ВВЕДЕНИЕ

Тема курсового проекта – разработка Автоматизированной информационной справочной системы «Учет деятельности в аптеке».

Актуальность курсового проекта заключается в учете сведений об аптечном пункте, сотрудниках, работающих в аптеке, поставщиках, обслуживающих данную аптеку, таблетках, учет доставок и продаж продукции и автоматизировать работу сотрудников. Автоматизированный учет в аптеке позволит повысить эффективность труда работников, снизить временные затраты по обслуживанию клиентов, вести учет проданных препаратов и медицинских изделий, а также формировать целый ряд печатной и отчетной документации.

Объектом исследования курсового проекта является автоматизация учета деятельности в аптеке.

Предметом исследования представляются методы, алгоритмы и возможности языка программирования PHP, для реализации автоматического программного продукта, взаимодействующего с базой данных.

Целью курсового проекта заключается написание программного обеспечения, которое позволит вести учет полного ассортимента препаратов аптеки, их наличия, поставщиков, отпускной стоимости товаров, возможность разделения таблетированных препаратов, информацию о сотрудниках.

Главными задачами данного курсового проекта можно назначить:

1) исследование предметной области, выделение сущностей, атрибутов и связей между ними;

2) определение требований к разрабатываемому программному продукту;

3) осуществление концептуального и физического проектирования, в ходе которого будет получена модель данных;

4) автоматизация работы аптеки;

5) выполнение функционального и полного тестирования.

Информационно-справочная система – это средство накопления, хранения и предоставления знаний.

Медицинские информационно-справочные системы предназначены для ввода, хранения, поиска и выдачи медицинской информации по запросу пользователя Подобные системы не осуществляют обработку информации, а только обеспечивают быстрый доступ к запрашиваемым сведениям.

1 АНАЛИЗ ЗАДАЧИ

На сегодняшний день существует большое количество различных программ для автоматизации аптек и аптечных сетей. Мы провели анализ разных видов программ по автоматизации учета информации аптек.

В этом анализе мы брали за основу:

1) самые известные из существующих программ для аптек на рынке;
2) самые функциональные программные обеспечения;
3) программы, которые настроены максимально и конкретно для ведения аптечного бизнеса.

Аналогами данной программы для учета информации аптеки можно выделить такие как:

1) Парацельс;
2) Аптека Морион;
3) АНР (на базе 1С-Аптека);
4) Скарб;
5) 1С-Аптека.

Для решения проблемы хранения информация, а также её чтения, было принято решение создать автоматизированную информационно-справочную систему «Учет деятельности в аптеке».

Внедрение в компанию информационно-справочной системы позволит достичь следующих целей:

1) содержать сведения о аптечном пункте, сотрудниках, работающих в аптеке, поставщиках, обслуживающих данную аптеку, таблетках, учет доставок и продаж продукции;
2) каждый ассортимент таблеток указываются: наименование таблеток, название, стоимость товара, разделение таблеток, количество упаковок;
3) ввод с клавиатуры данных о товаре;
4) вывод на экран информации о аптечном пункте;
5) вывод на экран информации о товарах, продающихся в аптечном

пункте, название которого введено с клавиатуры;

6) сортировка информации о товарах по их стоимости, по дате выпуска.

Проектируемая автоматизируемая информационно-справочная система может быть использована во всех аптеках, так как в программном продукте собраны все необходимые функции для учета деятельности в аптеке.

2 ПРОЕКТИРОВАНИЕ ЗАДАЧИ

2.1 Концептуальная модель данных

Концептуальное проектирование начинается с анализа предметной области, включает анализ концептуальных требований и информационных потребностей, выявление информационных объектов и связей между ними, построение концептуальной модели (схемы) данных.

Главными элементами концептуальной модели данных являются объекты и отношения. Объекты представляют собой любой конкретный (реальный) объект в рассматриваемой области.

Объекты в каждый момент времени характеризуются определенным состоянием, которое описывается набором свойств и отношений (или связей) с другими объектами. Характеристика, описывающая какое-либо свойство сущности, которое можно сформулировать и записать, называется атрибутом. Атрибут, который однозначно определяет сущность, называется идентификатором.

Концептуальная модель базы данных это некая наглядная диаграмма, нарисованная в принятых обозначениях и подробно показывающая связь между объектами и их характеристиками. Создается концептуальная модель для дальнейшего проектирования базы данных и перевод ее, например, в реляционную базу данных. На концептуальной модели в визуально удобном виде прописываются связи между объектами данных и их характеристиками.

Для единообразия программирования баз данных введены следующие понятия для концептуальных баз данных:

1) объект или сущность. Это фактическая вещь или объект (для людей) за которой пользователь (заказчик) хочет наблюдать;

2) атрибут – это характеристика объекта, соответствующая его сущности;

3) третье понятие в проектировании концептуальной базы данных это

связь или отношения между объектами.

Создание концептуальной модели данных предметной области является начальной стадией проектирования системы баз данных, в основе которой лежит анализ свойств объектов предметной области и информационных потребностей тех, кто будет эксплуатировать систему.

Эта стадия называется концептуальным проектированием системы, а ее результат – концептуальной моделью предметной области. Объектом моделирования является предметная область будущей системы.

Создание концептуальной модели средствами PowerDesigner выполняются следующим образом:

1) добавление необходимых сущностей на лист диаграмм;
2) определение свойств всех сущностей;
3) установление связей между сущностями и определение их свойств;
4) добавление объекта Заголовок и определение его свойств.

Исходя из сделанных предположений, концептуальная модель для базы данных «Учет деятельности в аптеке» выглядит в соответствии с рисунком 2.1.

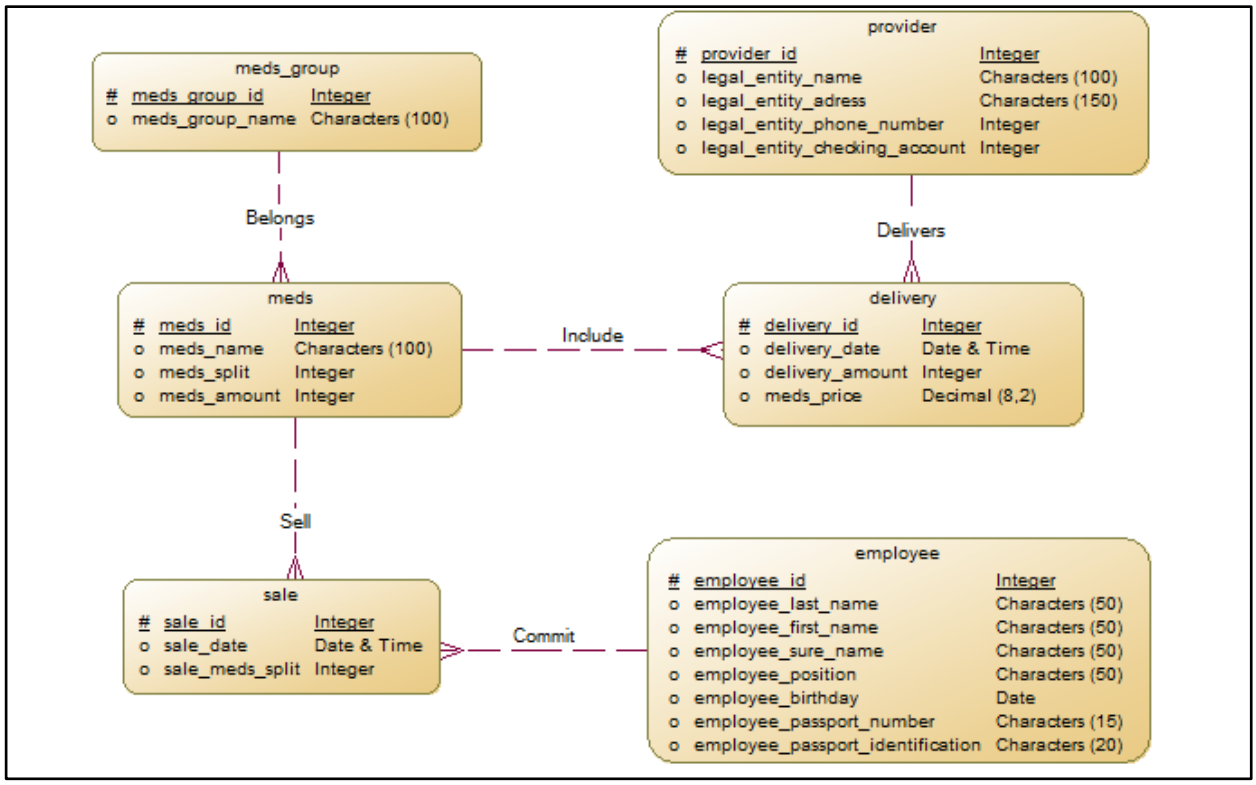

Рисунок 2.1 – Концептуальная модель данных

2.2 Физическая модель базы данных

Физическая модель БД определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Первыми системами хранения и доступа были файловые структуры и системы управления файлами, которые фактически являлись частью операционных систем. СУБД создавала над этими файловыми моделями свою надстройку, которая позволяла организовать всю совокупность файлов таким образом, чтобы она выглядела как единое целое и получала централизованное управление от СУБД. Однако непосредственный доступ осуществлялся на уровне файловых команд, которые СУБД использовала при манипуляции всеми файлами, составляющие хранимые данные одной или нескольких баз данных.

Однако механизмы буферизации и управления файловыми структурами не приспособлены для решения задач собственно СУБД, эти механизмы разрабатывались просто для традиционной обработки файлов, и с ростом объемов хранимых данных они стали неэффективными для использования СУБД. Тогда постепенно произошёл переход от базовых файловых структур к непосредственному управлению размещением данных на внешних носителях самой СУБД.

