Реферат

Структура работы представлена следующими разделами:

Введение. В разделе формулируется цель работы, поставленные задачи и необходимость данной работы.

Раздел 1. Анализ предметной области автоматизации процессов в медицинской организации. Изучение и анализ взаимосвязей внутри медицинской организации для их последую-щей автоматизации. Рассмотрение существующих систем управления медицинскими организациями. Требования к системе. Изучение и сравнительный анализ моделей жизненного цикла ПО с целью выбора наиболее подходящей. Изучение и сравнительный анализ подходов построения современных web-систем.

Раздел 2. Моделирование АСУ MedMS. Моделирование архитектурного подхоа АСУ MedMS. Моделирование базы данных.

Раздел 3. Реализация web-системы MedMS. Технологии используемые при реализации web-приложения. Создание базы данных.

Объем работы составляет 22 страницы.

Пояснительная записка содержит 5 схем.

В ходе проведение исследования было изучено и проанализировано 14 источников литературы.

Содержание

Введение

Автоматизированные информационные системы (АИС) в сфере медицины на сегодняшний день становятся незаменимыми частями медицинских организаций по всему миру. Огромное количество потоков разнородной информации вынуждают медицинские организации к автоматизации, частью которой является использование АИС.

На данный момент существует несколько уже готовых АИС специализированных на медицинских процессах каждая из которых обладает своими преимуществами и недостатками. Более подробно о уже существующих системах в главе **1.2**

В данной работе будут рассмотрены процессы, сущности и взаимодействия между ними в рамках медицинской организации. В качестве решения проблемы автоматизации выбрано создание единой централизованной информационной системы MedMS (Medical Management System). Этапы моделирования, проектирования, создания и внедрения описаны в рамках данной работы.

Цель данной работы заключается в следующем:

Разработать автоматизированную информационную web-систему MedMS, для автоматизации работы с пациентами и организационных процессов внутри медицинской организации.

Для достижения поставленной цели сформулированы следующие задачи:

1. Изучить все внутренние взаимодействия внутри медицинской организации, проблемы которые существуют при отсутствии автоматизации.
2. Изучить готовые варианты.
3. Сформировать требования к проектируемой web-системе
4. Спроектировать и реализовать систему MedMS.

1. Анализ предметной области автоматизации процессов в медицинской организации

Медицинская организация – организация, осуществляющие деятельность в области здравоохранения или оказания медицинских услуг, занимающаяся мероприятиями по поддержанию здоровья и оказания медицинской помощи людям посредством изучения, диагностики, лечения и возможной профилактики болезней и травм.

Основные формы медицинских организаций:

- Клиника – пациенты получают медицинские услуги амбулаторно (на приеме и на дому)
- Больница – пациенты получают медицинские услуги стационарно.
- Лаборатория – проводит медицинские исследование.
- Скорая медицинская помощь – прием заявок оказание мед. услуг на дому с возможной транспортировкой в другие мед организации.

1.1. Изучение и анализ взаимосвязей внутри медицинской организации для их последую-щей автоматизации.

Отличительные особенности медицинских организаций для АИС относительно клиники:

- Больница – добавляется услуга нахождения в стационаре
- Лаборатории – убираются услуги врачей.
- Скорая помощь – Убираются все услуги кроме тех, которые может оказать бригада скорой помощи.

Пользователи АИС / Роли в медицинской организации:

1. Пациенты (так как сотрудники организации могут пользоваться мед услугами учреждения, все пользователи по умолчанию пациенты)

2. Врачи – специалисты с высшим медицинским образованием, оказывающие услуги пациентам, ведут приемы, ставят диагнозы, выписывают лекарственные средства, выносят заключения по некоторым услугам (Рентген, МРТ, ЭКГ и т. д). Необходима история болезней пациента при приеме.
3. Лаборанты – специалист по исследованию образцов. Для заполнения информации о заказе, не нужна история болезней пациента.
4. ЛПИ – Лица принимающие исследования / собирающее образцы (Рентгенолог, медсестра и т.д.), не нужна история болезней пациента.
5. Регистратура – лица принимающее обращения пациентов, информирует, направляет, ведет запись на медицинские услуги.
6. Управляющие складом – лица ведущие контроль над медицинским складом, ведущие учет и инвентаризацию.
7. Администраторы – лица управляющие пользователями и правами всех пользователей в системе.

