', 2);

-- --------------------------------------------------------

--

-- Структура таблицы `workers`

--

CREATE TABLE `workers` (

`id` int NOT NULL,

`people\_id` int NOT NULL,

`tasks\_id` int NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Индексы сохранённых таблиц

--

--

-- Индексы таблицы `cars`

--

ALTER TABLE `cars`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `idcars\_UNIQUE` (`id`),

ADD UNIQUE KEY `GNum\_UNIQUE` (`GosNum`);

--

-- Индексы таблицы `groups`

--

ALTER TABLE `groups`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `group\_name` (`name`);

--

-- Индексы таблицы `people`

--

ALTER TABLE `people`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `TabNum\_UNIQUE` (`tabNum`),

ADD UNIQUE KEY `idPeople\_UNIQUE` (`id`);

--

-- Индексы таблицы `places`

--

ALTER TABLE `places`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `idplaces\_UNIQUE` (`id`);

--

-- Индексы таблицы `tasks`

--

ALTER TABLE `tasks`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `idTask\_UNIQUE` (`id`);

--

-- Индексы таблицы `taskstates`

--

ALTER TABLE `taskstates`

ADD PRIMARY KEY (`id`);

--

-- Индексы таблицы `trips`

--

ALTER TABLE `trips`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `idtrips\_UNIQUE` (`id`);

--

-- Индексы таблицы `users`

--

ALTER TABLE `users`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `login` (`login`);

--

-- Индексы таблицы `workers`

--

ALTER TABLE `workers`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `idworkers\_UNIQUE` (`id`);

--

-- AUTO\_INCREMENT для сохранённых таблиц

--

--

-- AUTO\_INCREMENT для таблицы `cars`

--

ALTER TABLE `cars`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=21;

--

-- AUTO\_INCREMENT для таблицы `groups`

--

ALTER TABLE `groups`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;

--

-- AUTO\_INCREMENT для таблицы `people`

--

ALTER TABLE `people`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=18;

--

-- AUTO\_INCREMENT для таблицы `places`

--

ALTER TABLE `places`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=14;

--

-- AUTO\_INCREMENT для таблицы `tasks`

--

ALTER TABLE `tasks`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=112;

--

-- AUTO\_INCREMENT для таблицы `taskstates`

--

ALTER TABLE `taskstates`

MODIFY `id` tinyint UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'id', AUTO\_INCREMENT=5;

--

-- AUTO\_INCREMENT для таблицы `trips`

--

ALTER TABLE `trips`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `users`

--

ALTER TABLE `users`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- AUTO\_INCREMENT для таблицы `workers`

--

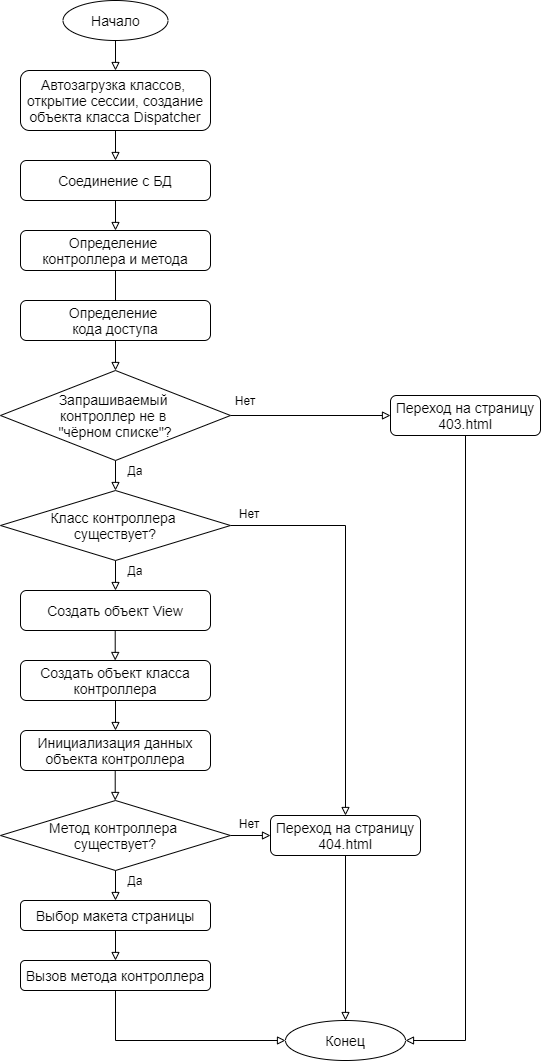
ALTER TABLE `workers`

MODIFY `id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=105;