Целью создания физической модели является обеспечение администратора соответствующей информацией для переноса логической модели данных в СУБД.

Физическая модель содержит всю информацию, необходимую для реализации конкретной БД.

Различают два уровня физической модели:

1) трансформационную модель;
2) модель СУБД.

Трансформационная модель содержит информацию для реализации отдельного проекта, который может быть частью общей ИС и описывать подмножество предметной области. Данная модель позволяет проектировщикам

и администраторам БД лучше представить, какие объекты БД хранятся в словаре данных, и проверить, насколько физическая модель удовлетворяет требованиям к ИС.

Модель СУБД автоматически генерируется из трансформационной модели и является точным отображением системного каталога СУБД.

Физический уровень представления модели зависит от выбранного сервера.

Физическая модель БД определяет способ размещения данных на носителях (устройствах внешней памяти), а также способ и средства организации эффективного доступа к ним. Поскольку СУБД функционирует в составе и под управлением операционной системы, то организация хранения данных и доступа к ним зависит от принципов и методов управления данными операционной системы. Исходя из сделанных предположений, физическая модель для базы данных «Учет деятельности в аптеке» выглядит в соответствии с рисунком 2.2.

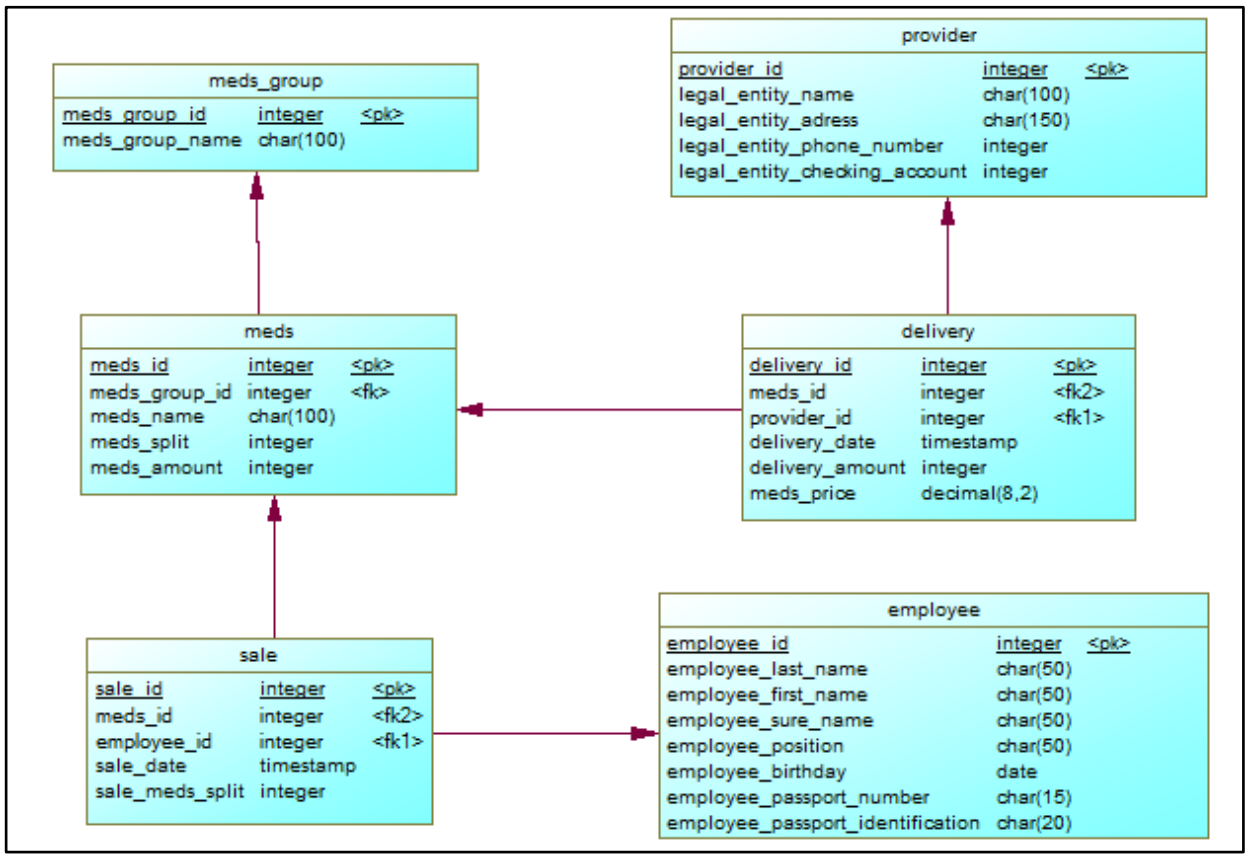

Рисунок 2.2 – Физическая модель базы данных

2.3 Выбор и обоснование среды разработки

База данных – представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины.

СУБД (Система управления базами данных) – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. Система управления базами данных (СУБД) является посредником между базой данных и ее пользователями.

SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. MySQL – свободная реляционная система управления базами данных.

PhpMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL.

Для разработки базы данных вначале требуется изучить предметную область. База данных должна быть сформирована таким образом, чтобы хранить в себе максимально большое количество нужной информации при этом не быть заполненной ненужными данными.

PhpMyAdmin - это бесплатный программный инструмент, написанный на PHP, предназначенный для администрирования MySQL через Интернет. PhpMyAdmin поддерживает широкий спектр операций с MySQL и MariaDB. Часто используемые операции (управление базами данных, таблицами, столбцами, связями, индексами, пользователями, разрешениями и т. д.) Можно выполнять через пользовательский интерфейс, при этом у вас все еще есть возможность напрямую выполнять любую инструкцию SQL.

Программа разрабатывалась в среде быстрой разработки – PhpStorm. PHP

это серверный язык программирования, созданный специально для web-разработки. PHP — один из самых популярных языков программирования, используемых при разработке сайтов и web-приложений. Его преимуществом перед другими языками является возможность внедрения PHP-кода непосредственно в HTML. HTML-файлы являются статичными и отдаются сразу в браузер, который в свою очередь обрабатывает их и выводит на экран содержимое файла с указанной в нем HTML-разметкой. Файлы же с расширением. PHP прежде чем отобразиться в браузере, обрабатываются интерпретатором PHP, расположенном на web-сервере. Интерпретатор преобразует отработанный PHP-код в HTML-код и отдает браузеру чистую HTML-страницу.

Рисунок 2.3 – Среда разработки PhpStorm

Таким образом, при разработке автоматизированной информационно-справочной системы приходится решать три основные задачи:

1) разработка базы данных для хранения информации,
2) разработка графического интерфейса пользователя клиентских

приложений и функции администратора.

PHP: Hypertext Preprocessor (Препроцессор Гипертекста) – это серверный язык программирования, созданный специально для web-разработки. PHP – один из самых популярных языков программирования, используемых при разработке сайтов и web-приложений. Его преимуществом перед другими языками является возможность внедрения PHP-кода непосредственно в HTML.

PHP известен, как серверный язык программирования. Это означает, что он работает на веб сервере. Большинство языков веб-программирования являются серверными языками, но некоторые, например, JavaScript, работают на стороне клиента, это означает, что они работают в веб-браузере.

Серверные языки дают вам больше гибкости, так как вы можете то, что трудно осуществить с помощью JavaScript – например, работа с файлами, базами данных, или работа с изображениями. Нужно сказать, что JavaScript распространился очень быстро в наши дни.

Выполнение кода со стороны сервера является более безопасным способом, чем на стороне клиента, как это делает JavaScript. Поскольку код JavaScript отправляется в веб-браузер, для посетителей сайта легко его просмотреть и редактировать. Даже на одной странице сайта можно с легкостью совмещать PHP и JavaScript. Код находящийся на стороне сервера остаётся веб-сервере и недоступен для посетителей сайта. PHP это инструмент, который находится на веб-сервере и там выполняет PHP скрипты.

Одним из отличительных моментов является применение паттерна MVC. Сам паттерн MVC не является какой-то новой идеей в архитектуре приложений, он появился еще в конце 1970-х годов в компании Xerox как способ организации компонентов в графическом приложение на языке Smalltalk.

Концепция паттерна MVC предполагает разделение приложения на три компонента: модель, контроллер и представление.

Модель (model): описывает используемые в приложении данные, а также логику, которая связана непосредственно с данными, например, логику валидации данных. Как правило, объекты моделей хранятся в базе данных.

В MVC модели представлены двумя основными типами: модели представлений, которые используются представлениями для отображения и передачи данных, и модели домена, которые описывают логику управления данными.

Модель может содержать данные, хранить логику управления этими данными. В то же время модель не должна содержать логику взаимодействия с пользователем и не должна определять механизм обработки запроса. Кроме того, модель не должна содержать логику отображения данных в представлении.

Представление (view): отвечают за визуальную часть или пользовательский интерфейс, нередко html-страница, через который пользователь взаимодействует с приложением. Также представление может содержать логику, связанную с отображением данных. В то же время представление не должно содержать логику обработки запроса пользователя или управления данными.

Контроллер (controller): представляет центральный компонент MVC, который обеспечивает связь между пользователем и приложением, представлением и хранилищем данных. Он содержит логику обработки запроса пользователя. Контроллер получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления, наполненного данными моделей.

PHP представляет собой программное обеспечение с открытым исходным кодом. Это означает, что любой пользователь может получить доступ и работать на PHP. Это помогает быть уверенным, что PHP будет работать в течение длительного времени. PHP можно свободно скачать и использовать. Это является причиной того, что многие хостинг-провайдеры широко используют PHP. Вы обнаружите, что подавляющее большинство веб-хостингом поддерживают работу PHP.