1.2. Рассмотрение существующих систем управления медицинскими организациями.

В рамках данной работы были изучены следующие системы с открытым исходным кодом:

1. HMS Open source Hospital Management System (HMS) (Nuxt.js, Vue.js)

Ссылка: https://github.com/ospic/webapp

Автор: Elirehema Paul (Dar es salaam, Tanzania)

Преимущества:

- Готовое решение с открытым исходным кодом
- Современный внешний вид

2. Clinic-management-system (PHP, TSQL)

Ссылка: https://github.com/bishosilwal/clinic-management-system

Автор: Bisho Silwal (Kathmandu, Nepal)

Преимущества:

- Готовое решение с открытым исходным кодом

Недостатки:

- Отсутствие возможности управления складом организации.
- Отсутствие механизмов расписания приемов/рабочих графиков сотрудников.
- Отталкивающий пользовательский интерфейс.

Общие недостатки систем:

- Отсутствие СНГ локализации (русский, казахский, киргизский и т.д.).
- Огромное количество кода, в котором могут быть преднамеренные уязвимости, что особенно важно когда идет работа с медицинскими данными пациентов.
- Отсутствует online доступ пациента к оформлению и результатам услуг.
- Изменения общих данных (цветовая тема, глубокой настройки параметров и т.д) возможно только через редактирование исходного кода.
- Отсутствие контейнеризации, что ведет к сложностям с запуском/перемещением/настройкой системы.

Вывод:

Все вышеперечисленные недостатки отталкивают медицинские учреждения к использованию автоматизированных систем в своей деятельности. В своем

проекте я учел все недостатки уже готовых АИС, изучил желание пациентов и требования клиники

1.3. Изучение и сравнительный анализ моделей жизненного цикла ПО с целью выбора наиболее подходящей

Жизненный цикл программного обеспечения — ряд событий, происходящих с системой в процессе ее создания и дальнейшего использования. (время от начального момента создания какого либо программного продукта, до конца его разработки и внедрения.)[1]

Жизненный цикл программного обеспечения можно представить в виде моделей.

Модель жизненного цикла программного обеспечения — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта.

Основные модели жизненного цикла ПО выделенные из [1]:

Каскадная модель жизненного цикла ПО (водопад)

Рисунок 1 Схема каскадной модели ЖЦ ПО

Преимущества:

- Последовательное выполнение этапов проекта в строгом фиксированном порядке
- Позволяет оценивать качество продукта на каждом этапе

Недостатки:

- Отсутствие обратных связей между этапами

- Не соответствует реальным условиям разработки программного продукта

Спиральная модель жизненного цикла ПО

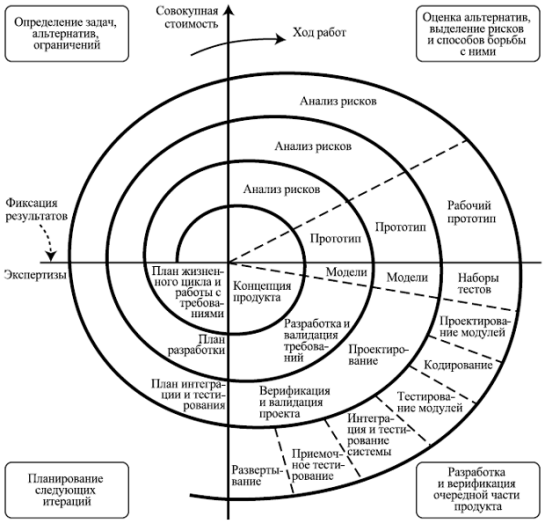

Рисунок 2 схема спиральной модели ЖЦ ПО

Преимущества:

- Быстрое получение результата
- Повышение конкурентоспособности
- Изменяющиеся требования — не проблема

Недостатки:

- Отсутствие регламентации стадий

Ввиду следующих характеристик предметной области была выбрана спиральная модель:

1. Возможность разграничения процессов предметной области по независимым модулям, что способствует разработке АСУ поэтапно.
2. Изначальные требования размыты и туманны. В начале разработки ни у одного эксперта в данной предметной области, на предприятии не было конечного видения результата разработки АСУ.
3. Очень сжатые сроки на минимальные необходимые модули, для запуска электронного учебного процесса.