COMMIT;

# ПРИЛОЖЕНИЕ Б

**Алгоритм работы информационной системы**



# ПРИЛОЖЕНИЕ В

**Листинг файла «.htaccess»**

**RewriteEngine On  
RewriteRule ^.\*$ /public/$0 [QSA,L]**

**Листинг файла «index.html»**

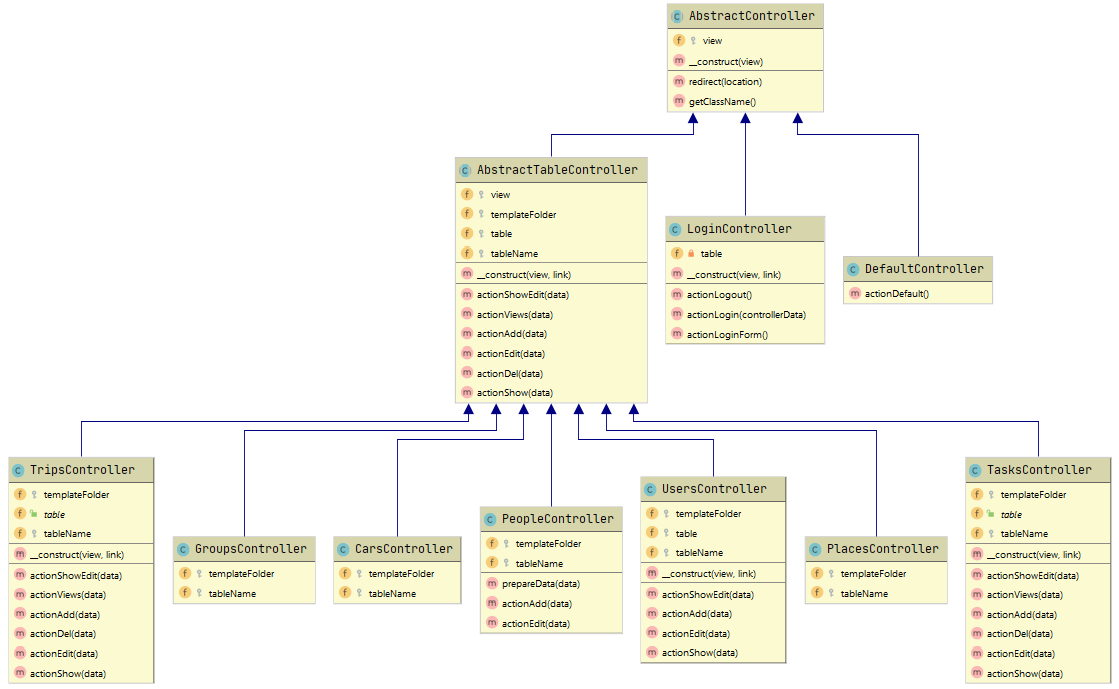
<?php  
  
*session\_start*();  
  
require '../vendor/autoload.php';  
  
(new Core\Dispatcher())->run();