Одной из замечательной функции PHP является то, что вы можете включить блоки PHP кода в HTML-страницы.

Вы можете обособить PHP блоки с помощью специальных символов. Когда веб-сервер получает информацию о странице, все PHP блоки запускаются движком PHP, в то время как, другие части страницы отправляются «как есть» в браузер.

Эта функция позволяет легко делать интерактивными обычные веб-страницы. Отличный инструмент для форм с обратной связью и форм с похожими функциями.

Вы можете использовать PHP для записи практически любого типа веб-приложения или сценария.

Обычные PHP приложения, включают в себя:

1) программное обеспечение для ведения блогов таких, как WordPress;
2) системы электронной коммерции Magento;
3) система управления контентом, в том числе Drupal и Joomla.

Таким образом можно сделать вывод, что выбранная среда разработки позволит создать программное обеспечение, соответствующее всем требованиям, завяленным в анализе задачи.

3 РЕАЛИЗАЦИЯ

3.1 Проектирование базы данных

Управление базами данных – один из самых важных моментов в разработке веб-сайтов приложений и других программных продуктов. Для многих программ необходимо вручную создавать базы данных перед тем, как они смогут быть установлены и настроены.

PhpMyAdmin – это простое, удобное и хорошо документированное решение, которое переведено на множество языков, в том числе и русский. Даже если у пользователя возникают какие-то вопросы по работе с этим приложением, в Интернете можно без труда найти ответы на большинство стандартных вопросов. PhpMyAdmin написан на языке PHP.

Что можно делать с помощью PhpMyAdmin:

1) создавать и корректировать базы данных, таблицы, записи;
2) создавать пользователей;
3) возможность исполнять SQL-команды;
4) система поиска по базе данных.

В общем, есть все необходимые средства, которые необходимы для работы с базой данных.

Нужно отметить, что практически все хостеры устанавливают именно PhpMyAdmin в свои панели управления для работы с базой данных. Это приложение является одним из самых популярных среди других приложений в этом классе.

PhpMyAdmin — это веб-приложение, которое распространяется с открытым кодом, написанное на языке web-программирования PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. PhpMyAdmin для работы с базой данных нужен браузер, который и будет передавать на сервер все команды. В качестве языка работы с БД используется широко известный SQL.

В данном инструменте создали необходимые таблицы, как видно из рисунка 3.1.

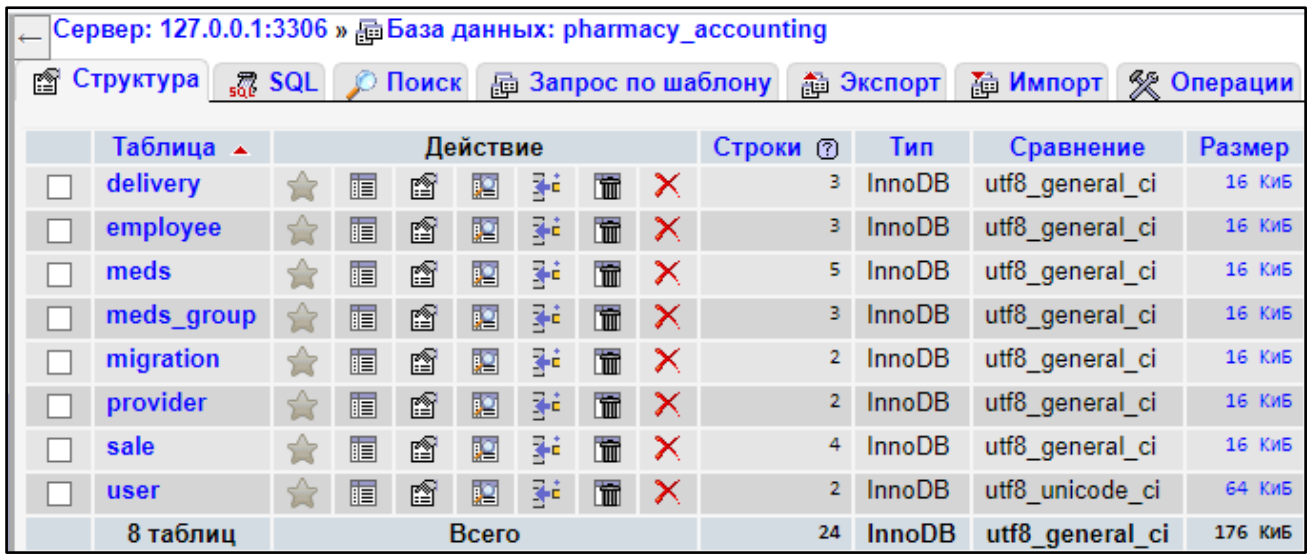

	Таблица	Действие	Строки	Тип	Сравнение	Размер
☐	delivery		3	InnoDB	utf8_general_ci	16 КиБ
☐	employee		3	InnoDB	utf8_general_ci	16 КиБ
☐	meds		5	InnoDB	utf8_general_ci	16 КиБ
☐	meds_group		3	InnoDB	utf8_general_ci	16 КиБ
☐	migration		2	InnoDB	utf8_general_ci	16 КиБ
☐	provider		2	InnoDB	utf8_general_ci	16 КиБ
☐	sale		4	InnoDB	utf8_general_ci	16 КиБ
☐	user		2	InnoDB	utf8_unicode_ci	64 КиБ
	8 таблиц	Всего	24	InnoDB	utf8_general_ci	176 КиБ

Рисунок 3.1 – Таблицы базы данных

SQL – Structured Query Language – структурированный язык запросов, язык управления базами данных для реляционных баз данных. Встроенный SQL в Access позволяет максимально гибко работать с данными и значительно ускоряет доступ к внешним данным.

Автоматизированная информационно-справочная система, имеет 8 сущностей:

1) доставка;
2) работники;
3) медикаменты;
4) тип лекарства;
5) миграция;
6) поставщики;
7) продажа;
8) пользователи.

Первая таблица – «Доставка», хранит в себе данные о дате доставки, коде медикамента, коде поставщика, информацию о количестве доставляемых медикаментов и стоимость за медикаменты. Структура и поля данной таблицы,

отображены на рисунке 3.2.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	delivery_id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	meds_id	int(11)			Нет	*Нет*		
3	provider_id	int(11)			Нет	*Нет*		
4	delivery_date	datetime			Нет	*Нет*		
5	delivery_amount	int(11)			Нет	*Нет*		
6	meds_price	decimal(8,2)			Нет	*Нет*		

Рисунок 3.2 – Поля таблицы «Доставка»

Вторая таблица – «Работники», хранит в себе информацию о сотруднике аптеки, включающая фамилию, имя, отчество, информацию о занимаемой должности, дату рождения работника, а также серийный и идентификационный номера паспорта. Структура и поля данной таблицы представлены на изображении 3.3.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	employee_id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	employee_last_name	varchar(50)	utf8_general_ci		Нет	*Нет*		
3	employee_first_name	varchar(50)	utf8_general_ci		Нет	*Нет*		
4	employee_sure_name	varchar(50)	utf8_general_ci		Нет	*Нет*		
5	employee_position	varchar(50)	utf8_general_ci		Нет	0		
6	employee_birhday	date			Нет	*Нет*		
7	employee_passport_number	varchar(15)	utf8_general_ci		Нет	*Нет*		
8	employee_passport_identification	varchar(20)	utf8_general_ci		Нет	*Нет*		

Рисунок 3.3 – Поля таблицы «Работники»

Третья таблица – «Медикаменты», хранит в себе код медицинской группы, информацию о названии медикамента, информацию о количестве медикаментов и стоимость лекарства. Структура и поля данной таблицы, отображены на рисунке 3.4.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	meds_id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	meds_group_id	int(11)			Нет	*Нет*		
3	meds_name	varchar(100)	utf8_general_ci		Нет	*Нет*		
4	meds_split	int(11)			Нет	*Нет*		
5	meds_amount	int(11)			Нет	*Нет*		

Рисунок 3.4 – Поля таблицы «Медикаменты»

Четвертая таблица – «Тип лекарства», хранит информацию о названии типа лекарства. Структура и поля данной таблицы, отображены на рисунке 3.5.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	meds_group_id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	meds_group_name	varchar(100)	utf8_general_ci		Нет	*Нет*		

Рисунок 3.5 – Поля таблицы «Тип лекарства»

Пятая таблица – «Миграция», хранит в себе информацию о времени применения. Структура и поля данной таблицы, отображены на рисунке 3.6.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	version	varchar(180)	utf8_general_ci		Нет	*Нет*		
2	apply_time	int(11)			Да	*NULL*		

Рисунок 3.6 – Поля таблицы «Миграция»

Шестая таблица – «Поставщики», хранит в себе информацию о поставщике, информацию о местоположении поставщика, информацию о телефоне и читаемом аккаунте. Структура и поля таблицы видны на рисунке 3.7.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	provider_id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	legal_entity_name	varchar(100)	utf8_general_ci		Нет	*Нет*		
3	legal_entity_adress	varchar(150)	utf8_general_ci		Нет	*Нет*		
4	legal_entity_phone_number	int(11)			Нет	*Нет*		
5	legal_entity_checking_account	int(11)			Нет	*Нет*		

Рисунок 3.7 – Поля таблицы «Поставщики»

Седьмая таблица – «Продажа», хранит в себе информацию о кодах медикамента и сотрудника, осуществившего продажу, датах продаж в аптеке и количество медикаментов. Структура и поля таблицы видны из рисунка 3.8.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	sale_id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	meds_id	int(11)			Нет	*Нет*		
3	employee_id	int(11)			Нет	*Нет*		
4	sale_date	datetime			Нет	*Нет*		
5	sale_meds_split	int(11)			Нет	*Нет*		

Рисунок 3.8 – Поля таблицы «Продажа»

Восьмая таблица – «Пользователи», хранит в себе информацию о

пользователе, и полную информацию для входа в программный продукт. Структура и поля данной таблицы, отображены на рисунке 3.9.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	id	int(11)			Нет	*Нет*		AUTO_INCREMENT
2	username	varchar(255)	utf8_unicode_ci		Нет	*Нет*		
3	auth_key	varchar(32)	utf8_unicode_ci		Нет	*Нет*		
4	password_hash	varchar(255)	utf8_unicode_ci		Нет	*Нет*		
5	password_reset_token	varchar(255)	utf8_unicode_ci		Да	*NULL*		
6	email	varchar(255)	utf8_unicode_ci		Нет	*Нет*		
7	status	smallint(6)			Нет	10		
8	created_at	int(11)			Нет	*Нет*		
9	updated_at	int(11)			Нет	*Нет*		

Рисунок 3.9 – Поля таблицы «Пользователи»

3.2 Логическая и физическая организация данных. Функции приложения

На основе концептуальной и физической моделях была построена база данных «Pharmacy_accounting», содержащая таблицы: доставка, работники, медикаменты, тип лекарства, миграция, поставщики, продажа, пользователи.