1.4. Изучение и сравнительный анализ подходов построения современных web-систем.

На сегодняшний день наиболее распространенным видом получения информационных услуг является использование веб-приложений через сеть интернет

Основные подходы построения web-систем выделенные из [2]:

1. **Традиционные SSR**(Server Side Rendering), **MPA**(Multi Page Application) – при каждом изменении данных или загрузке новой информации страница обновляется. Сервер обсчитывает данные, выполняет бизнес логику, после чего сервер формирует (рендерит HTML) страницу и отправляет готовую страницу пользователю.

   Преимущества:

   - Минимальные технические требования к клиентской стороне, к hardware и software вплоть до отсутствия поддержки JavaScript.
   - Простая SEO оптимизация — можно оптимизировать каждую из страниц приложения под нужные ключевые запросы;

   Недостатки:

   - Тесная связь между бекендом и фронтендом, поэтому их не получается развивать параллельно
   - По протоколу передачи данных (HTTP/HTTPS) при каждом запросе передается большой объем данных (полностью готовая страница HTML, CSS, JS)

2. **Одностраничные SPA**(Single Page Application), **CSR**(Client Side Rendering), – после первой загрузки страницы пользователь видит весь

основной контент, а при переходах на другие страницы или внесении данных, вместо полной перезагрузки нужные элементы подгружаются и отрисовываются динамически средствами браузера клиента.
Для связи с сервером используется API.
Реализуется с помощью современных, продвинутых фреймворков JavaScript: Vue, React, Angular

Преимущества:

- Высокая скорость работы.
- Стирается грань между веб-системой и десктопным приложением.
- Мгновенная реакция приложения на действия пользователя, без задержек.
- Гибкость и отзывчивость пользовательского интерфейса.
- Независимость разработки Бэк-енд и фронт-енд, что ускоряет разработку
- Кэширование данных — приложение отправляет всего один запрос, собирает данные, а после этого может функционировать в offline-режиме.

Недостатки:

- Требовательны к клиентской стороне, как с точки зрения hardware, так и software.
- Необходима поддержка JavaScript

3. **Иммитация SPA**, SSR – Комбинирование подходов п.1 и п.2. Серверная часть реализуется аналогично с п.1, однако при получении ответа на запрос пользователя, получив сгенерированную страницу сторона клиента ищет различия в DOM дереве HTML до запроса и после, и в случае их различия, заменяет изменившиеся элементы

Реализуется средствами библиотеки Turbolinks, доступная для языков Ruby, PHP, Python

Преимущества:

- Скорость работы выше традиционных MPA.
- Простота реализации на уже готовом MPA SSR приложении, путем установки соответственного пакета NPM
- При переходах на другие страницы или внесении данных, вместо полной перезагрузки нужные элементы подгружаются.
-
- Мгновенная реакция приложения на действия пользователя, без задержек.
- Стирается грань между веб-системой и десктопным приложением.
- Мгновенная реакция приложения на действия пользователя, без задержек.
- Гибкость и отзывчивость пользовательского интерфейса.
- Независимость разработки Бэк-енд и фронт-енд, что ускоряет разработку
- Кэширование данных — приложение отправляет всего один запрос, собирает данные, а после этого может функционировать в offline-режиме.

Недостатки:

- Скорость работы ниже одностраничных SPA.
- При реализации на уже готовом MPA SSR, возникает проблема с событием ready (Страница готова) так как страница не рендерится а изменяется.
- Конфликтует с некоторыми библиотеками JS.
- Требовательны к клиентской стороне, как с точки зрения hardware, так и software.