**Листинг файла «composer.json»**

{  
 **"prefer-stable"**: **true**,  
 **"require"**: {  
 **"texlab/mydb"**: **"^0.0.4"**,  
 **"ext-mysqli"**: **"\*"**,  
 **"texlab/html"**: **"^0.09"** },  
 **"autoload"**: {  
 **"psr-4"**: {  
 **"Controller\\"**: **"src/Controller"**,  
 **"Core\\"**: **"src/Core"**,  
 **"View\\"**: **"src/View"**,  
 **"Model\\"**: **"src/Model"** }  
 },  
 **"require-dev"**: {  
 **"phpunit/phpunit"**: **"^9.2"**,  
 **"phpstan/phpstan"**: **"^0.12"**,  
 **"vimeo/psalm"**: **"^3.0@dev"**,  
 **"squizlabs/php\_codesniffer"**: **"3.\*"** },  
 **"minimum-stability"**: **"dev"**,  
 **"scripts"**: {  
 **"test"**: **"phpunit --testsuite alltests"**,  
 **"phpstan"**: **"phpstan analyse -c phpstan.neon"**,  
 **"psalm"**: **"psalm"**,  
 **"codestyle"**: **"phpcs"**,  
 **"codestyle:fix"**: **"phpcbf"** }  
}

# ПРИЛОЖЕНИЕ Г

**Диаграмма классов-контроллеров**



# ПРИЛОЖЕНИЕ Д

**Листинг файла «Config.php»**

<?php  
  
namespace Core;  
  
class Config  
{  
 public const *MYSQL\_HOST* = 'localhost';  
 public const *MYSQL\_USER\_NAME* = 'root';  
 public const *MYSQL\_PASSWORD* = 'root';  
 public const *MYSQL\_DATABASE* = 'wb\_db';  
 public const *PAGE\_SIZE* = 5;  
 public const *SALT* = 'lalalala';  
}

**Листинг файла «Dispatcher.php»**

<?php  
  
namespace Core;  
  
use TexLab\MyDB\DB;  
use View\View;  
  
class Dispatcher  
{  
 protected $view;  
 protected $controllerName;  
 protected $actionName;  
 protected $link;  
  
 public function \_\_construct()  
 {  
 $this->link = DB::*Link*([  
 'host' => Config::*MYSQL\_HOST*,  
 'username' => Config::*MYSQL\_USER\_NAME*,  
 'password' => Config::*MYSQL\_PASSWORD*,  
 'dbname' => Config::*MYSQL\_DATABASE* ]);  
  
 $this->view = new View();  
 $this->controllerName = "Controller\\" . (*ucfirst*(*strtolower*($\_GET['type'] ?? 'login'))) . "Controller";  
$this->actionName = "action" . ($\_GET['action'] ?? 'loginform');  
 }

public function run()  
 {  
 $blacklist = include "blacklist.php";  
 $cod = $\_SESSION['user']['cod'] ?? 'guest';  
  
 if ($this->actionName == 'loginform') {  
 $this->view->setLayout('plainLayout');  
 } else {  
 $this->view->setLayout('mainLayout');  
 }  
  
 if (!(*in\_array*($\_GET['type'], $blacklist[$cod]))) {  
 if (*class\_exists*($this->controllerName)) {  
 $controller = new $this->controllerName(  
 $this->view,  
 $this->link  
 );  
 $controllerData = ['post' => $\_POST, 'get' => $\_GET];  
  
 if (*method\_exists*($controller, $this->actionName)) {  
 $controller->{$this->actionName}($controllerData);  
 $this  
 ->view  
 ->addData(['controllerType' => $\_GET['type']])  
 ->view();  
 } else {  
 *header*("HTTP/1.0 404 Not Found");  
 }  
 } else {  
 *header*("HTTP/1.0 404 Not Found");  
 }  
 } else {  
 *header*("HTTP/1.0 403 Forbidden");  
 }  
 }  
}