Таблица 3.1 – «Доставка»

Поле	Тип	Назначение
delivery_id	integer	Ключевое поле
meds_id	integer	Внешний ключ таблицы «Медикаменты»
provaider_id	integer	Внешний ключ таблицы «Поставщики»
delivery_date	datetime	Дата поставки
delivery_amount	integer	Количество упаковок
meds_price	decimal	Цены препарата

Таблица 3.2 – «Работники»

Поле	Тип	Назначение
employee_id	integer	Ключевое поле

Окончание таблицы 3.2 – «Работники»

employee_last_name	varchar	Фамилия сотрудника
employee_first_name	varchar	Имя сотрудника
employee_sure_name	varchar	Отчество сотрудника
employee__position	varchar	Должность сотрудника
employee_birthday	date	Дата рождения сотрудника
employee_passport_number	varchar	Паспортный номер сотрудника
employee_passport_identification	varchar	Идентификационный номер сотрудника

Таблица 3.3 – «Медикаменты»

Поле	Тип	Назначение
meds_id	integer	Ключевое поле
meds_group_id	integer	Внешний ключ таблицы «Тип лекарств»
meds_name	varchar	Наименование лекарства
meds_split	integer	Разделение лекарства
meds_amount	integer	Количество лекарства

Таблица 3.4 – «Тип лекарства»

Поле	Тип	Назначение
meds_group_id	integer	Ключевое поле
meds_group_name	varchar	Название медицинской группы препаратов

Таблица 3.5 – «Миграция»

Поле	Тип	Назначение
version	varchar	Ключевое поле
apply_time	integer	Время миграции

Таблица 3.6 – «Поставщики»

Поле	Тип	Назначение
provaider_id	integer	Ключевое поле
legal_entity_name	varchar	Наименование юридического лица поставщика
legal_entity_adress	varchar	Адрес юридического лица поставщика
legal_entity_phone_number	integer	Телефонный номер юридического лица поставщика
legal_entity_checking_account	integer	Расчетный счет юридического лица поставщика

Таблица 3.7 – «Продажа»

Поле	Тип	Назначение
sale_id	integer	Ключевое поле
meds_id	integer	Внешний ключ таблицы «Медикаментов»
employee_id	integer	Внешний ключ таблицы «Работники»
sale_date	datetime	Дата продажи
sale_meds_split	integer	Проданные разделения лекарств

Таблица 3.8 – «Пользователи»

Поле	Тип	Назначение
id	integer	Ключевое поле
username	varchar	Логин

Окончание таблицы 3.8 – «Пользователи»

auth_key	varchar	Ключ аутентификации
password_hash	varchar	Хэш пароля
password_reset_token	varchar	Токен сброса пароля
email	varchar	Электронный адрес
status	smallint	Статус
created_at	integer	Созданный номер
updated_at	integer	Редактируемый номер

3.3 Функции и элементы управления

Построенный поверх DAO, построитель запросов позволяет конструировать SQL выражения в программируемом и независимом от СУБД виде. В сравнении с написанием чистого SQL выражения, использование построителя помогает вам писать более читаемый связанный с SQL код и генерировать более безопасные SQL выражения.

Использование построителя запросов, как правило, включает два этапа:

Создание объекта yii\db\Query представляющего различные части (такие как SELECT, FROM) SQL выражения SELECT.

Выполнить запрос методом yii\db\Query (таким как all()) для извлечения данных из базы данных.

Создав объект yii\db\Query возможно вызывать различные методы для создания различных частей SQL выражения. Имена методов напоминают ключевые слова SQL, используемые в соответствующих частях SQL запроса. Например, чтобы указать FROM часть запроса, вам нужно вызвать метод from(). Все методы построителя запросов возвращают свой объект, который позволяет объединять несколько вызовов в цепочку.

Главными физическими функциями программы являются:

1) добавление;
2) удаление;

3) редактирование;

4) поиск;

5) выборка.

Запрос на добавление данных в таблице «Доставка», показано на рисунке 3.10.

```
public function actionCreate()
{
    $model = new Delivery();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->delivery_id]);
    }

    return $this->render('create', [
        'model' => $model,
    ]);
}
```

Рисунок 3.10 – Запрос на добавление данных

Запрос на редактирование данных в таблице «Доставка», показано на рисунке 3.11.

```
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->delivery_id]);
    }

    return $this->render('update', [
        'model' => $model,
    ]);
}
```

Рисунок 3.11 – Запрос на редактирование данных

Запрос на удаление данных из таблицы «Доставка», показано на рисунке 3.12.

```
public function actionDelete($id)
{
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}
```

Рисунок 3.12 – Запрос на удаление данных

Запрос на выборку данных из таблицы «Доставка», показано на рисунке 3.13.

```
public function actionIndex()
{
    $searchModel = new DeliverySearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}
```

Рисунок 3.13 – Запрос на выборку данных

Запрос на выборку данных из таблицы «Доставка», показано на рисунке 3.14.

```
public function actionView($id)
{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}
```

Рисунок 3.14 – Запрос на выборку данных

Запрос на поиск данных из таблицы «Доставка», показано на рисунке 3.15.

```
protected function findModel($id)
{
    if (($model = Delivery::findOne($id)) !== null) {
        return $model;
    }

    throw new NotFoundHttpException('The requested page does not exist.');
}
```

Рисунок 3.15 – Запрос на поиск данных

Для входа в проект необходимо запустить «OpenServer», и во вкладке «Дополнительно» выбрать «Консоль» и прописать следующие команды, как показано на рисунке 3.16.

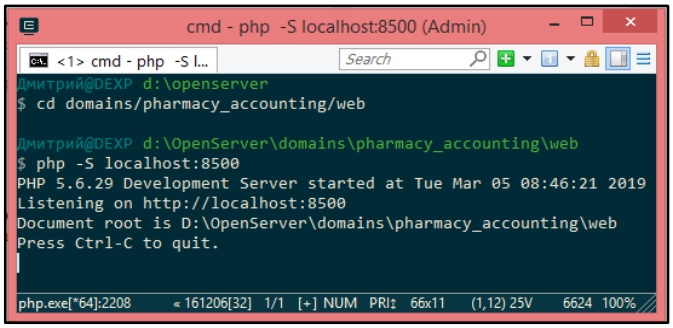

Рисунок 3.16 – Вход в проект

Главное меню курсового проекта содержит 5 форм:

1) главная;
2) о программе;
3) контакты;
4) регистрация;
5) вход.

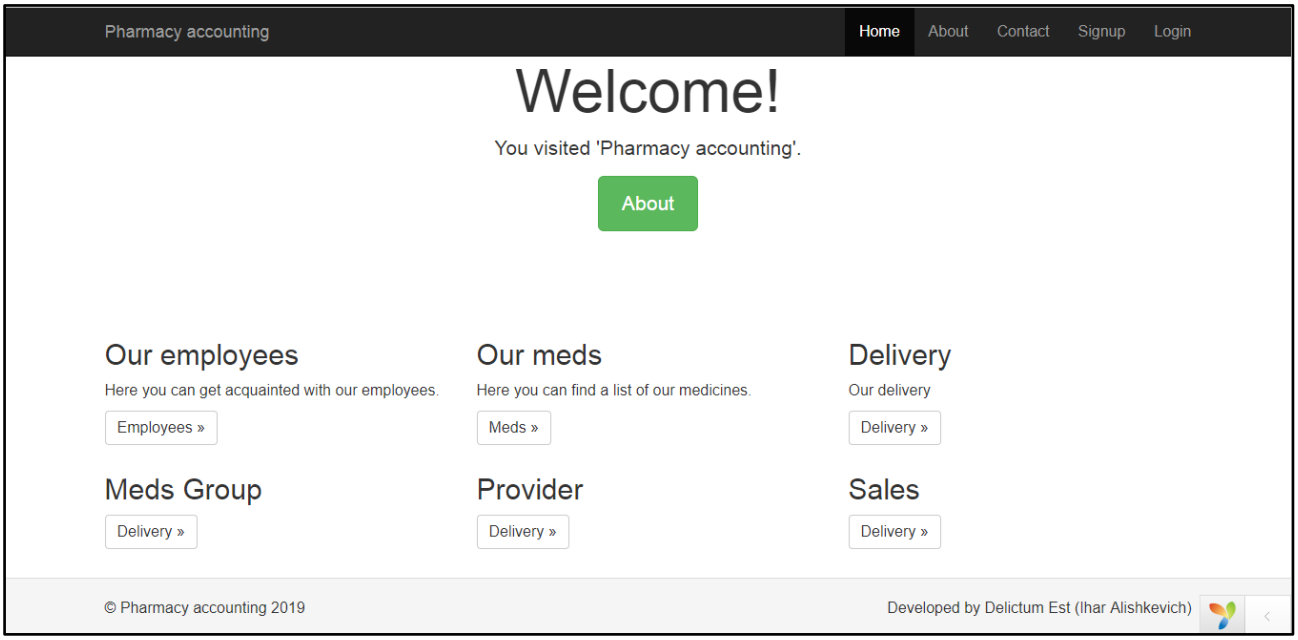

Рисунок 3.17 – Главное окно программы

Таблица 3.9 – Подробное описание форм приложения

Форма	Описание
Home	Главная форма приложения
About	Форма отображающая информацию о проекте

Окончание таблицы 3.9 – Подробное описание форм приложения

Contact	Форма отображающая контакты
SignUp	Форма регистрации
Login	Форма входа
Our employees	Форма доставки
Our Meds	Форма медикаментов
Delivery	Форма миграции
Meds Group	Форма типов лекарств
Provider	Форма поставщиков
Sales	Форма цен

Проделана работа по создания курсового проекта автоматизированной информационной-справочной системы «Учета деятельности в аптеке», проработаны информационные источники для создания курсового проекта.