- Необходима поддержка JavaScript.

4. **PWA**(Progressive web application) – технология веб-разработки, которая визуально и функционально трансформирует сайт в приложение для различных устройств. [4].

   Является логическим продолжением SPA п.2 [3]

   Преимущества:

   - Все преимущества технологии SPA
   - Можно установить на домашний экран смартфона или же на рабочий стол компьютера, и в дальнейшим это приложение будет функционировать как нативное.
   - Улучшение производительности с помощью предварительной загрузки ресурсов.
   - Работоспособность, независимо от наличия подключения к сети.
   - Возможность добавлять PWA в онлайн-магазины мобильных приложений App Store и Google Play.

   Недостатки:

   - Все недостатки технологии SPA.
   - Не все операционные системы поддерживает полный функционал PWA.
   - По сравнению с нативными приложениями, PWA не могут делать все, что могут обычные приложения. (отсутствует доступ к таким важным функциям устройства, как Bluetooth, внешнее освещение, датчики приближения, возможность управления расширенными элементами камеры и другие.)
   - Работа офлайн ограничена

2. Моделирование АСУ MedMS.

2.1. Моделирование архитектурного подхоа АСУ MedMS.

Для АСУ MedMS был выбран подход одностраничный SPA CSR:

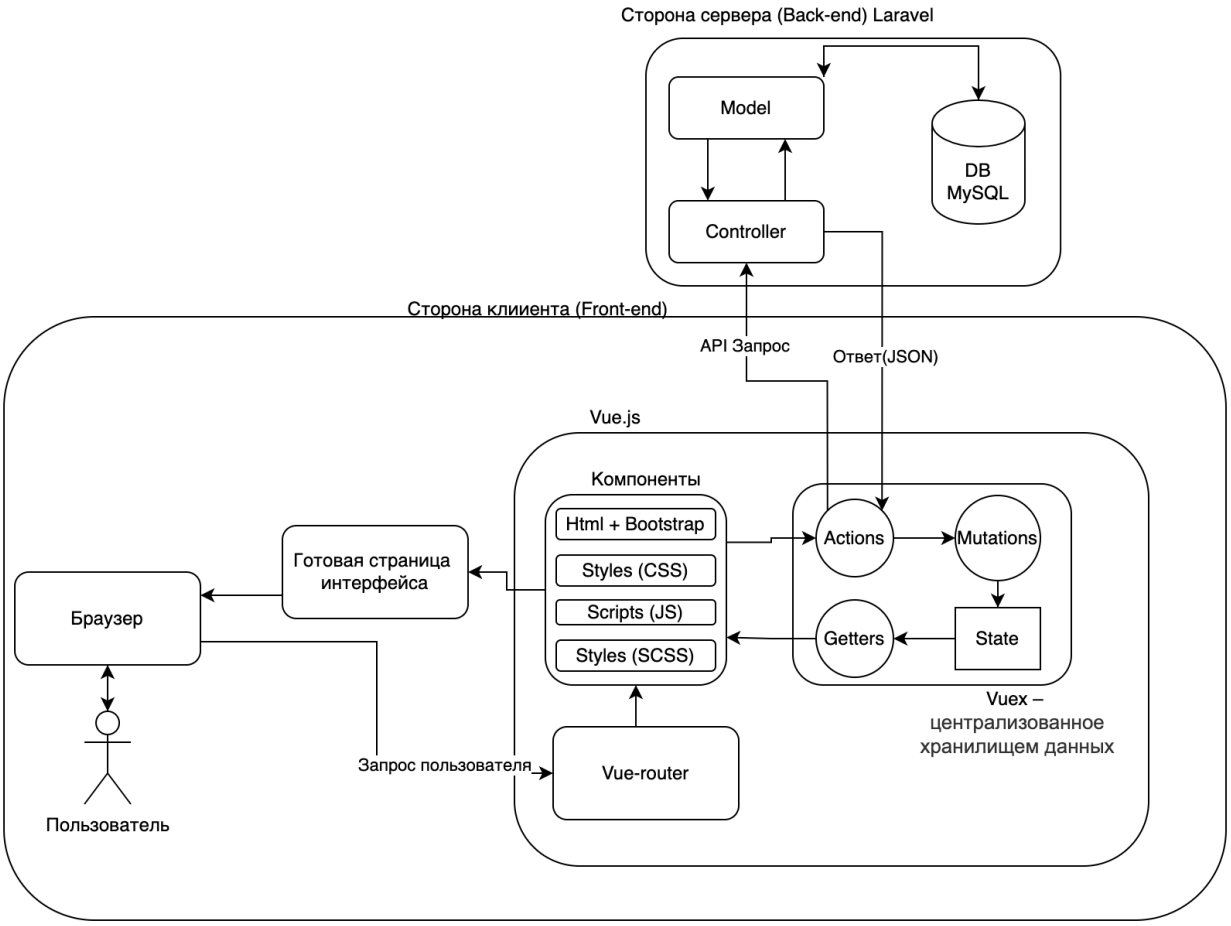

Рисунок 3 Схема работы SPA CSR приложения

2.2. Моделирование базы данных.

В медицинской организации был выделен следующий набор сущностей:

1. Услуги (services)
2. Категории Услуг (categories)
3. Пользователи (users)
4. Права доступа
5. Роли
6. Оформленные заказы
7. Оборудование
8. Передвижение оборудования
9. Настройки
10. Типы иконок
11. Иконки
12. Медиа
13. Шаблоны документов
14. Документы пациентов
15. Рабочий график сотрудников