**Листинг файла «View.php»**

<?php  
  
namespace View;  
  
class View  
{  
 private string $layout;  
 private string $template;  
 private string $path;  
 private array $data = [];  
 private string $folder;  
  
 public function \_\_construct()  
 {  
 $this->path = $\_SERVER['DOCUMENT\_ROOT'] . *dirname*($\_SERVER['SCRIPT\_NAME']) . '/../templates/';  
 }  
  
 public function setLayout(string $layout)  
 {  
 $this->layout = $layout;  
 return $this;  
 }  
  
 public function setFolder(string $folder)  
 {  
 $this->folder = $folder . '/';  
 return $this;  
 }  
  
 public function setTemplate(string $template)  
 {  
 $this->template = $template;  
 return $this;  
 }  
  
 public function setData(array $data)  
 {  
 $this->data = $data;  
 return $this;  
 }  
  
 public function addData(array $data)  
 {  
 $this->data = *array\_merge*($this->data ?? [], $data);  
 return $this;  
 }  
  
 public function getData(): array  
 {  
 return $this->data;  
 }  
  
 public function view()  
 {  
 $controllerType = $this->data['controllerType'];  
 include "$this->path$this->layout.php";  
 }  
  
 public function body()  
 {  
 if (*is\_array*($this->data)) {  
 *extract*($this->data);  
 }  
 include "$this->path$this->folder$this->template.php";  
 }  
}

**Листинг файла «LoginModel.php»**

<?php  
  
namespace Model;  
  
use TexLab\MyDB\Runner;  
  
class LoginModel extends Runner  
{  
 public function checkUser(string $login, string $password)  
 {  
 return $this->runSQL(<<<SQL  
SELECT `groups`.`cod` , `users`.`name`, `groups`.`name` as 'group'  
FROM `users` , `groups`  
WHERE `users`.`group\_id` = `groups`.`id`  
AND BINARY `users`.`login` = '$login'  
AND BINARY `users`.`password` = '$password'  
SQL  
 )[0];  
 }  
}

**Листинг файла «TaskModel.php»**

<?php  
  
*/\*\* @var array $page  
 \* @var int $pageSize  
 \*/*namespace Model;  
  
use TexLab\MyDB\DbEntity;  
  
class TasksModel extends DbEntity  
{  
 public function getPlaceList(): array  
 {  
 $res = [];  
 foreach ($this->runSQL('SELECT `id`, `name` FROM `places`') as $row) {  
 $res[$row['id']] = $row['name'];  
 }  
 return $res;  
 }  
  
 public function getStatus(): array  
 {  
 $res = [];  
 foreach ($this->runSQL('SELECT `id`, `name` FROM `taskstates`') as $row) {  
 $res[$row['id']] = $row['name'];  
 }  
 return $res;  
 }  
  
 public function getPeopleList(): array  
 {  
 $res = [];  
 foreach (  
 $this->runSQL(  
 'SELECT `id`,`firstName`,`name`,`fatherName` FROM `people` ORDER BY `firstName`'  
 ) as $row  
 ) {  
 $res[$row['id']] = $row['firstName']  
 . ' '  
 . *mb\_substr*($row['name'], 0, 1)  
 . '.'  
 . *mb\_substr*($row['fatherName'], 0, 1)  
 . '.';  
 }  
return $res;  
 }  
  
 public function getWorkersIds(int $idTask): array  
 {  
 $res = [];  
 foreach (  
 $this->runSQL(  
 'SELECT `people\_id` FROM `workers` WHERE `tasks\_id`=' . $idTask  
 ) as $row  
 ) {  
 $res[] = $row['people\_id'];  
 }  
 return $res;  
 }  
  
 public function addWorkers(int $taskId, array $peopleIds)  
 {  
 $worker = new DbEntity('workers', $this->mysqli);  
 foreach ($peopleIds as $peopleId) {  
 $worker->add(  
 [  
 'people\_id' => $peopleId,  
 'tasks\_id' => $taskId  
 ]  
 );  
 }  
 }  
  
 public function delWorkers(int $taskId)  
 {  
 $workers = new DbEntity('workers', $this->mysqli);  
 $workerIds = $workers->runSQL(  
 "SELECT `id` FROM `workers` WHERE `tasks\_id`= $taskId"  
 );  
 foreach ($workerIds as $workerId) {  
 $workers->del(['id' => $workerId['id']]);  
 }  
 }  
  