4 ТЕСТИРОВАНИЕ ПО

4.1 Функциональное тестирование

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям.

Таблица 4.1 – Функциональное тестирование

№	Функциональный элемент	Ожидаемый результат	Полученный результат
1	Кнопка «Create employee» на форме «Our employee»	Добавляет данные о сотрудниках в таблицу «Employee», а именно «Employee Last Name», «Employee First Name», «Employee Sure Name», «Employee Position», «Employee Birhday», « Employee Passport Number»	Успешно добавлено
2	Кнопка «Delete» на форме «Employee»	Удаляет запись в таблице «Employee» по ключевому полю	Успешно удалено
3	Кнопка «Update» на форме «Employee»	Изменяет данные о сотрудниках в таблице «Employee», а именно «Employee Last Name», «Employee First Name», «Employee Sure Name», «Employee Position», «Employee Birhday», « Employee Passport Number»	Успешно изменено
4	Кнопка «Create Meds» на форме «Meds»	Добавляет данные о медикаментах в таблицу «Meds», а именно выбираем из выпадающего списка «Meds Group ID», «Meds Name», «Meds Split», «Meds Amount»	Успешно добавлено

Продолжение таблицы 4.1

5	Кнопка «Delete» на форме «Meds»	Удаляет записи в таблице «Meds» по ключевому полю.	Успешно удалено
6	Кнопка «Update» на форме «Meds»	Изменяет данные о медикаментах в таблице «Meds», а именно «Meds Group ID», «Meds Name», «Meds Split», «Meds Amount».	Успешно изменено
7	Кнопка «Create Delivery» на форме «Deliveries»	Добавляет данные о поставке в таблицу «Deliveries», а именно выбираем из выпадающего списка «Meds ID», «Provaider ID», «Delivery Date», «Delivery Amount», «Meds Price».	Успешно добавлено
8	Кнопка «Delete» на форме «Deliveries»	Удаляет записи в таблице «Deliveries» по ключевому полю	Успешно удалено
9	Кнопка «Update» на форме «Deliveries»	Изменяет данные о поставке в таблице «Deliveries», а именно «Meds ID», «Provaider ID», «Delivery Amount», «Meds Price».	Успешно изменено
10	Кнопка «Create Meds Group» на форме «Meds Group»	Добавляет данные о типах лекарств в таблицу «Meds Group», а именно «Meds Group Name»	Успешно добавлено
11	Кнопка «Delete» на форме «Meds Group»	Удаляет записи в таблице «Meds Group» по ключевому полю.	Успешно удалено
12	Кнопка «Update» на форме «Meds Group»	Изменяет данные о типах лекарств в таблице «Meds Group», а именно «Meds Group Name»	Успешно изменено

Окончание таблицы 4.1

13	Кнопка «Create Provider» на форме «Providers»	Добавляет данные о поставщиках в таблицу «Providers», а именно «Legal Entity Name», «Legal Entity Adress», «Legal Entity Phone» Number», «Legal Entity Checking Account»	Успешно добавлено
14	Кнопка «Update» на форме «Providers»	Изменяет данные о поставщиках в таблице «Providers», а именно «Legal Entity Name», «Legal Entity Adress», «Legal Entity Phone» Number», «Legal Entity Checking Account»	Успешно изменено
15	Кнопка «Delete» на форме «Providers»	Удаляет записи в таблице «Providers» по ключевому полю.	Успешно удалено
16	Кнопка «Create Sale» на форме «Sales»	Добавляет данные о стоимости в таблицу «Sales», а именно из выпадающего списка «Meds ID», «Employee ID», «Sale Date», «Sale Meds Split»	Успешно добавлено
17	Кнопка «Delete» на форме «Sales»	Удаляет записи в таблице «Sales» по ключевому полю.	Успешно удалено
18	Кнопка «Update» на форме «Sales»	Изменяет данные о стоимости в таблице «Sales», а именно «Meds ID», «Sale Date», «Sale Meds Split»	Успешно изменено

После проведения функционального тестирования, все ошибки были исправлены.

4.2 Полное тестирование

Добавим нового сотрудника путем заполнения следующих полей:

«Employee Last Name», «Employee First Name», «Employee Sure Name», «Employee Position», «Employee Birhday», « Employee Passport Number», « Employee Passport Identification».

На данной форме добавим следующего сотрудника: «Петров», «Петр», «Петрович», «Продавец», «1995-07.14», «КН», «463563563».

Форма добавления «Employee» отображена на рисунке 4.1.

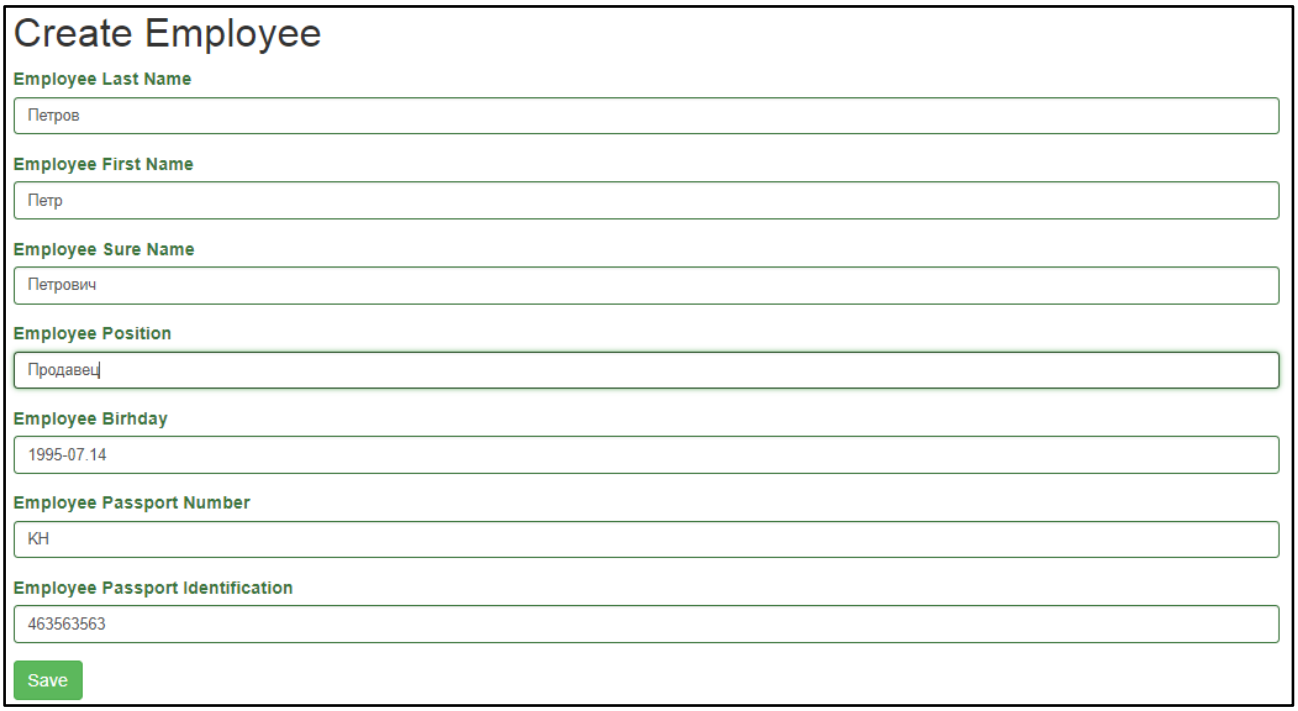

Рисунок 4.1 – Форма «Employee»

Затем просмотрим заполненые данные на рисунке 4.2.