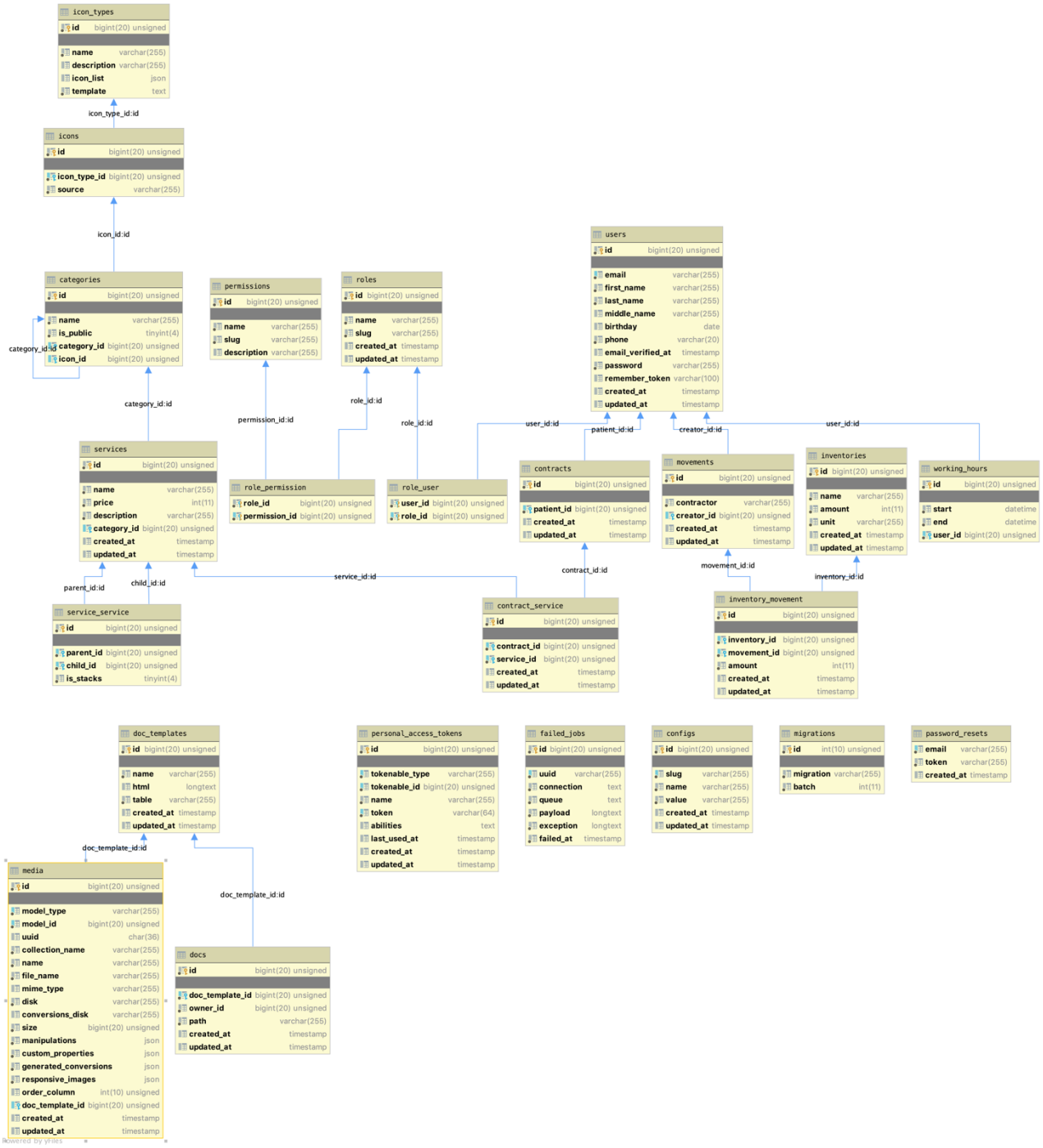

Рисунок 4 физическая модель базы данных

3. Реализация web-системы MedMS.

3.1. Технологии используемые при реализации web-приложения.

Подход SPA CSR был реализован с использованием следующих технологий.

На стороне сервера были использованы:

- Laravel 8 framework (PHP)
- MySQL (СУБД)
- Docker (Контейнеризация)
- NGINX (сервер)
- Ubuntu 20.04 (ОС сервера)
- NPM (Пакетный менеджер JavaScript)
- Webpack mix (Сборка и минификация front-end)
- API (доступ к серверу)

На стороне клиента:

- HTML
- CSS
- Bootstrap
- Vanilla JS
- Вспомогательные JS библиотеки
    1. Chart.js (Библиотека для работы с графиками)
    2. Datatables.net (Работа с таблицами, поиск/сортировка/экспорт)
    3. Fullcalendar.js (Работа с расписанием/календарями)
    4. V-viewer (Просмотр pdf документов)
    5. vue-document-editor (WYSIWYG-редакторов, для создания шаблонов документов)

- Экосистема Vue.js
    1. Vue,js
    2. Vue-router (одностраничная SPA маршрутизация)
    3. Vuex (управление состоянем приложения)
    4. Vuex-persistedstate (хранение состояния при перезагрузке)
    5. Vue-meta (изменение мета тегов одностраничного приложеня)
- Axios (асинхронные API запросы на сервер)
- Lodash (предоставляет служебные функции для общих задач)

3.2. Создание базы данных.

База данных создается автоматизировано контейнеризатором docker-compose.

Таблицы в базе данных создавались с помощью миграций ORM Eloquent [13] (Расположены в порядке исполнения миграций):

2014_10_12_000000_create_users_table.php
2014_10_12_100000_create_password_resets_table.php
2019_08_19_000000_create_failed_jobs_table.php
2019_12_14_000001_create_personal_access_tokens_table.php
2022_02_13_142722_create_roles_table.php
2022_02_13_142728_create_permissions_table.php
2022_02_13_142734_create_role_user_table.php
2022_02_13_142739_create_role_permission_table.php
2022_03_20_142848_create_categories_table.php
2022_03_20_142916_create_services_table.php
2022_03_20_142944_create_service_service_table.php
2022_03_22_112139_create_contracts_table.php
2022_03_23_082836_create_contract_service_table.php