 */\*\*  
 \* @param $pageSize  
 \* @param $page  
 \* @return array  
 \*/* public function getTask($pageSize, $page): array  
 {  
  
 return $this  
 ->setSelect('tasks.id, tasks.status, tasks.date, places.name, tasks.content, tasks.comment')  
 ->setFrom('tasks, places')  
 ->setWhere('places.id = tasks.places\_id')  
 ->setOrderBy('tasks.date')  
 ->setPageSize($pageSize)  
 ->getPage($page);  
 }  
  
 public function getTaskRun($pageSize, $page): array  
 {  
 return $this  
 ->setSelect('tasks.id, tasks.status, tasks.date, places.name, tasks.content, tasks.comment')  
 ->setFrom('tasks, places')  
 ->setWhere('places.id = tasks.places\_id AND tasks.status = 2')  
 ->setOrderBy('tasks.date')  
 ->setPageSize($pageSize)  
 ->getPage($page);  
 }  
}

**Листинг файла «TripsModel.php»**

<?php  
  
*/\*\* @var array $page  
 \* @var int $pageSize  
 \*/*namespace Model;  
  
use TexLab\MyDB\DbEntity;  
  
class TripsModel extends DbEntity  
{  
  
 */\*\*  
 \* @param $pageSize  
 \* @param $page  
 \* @return array  
 \*/* public function getTrips($pageSize, $page): array  
 {  
 $res = $this  
 ->runSQL(<<<'TAG'  
SELECT `trips`.`id`, `PutNum`, C.`model`, C.`GosNum`, D.`firstName`, D.`name`, D.`fatherName`, P.`name` as place,   
T.`content`, `timeStart`, `timeFinish`  
FROM `trips`,   
(SELECT `cars`.`id`, `model`, `GosNum` FROM `cars`) AS C,  
(SELECT `people`.`id`, `firstName`, `name`, `fatherName` FROM `people`) AS D,  
(SELECT `id`, `content` FROM `tasks`) AS T,  
(SELECT `tasks`.`id`, `name` FROM `places`, `tasks` WHERE `places`.`id`=`tasks`.`places\_id`) AS P  
WHERE  
(C.id = `trips`.`cars\_id`)AND(D.`id`=`trips`.`people\_id`)AND(T.`id`=`trips`.`tasks\_id`)AND(P.`id`=`trips`.`tasks\_id`)  
ORDER BY `PutNum`  
TAG  
 );  
 foreach ($res as $key => $row) {  
 $res[$key]['car'] = $row['model'] . ' гос.№' . $row['GosNum'];  
 unset($res[$key]['model']);  
 unset($res[$key]['GosNum']);  
 $res[$key]['driver'] = $row['firstName'] . ' '  
 . *mb\_substr*($row['name'], 0, 1) . '.'  
 . *mb\_substr*($row['fatherName'], 0, 1) . '.';  
 unset($res[$key]['firstName']);  
 unset($res[$key]['name']);  
 unset($res[$key]['fatherName']);  
 $temp = $row['timeStart'];  
 unset($res[$key]['timeStart']);  
 $res[$key]['timeStart'] = $temp;  
 $temp = $row['timeFinish'];  
 unset($res[$key]['timeFinish']);  
 $res[$key]['timeFinish'] = $temp;  
 $temp = $row['place'] . " - " . $row['content'];  
 unset($res[$key]['place']);  
 unset($res[$key]['content']);  
 $res[$key]['task'] = $temp;  
 }  
 return $res;  
 }  
  
 public function getCarsList(): array  
 {  
 $res = [];  
 foreach ($this->runSQL('SELECT `id`, `GosNum`, `model` FROM `cars`') as $row) {  
 $res[$row['id']] = $row['model'] . " " . $row['GosNum'];  
 }  
 return $res;  
 }  
  
 public function getDriversList(): array  
 {  
 $res = [];  
 foreach (  
 $this->runSQL(  
 'SELECT `id`,`firstName`,`name`,`fatherName` FROM `people` WHERE `Driver`=1 ORDER BY `firstName`'  
 ) as $row  
 ) {  
 $res[$row['id']] = $row['firstName']  
 . ' '  
 . *mb\_substr*($row['name'], 0, 1)  
 . '.'  
 . *mb\_substr*($row['fatherName'], 0, 1)  
 . '.';  
 }  
 return $res;  
 }  
  