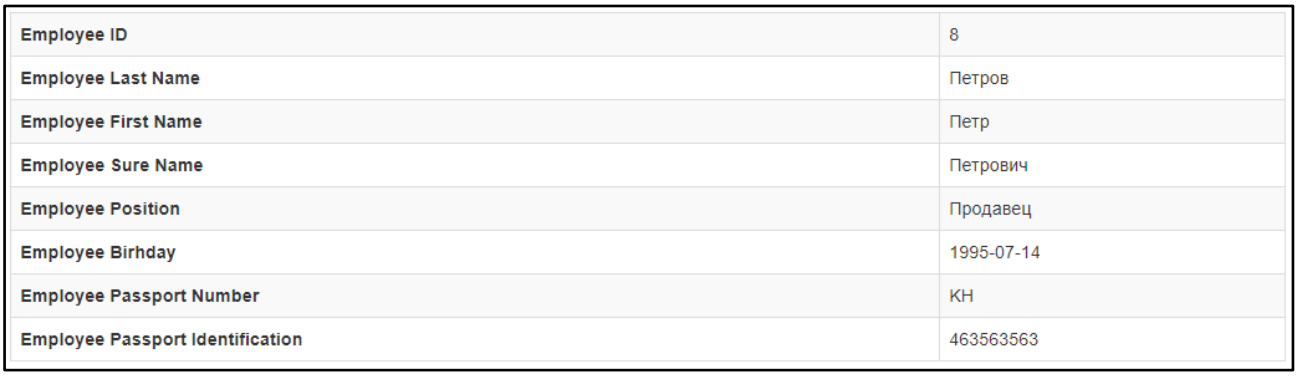

Рисунок 4.2 – Добавленые данные о сотруднике

Теперь добавим данные о новой поставке в форме «Deliveries». На появившейся форме заполнили следующие поля: выбрали из выпадающего

списка «Meds ID» и «Provider ID», и заполнили поля данными «Delivery Date», «Delivery Amount», «Meds Price». На данной форме добавим следующие данные: «Velaxin», «provider name», «2019-02-28 10:30:12», «555», «4656», как показано на рисунке 4.3.

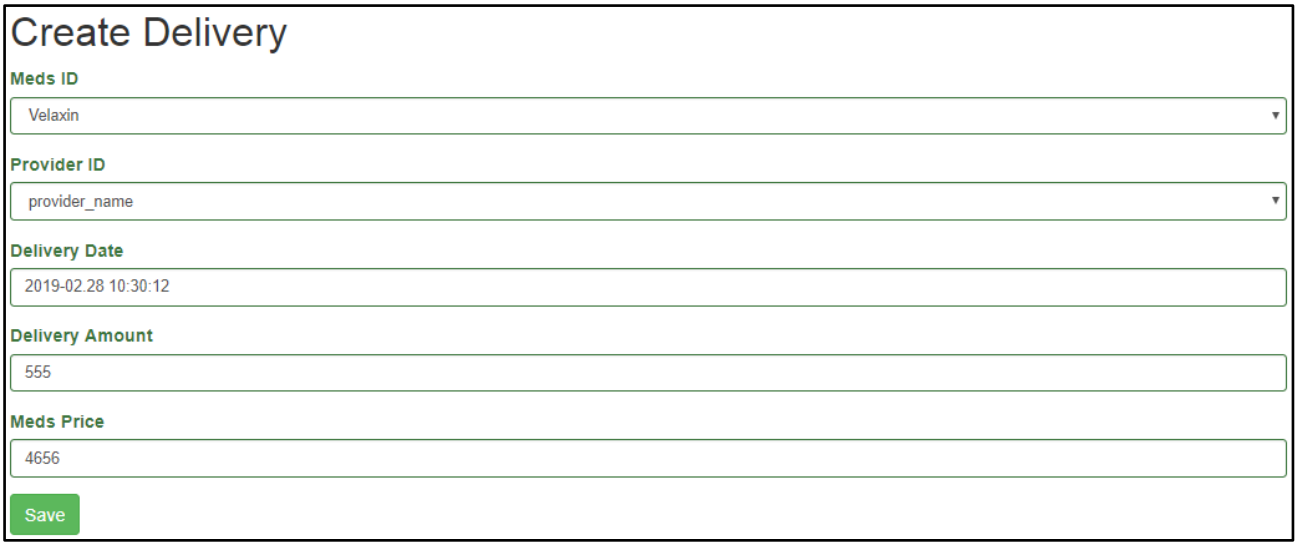

Рисунок 4.3 – Заполнение данных о перевозке

Затем нажали на кнопку «Save», заполненные данные продемонстрированы на рисунке 4.4.

Delivery ID	4
Meds ID	7
Provider ID	2
Delivery Date	2019-02-28 10:30:12
Delivery Amount	555
Meds Price	4656.00

Рисунок 4.4 – Добавленые данные о перевозке

Теперь добавим данные о медикаментах в форме «Meds». На появившейся форме заполнили следующие поля: выбрали из выпадающего списка «Meds Group ID», и заполнили поля данными «Meds Name», «Meds Split», «Meds Amount». На данной форме добавим следующие данные: «Painkiller», «Степан», «556», «228», как показано на рисунке 4.5.

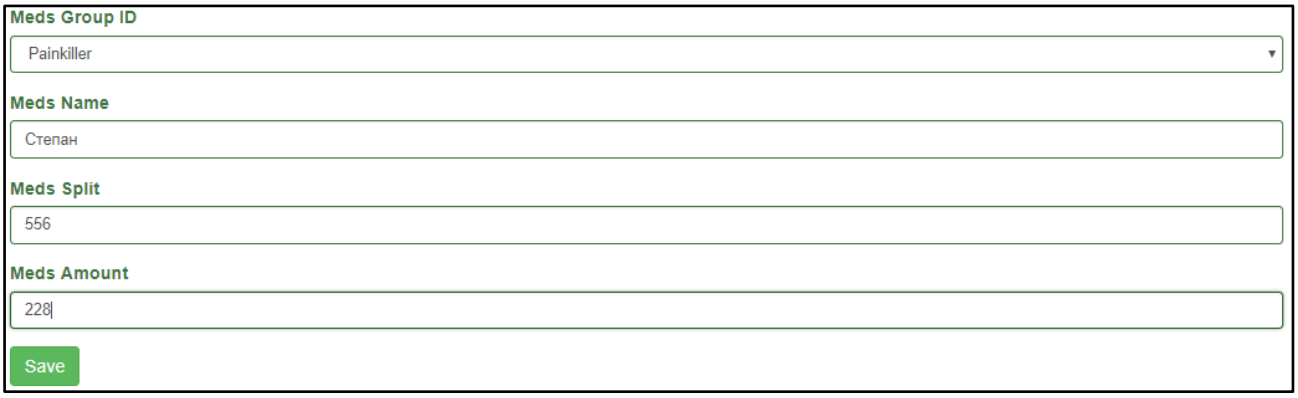

Рисунок 4.5 – Добавление данных о медикаментах

Затем нажали на кнопку «Save», заполненные данные продемонстрированы на рисунке 4.6.

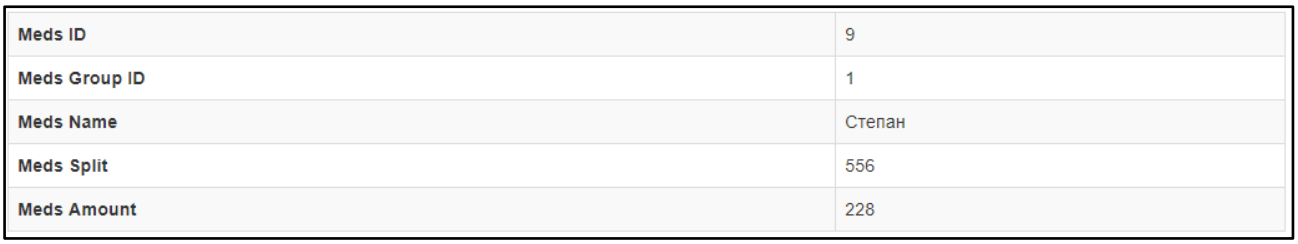

Рисунок 4.6 – Добавление данных о медикаментах

Таким образом было проведено полное тестирование программного продукта.

ЗАКЛЮЧЕНИЕ

Задачей курсового проекта являлась разработка программы – Автоматизированной информационная справочная система «Учет деятельности в аптеке».

В ходе выполнения задания практики, было разработано программное средство с использованием языка программирования «PHP», а также, с помощью языка структурированных запросов «SQL» на основе системы управления базами данных «MySQL».

В разделе «Анализ задачи» изучена предметная область, в ходе которой была установлена среда использования, сравнение с существующими аналогами, что позволило грамотно приступить к осуществлению последующих шагов

В проектировании задачи были разработаны концептуальная и физическая модели данных в соответствии с созданным программным продуктом, на которых была отображена необходимая информация в виде сущностей, их атрибутов и связей между ними. При проектировании использовалось программное обеспечение «Power designer».

Дальнейшим шагом был сделан выбор в пользу среды разработки PhpStorm.

В ходе написания курсового проекта были разработаны следующие функции:

1) добавление, удаление и редактирование всех форм программы;
2) просмотр сотрудников;
3) просмотр медикаментов;
4) просмотр поставщиков;
5) просмотр доставок;
6) сортировка и выборка во всех формах курсового проекта;
7) регистрация и авторизация.

Для реализации данного курсового проекта была выбрана среда

разработки PhpStorm. Она была выбрана из-за своей простоты использования, а также за интуитивно понятный интерфейс, в том числе быстродействия.

В разделе реализации была реализована база данных, а также осуществлены задумываемые функции программы. Результатом данного раздела стал готовый программный продукт.

После вся программа неоднократно тестировалась на правильность получаемых результатов, все возникающие ошибки были устранены в ходе работы. Тестирование происходило с помощью метода «Чёрного ящика». После проведенного тестирования можно установить, что программа удовлетворяет всем поставленным перед ней требованиям и является готовым программным средством.

Основное внимание уделено изучению способов проектирования приложений, использованию паттернов, объектно-ориентированному и системному программированию.

Разработанное обеспечение быстро и безошибочно справляется с поставленной задачей хранения и обработки информации. В удобном интерфейсе воплощены все необходимые для данной работы возможности.

Основные модификации по дальнейшему улучшению заключаются в:

1) реинжиниринге;
2) интернационализации и локализации;
3) портировании и миграции программного обеспечения;
4) расширение до сетевого использования аптек.

Достоинством проекта является простота и интуитивность программного средства.