2022_04_12_102615_create_configs_table.php

2022_04_18_082928_create_working_hours_table.php

2022_04_24_163151_create_doc_templates_table.php

2022_04_24_163232_create_docs_table.php

2022_05_02_214448_create_icon_types_table.php

2022_05_02_214453_create_icons_table.php

2022_05_04_021052_connect_categories_icons.php

2022_05_06_103757_create_media_table.php

2022_05_09_002439_create_inventories_table.php

2022_05_09_002501_create_movements_table.php

2022_05_09_002526_create_inventory_movement_table.php

2022_05_13_161127_create_service_user_table.php

3.3. Модульная файловая структура системы

Одно из требований к системе – Модульность. В АСУ LMS есть 2 типа модулей Back-end и Front-end.

При реализации выполнялся принцип открытости к расширению и закрытости к модификации. Если возникала необходимость связи модулей, зачастую это достигалось расширением моделей при помощи Трейтов (Множественное наследование PHP).

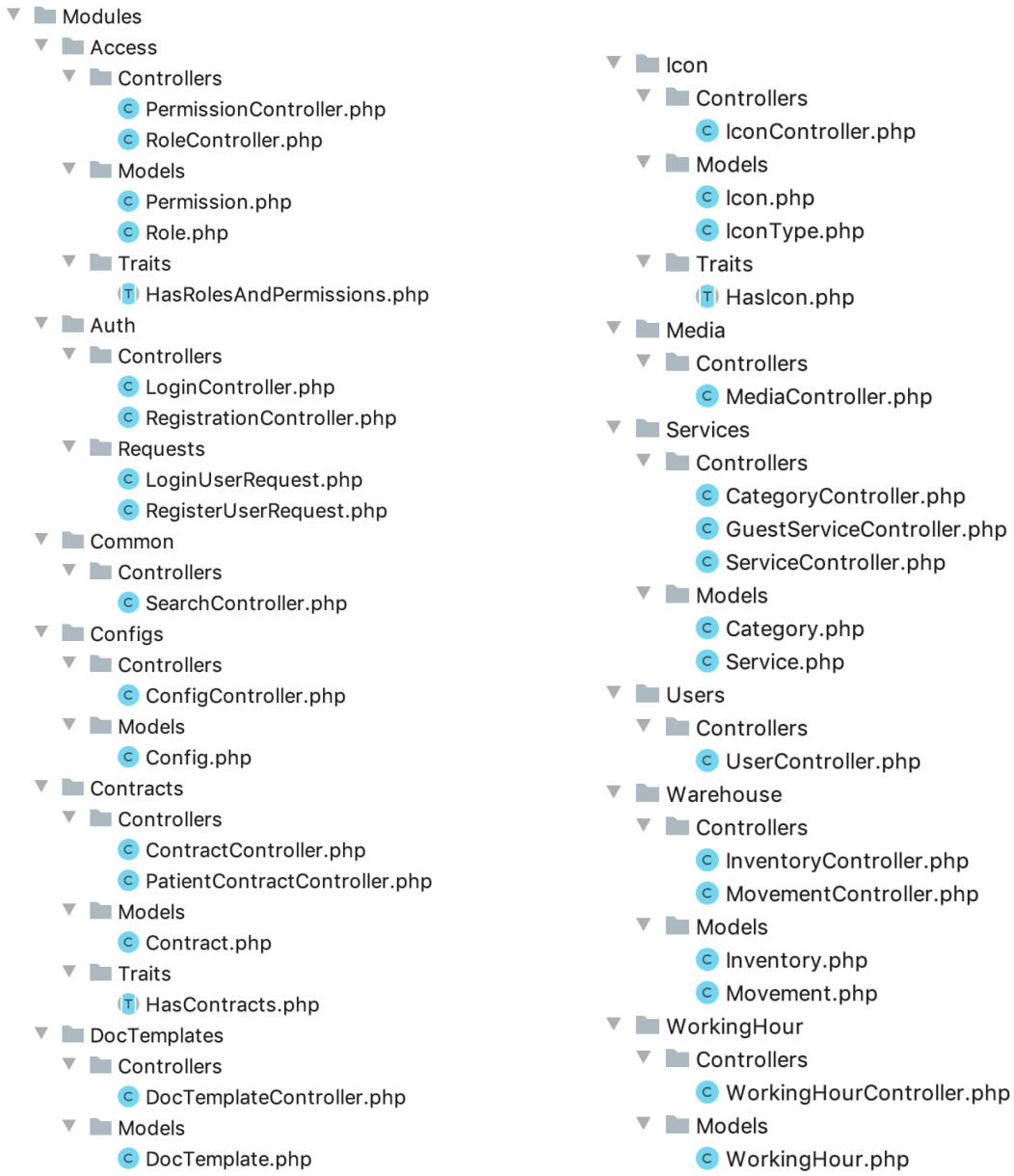

Рисунок 5 Back-end модули

Заключение

В данном проекте проанализированы: информационные потребности медицинских организаций, различные способы создания и поддержки АИС, опыт уже созданных и внедренных систем управления медицинскими организациями.

В ходе выполнения данной работы были изучены: Различные модели жизненного цикла ПО, Современные подходы к разработке web-приложений, Реляционные базы данных и методы их моделирования, язык создания web-приложений PHP и фреймворк Laravel, способы контейнеризации docker-compose, поддержка web-сервера Nginx, администрирование UNIX систем, создание динамичных интерфейсов SPA приложений, фреймворк Vue.js.

На основании полученных знаний была разработана система MedMS, удовлетворяющая информационным запросам всех участников процесса получения пациентами медицинских услуг.

Данная версия системы была протестирована и внедрена в эксплуатацию в клинике при медицинском университете.