 public function getTasksList(): array  
 {  
 $res = [];  
 foreach (  
 $this->runSQL(  
 'SELECT `tasks`.`id`, `places`.`name` AS place, `tasks`.`content` AS task FROM `tasks`, `places`   
WHERE `tasks`.`places\_id`=`places`.`id` ORDER BY `places`.`name`'  
 ) as $row  
 ) {  
 $res[$row['id']] = $row['place']  
 . ' - '  
 . $row['task'];  
 }  
 return $res;  
 }  
  
 public function getTripsPlan($pageSize, $page): array  
 {  
 $res = $this  
 ->runSQL(<<<'TAG'  
SELECT `trips`.`id`, `PutNum`, C.`model`, C.`GosNum`, D.`firstName`, D.`name`, D.`fatherName`, P.`name` as place,   
T.`content`, T.`date`, `timeStart`, `timeFinish`  
FROM `trips`,   
(SELECT `cars`.`id`, `model`, `GosNum` FROM `cars`) AS C,  
(SELECT `people`.`id`, `firstName`, `name`, `fatherName` FROM `people`) AS D,  
(SELECT `id`, `date`, `content` FROM `tasks`) AS T,  
(SELECT `tasks`.`id`, `name` FROM `places`, `tasks` WHERE `places`.`id`=`tasks`.`places\_id`) AS P  
WHERE  
(C.id = `trips`.`cars\_id`)AND(D.`id`=`trips`.`people\_id`)AND(T.`id`=`trips`.`tasks\_id`)AND(P.`id`=`trips`.`tasks\_id`)  
ORDER BY T.`date` AND `timeStart`  
TAG  
 );  
 foreach ($res as $key => $row) {  
 $res[$key]['car'] = $row['model'] . ' гос.№' . $row['GosNum'];  
 unset($res[$key]['model']);  
 unset($res[$key]['GosNum']);  
 $res[$key]['driver'] = $row['firstName'] . ' '  
 . *mb\_substr*($row['name'], 0, 1) . '.'  
 . *mb\_substr*($row['fatherName'], 0, 1) . '.';  
 unset($res[$key]['firstName']);  
 unset($res[$key]['name']);  
 unset($res[$key]['fatherName']);  
 $temp = $row['date'] . ' ' . $row['timeStart'];  
 unset($res[$key]['date']);  
 unset($res[$key]['timeStart']);  
 $res[$key]['timeStart'] = $temp;  
 $temp = $row['timeFinish'];  
 unset($res[$key]['timeFinish']);  
 $res[$key]['timeFinish'] = $temp;  
 $temp = $row['place'] . " - " . $row['content'];  
 unset($res[$key]['place']);  
 unset($res[$key]['content']);  
 $res[$key]['task'] = $temp;  
 }  
 return $res;  
 }  
}

**Листинг файла «UserModel.php»**

<?php  
  
*/\*\* @var array $page \*/*namespace Model;  
  
use TexLab\MyDB\DbEntity;  
  
class UserModel extends DbEntity  
{  
 public function getGroups(): array  
 {  
 $res = [];  
 foreach ($this->runSQL("SELECT `id`, `name` FROM `groups` ORDER BY `name`") as $row) {  
 $res[$row['id']] = $row['name'];  
 }  
 return $res;  
 }  
  
 */\*\*  
 \* @param int $pageSize  
 \* @param array $page  
 \* @return array  
 \*/* public function getUsers($pageSize, $page): array  
 {  
  
 return $this  
 ->setSelect(  
 '`users`.`id`, `users`.`login`, `users`.`password`,`users`.`email`, `users`.`name`, `groups`.`cod`'  
 )  
 ->setFrom('`users`, `groups`')  
 ->setWhere('`users`.`group\_id` = `groups`.`id`')  
 ->setOrderBy('`groups`.`name`')  
 ->setPageSize($pageSize)  
 ->getPage($page);  
 }  
}