Исходя из вышесказанного можно утверждать о выполненной цели курсового проекта. Все поставленные задачи реализованы в соответствии с заданным условием.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Скляр Д. Изучаем PHP. Руководство по созданию интерактивных сайтов / Скляр Д. – «Альфа-книга», 2017

2 Никсон, Р. Создание динамических веб-сайтов с помощью PHP, MySQL, JavaScript, CSS и HTML5. 3-е издание / Никсон Р. – «O'relly», 2019

3 Зандстра М. PHP. Объекты, шаблоны и методики программирования / Зандстра М – «Ozon», 2011

4 Эванс Э. Предметно-ориентированное проектирование (DDD). Структуризация сложных программных систем / Эванс Э. – «Domain driven», 2010

5 Хопкинс К. PHP. Быстрый старт / Хопкинс К. – «Эксмо», 2014

6 Сафронов М. Разработка веб-приложений в Yii 2 / Сафронов М. – «ДМК», 2015

7 Локхарт Д. Современный PHP. Новые возможности и передовой опыт / Локхарт Д. – «ДМК», 2016

8 Колисниченко Д. PHP и MySQL. Разработка Web-приложений / Колисниченко Д. – «BHV», 2017

9 Симдянов И. PHP 7. В подлиннике / Симдянов И. – «BHV», 2016

10 Шварц Б., Зайцев П., Ткаченко В. и др. - MySQL. Оптимизация производительности (2-е издание)

11 Куликов С. Тестирование программного обеспечения. Базовый курс / С. Куликов – Четыре четверти, 2017

ПРИЛОЖЕНИЕ А

Таблица стилей

```
html,
body {
    height: 100%;
    font-size: 16px;
}

.btn {
    font-size: 16px;
    size: 50px;
}

.wrap {
    min-height: 100%;
    height: auto;
    margin: 0 auto -60px;
    padding: 0 0 60px;
}

.wrap > .container {
    padding: 70px 15px 20px;
}

.footer {
    height: 60px;
    background-color: #f5f5f5;
    border-top: 1px solid #ddd;
    padding-top: 20px;
}

.jumbotron {
    text-align: center;
    background-color: transparent;
}

.jumbotron .btn {
    font-size: 21px;
    padding: 14px 24px;
}

.not-set {
    color: #c55;
    font-style: italic;
}

/* add sorting icons to gridview sort links */
a.asc:after, a.desc:after {
    position: relative;
    top: 1px;
    display: inline-block;
    font-family: 'Glyphicons Halflings';
    font-style: normal;
    font-weight: normal;
    line-height: 1;
    padding-left: 5px;
}

a.asc:after {
    content: /*"\e113"*/ "\e151";
```

```
}

a.desc:after {
    content: /*"\e114"*/ "\e152";
}

.sort-numerical a.asc:after {
    content: "\e153";
}

.sort-numerical a.desc:after {
    content: "\e154";
}

.sort-ordinal a.asc:after {
    content: "\e155";
}

.sort-ordinal a.desc:after {
    content: "\e156";
}

.grid-view th {
    white-space: nowrap;
}

.hint-block {
    display: block;
    margin-top: 5px;
    color: #999;
}

.error-summary {
    color: #a94442;
    background: #fdf7f7;
    border-left: 3px solid #eed3d7;
    padding: 10px 20px;
    margin: 0 0 15px 0;
}

/* align the logout "link" (button in form) of the navbar */
.nav li > form > button.logout {
    padding: 15px;
    border: none;
}

@media(max-width:767px) {
    .nav li > form > button.logout {
        display:block;
        text-align: left;
        width: 100%;
        padding: 10px 15px;
    }
}

.nav > li > form > button.logout:focus,
.nav > li > form > button.logout:hover {
    text-decoration: none;
}

.nav > li > form > button.logout:focus {
    outline: none;
}
```

Конфигурационный файл подключения к базе данных

```php
<?php

return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=pharmacy_accounting',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',

    // Schema cache options (for production environment)
    //'enableSchemaCache' => true,
    //'schemaCacheDuration' => 60,
    //'schemaCache' => 'cache',
];
```

Основной конфигурационный файл

```php
<?php

$params = require __DIR__ . '/params.php';
$db = require __DIR__ . '/db.php';

$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'aliases' => [
        '@bower' => '@vendor/bower-asset',
        '@npm'   => '@vendor/npm-asset',
    ],
    'components' => [
        'request' => [
            // !!! insert a secret key in the following (if it is empty) - this
is required by cookie validation
            'cookieValidationKey' => 'xoUGteU-rTiWvY5eDY4GcJ2zSX2o2AwP',
        ],
        'cache' => [
            'class' => 'yii\caching\FileCache',
        ],
        'user' => [
            'identityClass' => 'app\models\User',
            'enableAutoLogin' => true,
        ],
        'errorHandler' => [
            'errorAction' => 'site/error',
        ],
        'mailer' => [
            'class' => 'yii\swiftmailer\Mailer',
            // send all mails to a file by default. You have to set
            // 'useFileTransport' to false and configure a transport
            // for the mailer to send real emails.
            'useFileTransport' => true,
        ],
        'log' => [
            'traceLevel' => YII_DEBUG ? 3 : 0,
            'targets' => [
                [
                    'class' => 'yii\log\FileTarget',
                    'levels' => ['error', 'warning'],
                ],
            ],
```

```
        ],
        'db' => $db,
        /*
        'urlManager' => [
            'enablePrettyUrl' => true,
            'showScriptName' => false,
            'rules' => [
            ],
        ],
        */
    ],
    'params' => $params,
];

if (YII_ENV_DEV) {
    // configuration adjustments for 'dev' environment
    $config['bootstrap'][] = 'debug';
    $config['modules']['debug'] = [
        'class' => 'yii\debug\Module',
        // uncomment the following to add your IP if you are not connecting from
localhost.
        //'allowedIPs' => ['127.0.0.1', '::1'],
    ];

    $config['bootstrap'][] = 'gii';
    $config['modules']['gii'] = [
        'class' => 'yii\gii\Module',
        // uncomment the following to add your IP if you are not connecting from
localhost.
        //'allowedIPs' => ['127.0.0.1', '::1'],
    ];
}

return $config;
```

Зависимости

```
{
    "name": "yiisoft/yii2-app-basic",
    "description": "Yii 2 Basic Project Template",
    "keywords": ["yii2", "framework", "basic", "project template"],
    "homepage": "http://www.yiiframework.com/",
    "type": "project",
    "license": "BSD-3-Clause",
    "support": {
        "issues": "https://github.com/yiisoft/yii2/issues?state=open",
        "forum": "http://www.yiiframework.com/forum/",
        "wiki": "http://www.yiiframework.com/wiki/",
        "irc": "irc://irc.freenode.net/yii",
        "source": "https://github.com/yiisoft/yii2"
    },
    "minimum-stability": "stable",
    "require": {
        "php": ">=5.4.0",
        "yiisoft/yii2": "~2.0.14",
        "yiisoft/yii2-bootstrap": "~2.0.0",
        "yiisoft/yii2-swiftmailer": "~2.0.0",
        "kartik-v/yii2-widget-datetimepicker": "^1.4"
    },
    "require-dev": {
        "yiisoft/yii2-debug": "~2.0.0",
        "yiisoft/yii2-gii": "~2.0.0",
        "yiisoft/yii2-faker": "~2.0.0",
```

```
        "codeception/base": "^2.2.3",
        "codeception/verify": "~0.3.1",
        "codeception/specify": "~0.4.3"
    },
    "config": {
        "process-timeout": 1800,
        "fxp-asset": {
            "enabled": false
        }
    },
    "scripts": {
        "post-install-cmd": [
            "yii\\composer\\Installer::postInstall"
        ],
        "post-create-project-cmd": [
            "yii\\composer\\Installer::postCreateProject",
            "yii\\composer\\Installer::postInstall"
        ]
    },
    "extra": {
        "yii\\composer\\Installer::postCreateProject": {
            "setPermission": [
                {
                    "runtime": "0777",
                    "web/assets": "0777",
                    "yii": "0755"
                }
            ]
        },
        "yii\\composer\\Installer::postInstall": {
            "generateCookieValidationKey": [
                "config/web.php"
            ]
        }
    },
    "repositories": [
        {
            "type": "composer",
            "url": "https://asset-packagist.org"
        }
    ]
}
```

Основной слой отображения

```
<?php

/* @var $this \yii\web\View */
/* @var $content string */

use app\widgets\Alert;
use yii\helpers\Html;
use yii\bootstrap\Nav;
use yii\bootstrap\NavBar;
use yii\widgets\Breadcrumbs;
use app\assets\AppAsset;

AppAsset::register($this);
?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>">
<head>
    <meta charset="<?= Yii::$app->charset ?>">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <?= Html::csrfMetaTags() ?>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
</head>
<body>
<?php $this->beginBody() ?>

<div class="wrap">
    <?php
    NavBar::begin([
        'brandLabel' => 'Pharmacy accounting',
        'brandUrl' => Yii::$app->homeUrl,
        'options' => [
            'class' => 'navbar-inverse navbar-fixed-top',
        ],
    ]);

    $menuItems = [
        ['label' => 'Home', 'url' => ['/site/index']],
        ['label' => 'About', 'url' => ['/site/about']],
        ['label' => 'Contact', 'url' => ['/site/contact']],
    ];

    if (Yii::$app->user->isGuest) {
        $menuItems[] = ['label' => 'Signup', 'url' => ['/site/signup']];
        $menuItems[] = ['label' => 'Login', 'url' => ['/site/login']];
    } else {
        $menuItems[] = '<li>'
            . Html::beginForm(['/site/logout'], 'post')
            . Html::submitButton(
                'Logout (' . Yii::$app->user->identity->username . ')',
                ['class' => 'btn btn-link logout']
            )
            . Html::endForm()
            . '</li>';
    }

    echo Nav::widget([
        'options' => ['class' => 'navbar-nav navbar-right'],
        'items' => $menuItems,
    ]);

    NavBar::end();
    ?>

    <div class="container">
        <?= Breadcrumbs::widget([
            'links' => isset($this->params['breadcrumbs']) ? $this-
>params['breadcrumbs'] : [],
        ]) ?>
        <?= Alert::widget() ?>
        <?= $content ?>
    </div>
</div>

<footer class="footer">
    <div class="container">
        <p class="pull-left">&copy; Pharmacy accounting <?= date('Y') ?></p>

        <p class="pull-right">Developed by Delictum Est (Ihar Alishkevich)</p>
    </div>
</footer>
```

```php
<?php $this->endBody() ?>
</body>
</html>
<?php $this->endPage() ?>
```