Список литературы

1. Берг Д.Б., Ульянова Е.А., Добряк. П.В. / Модели жизненного цикла : учеб. пособие // — Екатеринбург : Изд-во Урал. ун-та, 2014. — 74 с.
2. Гридин В.Н., Анисимов В.И., Васильев С.А. / Методы повышения производительности современных веб-приложений // Известия ЮФУ. Технические науки. 2020. №2 –С.193-200.
3. Валитова Н.Л., Кремлева Э.Ш., Кашафутдинов Р.К. / Перспективы применения технологии PWA // ОТО. 2020. №1. С.115-124.
4. Газизуллин Н.И., Плещинская И.Е. / Разработка прогрессивного веб-приложения с помощью технологии PWA // StudNet. 2020. №8. С.620-625.
5. MySQL: использование и администрирование. Викрам Васвани / В. Васвани — Санкт-Петербург.: Издательский центр «Питер», 2011. — 368 с.
6. Н.А. Борсук, О.О. Козеева / Сравнительный анализ языков программирования Python и php // Символ науки. –2017. –4. –С.33-36.
7. Е.С. Шевченко / Сравнительное тестирование PHP-фреймворков // Вестник науки и образования. –2019. –10-2 (64). –С.40-45.
8. Кузовкин А.В., Цыганов А.А., Щукин Б.А. / Управление данными: учебник для студ. высших учеб. заведений // — М.: Издательский центр «Академия», 2010. — 256 с.
9. Мезенцев К.Н. / Автоматизированные информационные системы : учебник для студ. учреждений сред. проф. образования — 4-е изд., стер. // — М. : Издательский центр «Академия», 2013. — 176 с.
10. Сысолетин Е.Г., Ростунцев С.Д. / Проектирование интернет-приложений : учеб.-метод. пособие // — Екатеринбург : Изд-во Урал. ун-та, 2015 — 92 с.
11. Adel F. / Архитектура сложных веб приложений // 2020 — 271 с.

12. Брусов А.С., Тарасов С.О. / Использование фреймворка Laravel 5. 0 для разработки web-приложений // Современные материалы, техника и технологии. –2015. –3 (3). –С.48-52.

13. ETL Tools / David Haertzen. // The Analytical Puzzle: Profitable Data Warehousing, Business Intelligence and Analytics. — Technics Publications, 2012. — 346 с.

14. https://docs.laravel-excel.com/