Основной контроллер

```php
<?php

namespace app\controllers;

use app\models\PasswordResetRequestForm;
use app\models\ResetPasswordForm;
use app\models\SignupForm;
use app\models\User;
use Yii;
use yii\base\InvalidParamException;
use yii\filters\AccessControl;
use yii\web\BadRequestHttpException;
use yii\web\Controller;
use yii\web\Response;
use yii\filters\VerbFilter;
use app\models\LoginForm;
use app\models\ContactForm;

class SiteController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['logout'],
                'rules' => [
                    [
                        'actions' => ['logout'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'logout' => ['post'],
                ],
            ],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function actions()
    {
        return [
            'error' => [
                'class' => 'yii\web\ErrorAction',
            ],
            'captcha' => [
```

```
                'class' => 'yii\captcha\CaptchaAction',
                'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
            ],
        ];
    }

    /**
     * Displays homepage.
     *
     * @return string
     */
    public function actionIndex()
    {
        return $this->render('index');
    }

    /**
     * Login action.
     *
     * @return Response|string
     */
    public function actionLogin()
    {
        if (!Yii::$app->user->isGuest) {
            return $this->goHome();
        }

        $model = new LoginForm();
        if ($model->load(Yii::$app->request->post()) && $model->login()) {
            return $this->goBack();
        }

        $model->password = '';
        return $this->render('login', [
            'model' => $model,
        ]);
    }

    /**
     * Logout action.
     *
     * @return Response
     */
    public function actionLogout()
    {
        Yii::$app->user->logout();

        return $this->goHome();
    }

    /**
     * Displays contact page.
     *
     * @return Response|string
     */
    public function actionContact()
    {
        $model = new ContactForm();
        if ($model->load(Yii::$app->request->post()) && $model-
>contact(Yii::$app->params['adminEmail'])) {
            Yii::$app->session->setFlash('contactFormSubmitted');

            return $this->refresh();
        }
```

```
        return $this->render('contact', [
            'model' => $model,
        ]);
    }

    /**
     * Displays about page.
     *
     * @return string
     */
    public function actionAbout()
    {
        return $this->render('about');
    }

    public function actionAddAdmin() {
        $model = User::find()->where(['username' => 'admin'])->one();
        if (empty($model)) {
            $user = new User();
            $user->username = 'admin';
            $user->email = 'thedrakoneragoni@gmail.com';
            $user->setPassword('admin');
            $user->generateAuthKey();
            if ($user->save()) {
                echo 'good';
            }
        }
    }

    public function actionSignup()
    {
        $model = new SignupForm();

        if ($model->load(Yii::$app->request->post())) {
            if ($user = $model->signup()) {
                if (Yii::$app->getUser()->login($user)) {
                    return $this->goHome();
                }
            }
        }

        return $this->render('signup', [
            'model' => $model,
        ]);
    }

    /**
     * Requests password reset.
     *
     * @return mixed
     */
    public function actionRequestPasswordReset()
    {
        $model = new PasswordResetRequestForm();

        if ($model->load(Yii::$app->request->post()) && $model->validate()) {
            if ($model->sendEmail()) {
                Yii::$app->session->setFlash('success', 'Check your email for
further instructions.');
                return $this->goHome();
            } else {
                Yii::$app->session->setFlash('error', 'Sorry, we are unable to
reset password for email provided.');
            }
```

```
        }

        return $this->render('requestPasswordResetToken', [
            'model' => $model,
        ]);
    }

    /**
     * Resets password.
     *
     * @param string $token
     * @return mixed
     * @throws BadRequestHttpException
     */
    public function actionResetPassword($token)
    {
        try {
            $model = new ResetPasswordForm($token);
        } catch (InvalidParamException $e) {
            throw new BadRequestHttpException($e->getMessage());
        }

        if ($model->load(Yii::$app->request->post()) && $model->validate() &&
$model->resetPassword()) {
            Yii::$app->session->setFlash('success', 'New password was saved.');
            return $this->goHome();
        }

        return $this->render('resetPassword', [
            'model' => $model,
        ]);
      }
}
```

Отображение домашней страницы:

```
<?php

/* @var $this yii\web\View */

$this->title = 'Pharmacy accounting';
?>
<div class="site-index">

    <div class="jumbotron">
        <h1>Welcome!</h1>

        <p class="lead">You visited 'Pharmacy accounting'.</p>

        <p><a class="btn btn-lg btn-success"
href="index.php?r=site%2Fabout">About</a></p>
    </div>

    <div class="body-content">

        <div class="row">
            <div class="col-lg-4">
                <h2>Our employees</h2>

                <p>Here you can get acquainted with our employees.</p>

                <p><a class="btn btn-default"
href="index.php?r=employee%2Findex/">Employees &raquo;</a></p>
            </div>
```

```
            <div class="col-lg-4">
                <h2>Our meds</h2>

                <p>Here you can find a list of our medicines.</p>

                <p><a class="btn btn-default"
href="index.php?r=meds%2Findex/">Meds &raquo;</a></p>
            </div>
            <div class="col-lg-4">
                <h2>Delivery</h2>

                <p>Our delivery</p>

                <p><a class="btn btn-default"
href="index.php?r=delivery%2Findex">Delivery &raquo;</a></p>
            </div>
            <br/>
            <div class="col-lg-4">
                <h2>Meds Group</h2>

                <p><a class="btn btn-default"
href="index.php?r=meds_group%2Findex">Delivery &raquo;</a></p>
            </div>
            <div class="col-lg-4">
                <h2>Provider</h2>

                <p><a class="btn btn-default"
href="index.php?r=provider%2Findex">Delivery &raquo;</a></p>
            </div>
            <div class="col-lg-4">
                <h2>Sales</h2>

                <p><a class="btn btn-default"
href="index.php?r=sale%2Findex">Delivery &raquo;</a></p>
            </div>
        </div>

    </div>
</div>
```

Модель пользователя:

```
<?php

namespace app\models;

use Yii;
use yii\base\NotSupportedException;
use yii\behaviors\TimestampBehavior;
use yii\db\ActiveRecord;
use yii\web\IdentityInterface;

class User extends ActiveRecord  implements IdentityInterface
{
    const STATUS_DELETED = 0;
    const STATUS_ACTIVE = 10;

    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return '{{%user}}';
    }
```

```
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            TimestampBehavior::className(),
        ];
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            ['status', 'default', 'value' => self::STATUS_ACTIVE],
            ['status', 'in', 'range' => [self::STATUS_ACTIVE,
self::STATUS_DELETED]],
        ];
    }

    /**
     * @inheritdoc
     */
    public static function findIdentity($id)
    {
        return static::findOne(['id' => $id, 'status' => self::STATUS_ACTIVE]);
    }

    /**
     * @inheritdoc
     */
    public static function findIdentityByAccessToken($token, $type = null)
    {
        throw new NotSupportedException('"findIdentityByAccessToken" is not
implemented.');
    }

    /**
     * Finds user by username
     *
     * @param string $username
     * @return static|null
     */
    public static function findByUsername($username)
    {
        return static::findOne(['username' => $username, 'status' =>
self::STATUS_ACTIVE]);
    }

    /**
     * @inheritdoc
     */
    public function getId()
    {
        return $this->getPrimaryKey();
    }

    /**
     * @inheritdoc
     */
    public function getAuthKey()
```

```
    {
        return $this->auth_key;
    }

    /**
     * @inheritdoc
     */
    public function validateAuthKey($authKey)
    {
        return $this->getAuthKey() === $authKey;
    }

    /**
     * Validates password
     *
     * @param string $password password to validate
     * @return bool if password provided is valid for current user
     */
    public function validatePassword($password)
    {
        return Yii::$app->security->validatePassword($password, $this-
>password_hash);
    }

    /**
     * Generates password hash from password and sets it to the model
     *
     * @param string $password
     */
    public function setPassword($password)
    {
        $this->password_hash = Yii::$app->security-
>generatePasswordHash($password);
    }

    /**
     * Generates "remember me" authentication key
     */
    public function generateAuthKey()
    {
        $this->auth_key = Yii::$app->security->generateRandomString();
    }

    public static function findByPasswordResetToken($token)
    {

        if (!static::isPasswordResetTokenValid($token)) {
            return null;
        }

        return static::findOne([
            'password_reset_token' => $token,
            'status' => self::STATUS_ACTIVE,
        ]);
    }

    public static function isPasswordResetTokenValid($token)
    {

        if (empty($token)) {
            return false;
        }

        $timestamp = (int) substr($token, strrpos($token, '_') + 1);
```

```
        $expire = Yii::$app->params['user.passwordResetTokenExpire'];
        return $timestamp + $expire >= time();
    }

    public function generatePasswordResetToken()
    {
        $this->password_reset_token = Yii::$app->security-
>generateRandomString() . '_' . time();
    }

    public function removePasswordResetToken()
    {
        $this->password_reset_token = null;
    }
}
```