АРХИТЕКТУРА И ИНСТРУМЕНТАРИЙ ДЛЯ РАЗРАБОТКИ МЕДИЦИНСКОЙ ВЕБ-СИСТЕМЫ АНАЛИЗА ЗАБОЛЕВАНИЙ ЛЁГКИХ

С.А. Семёнов, С.В. Аксёнов
Томский политехнический университет
E-mail: sergey9@tpu.ru

Введение

Заболевания лёгких в настоящее время входит в десятку самых распространённых причин смертей среди населения. Для выявления заболеваний лёгких используются данные полученные с помощью эффективных радиологических исследований, которые являются не травмирующими и не инвазивными. На протяжении жизни, а также в процессе лечения, радиологические исследования проводятся неоднократно. На текущий момент процедура по выявлению заболеваний лёгких выполняется вручную радиологами.

В данном работе используются следующие методы радиологических исследований: флюорография и компьютерная томография. Выходные данных радиологических исследований представляют собой снимки, которые подаются на вход нейронной сети. Помимо снимков предлагаемая система позволяет хранить и обрабатывать дополнительные медицинские анализы.

В данной работе, представлена архитектура разрабатываемого программного обеспечения (ПО), а также анализ и подбор стека технологий, позволяющего написать качественного программного обеспечения. В разработке предлагаемой системы совместно участвуют команды программистов и врачей.

Описание функционала

Функционал разрабатываемого веб-приложения включает:

1. Регистрация и проверка прав пользователя;
2. Управление базой пациентов – добавление, редактирование;
3. Управление и анализ результатов общих медицинских исследований;
4. Управление и анализ снимков полученных с помощью радиологических исследований.

На основе выше указанного функционала, проводилось прототипирование пользовательского интерфейса. Прототипирование пользовательского интерфейса – представляют собой итеративный метод разработки интерфейсов, при котором пользователи активно участвуют в создании пользовательского интерфейса для системы [1, 2].

Результатом прототипирования являются следующие интерфейсы (страницы вебприложения):

1. Регистрация пользователя;
2. Вход в систему;
3. Интерфейс добавления нового пациента и редактирования данных ранее добавленных пациентов;
4. Интерфейс проведения анализа и редактирования общих медицинских исследований;
5. Интерфейс проведения анализа радиологических исследований.

Проектирование прототипов осуществлялось с помощью программного пакета, предназначенного для прототипирования Marvel [3].

Архитектура

Предлагаемое программное обеспечения, представляет собой веб-приложение, развернутое в облаке (доступное через интернет). Вебприложение – клиент-серверное приложение, в котором клиент взаимодействует с вебсервером при помощи браузера. Логика вебприложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами[4].

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер». Клиентская часть (браузер) реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует ответ (веб-страницу, json и т. п.) и отправляет его клиенту по сети с использованием протокола HTTP. При разработке веб-приложения клиентскую часть называют frontend, а серверную backend.

Разрабатываемое веб-приложение предлагается реализовать на основе N-слойной архитектуры. В данном случае трехслойная архитектура. Как правило, в приложении определяются слои пользовательского интерфейса (User Interface), бизнес-логики (Business Logic) и доступа к данным (Data Access). В рамках такой архитектуры пользователи выполняют запросы через слой пользовательского интерфейса, который взаимодействует только со слоем бизнеслогики. Слой бизнес-логики, в свою очередь, может вызывать слой доступа к данным для обработки запросов. Слой пользовательского

интерфейса не должен выполнять запросы напрямую к слою доступа к данным и какими-либо другими способами напрямую взаимодействовать с данными или внешними сервисами.

Аналогичным образом, слой бизнес-логики должен взаимодействовать с данными и внешними сервисами только через слой доступа к данным. Таким образом, для каждого слоя четко определена своя обязанность [5]. Архитектура проектируемого приложения представлена на рисунке 1.

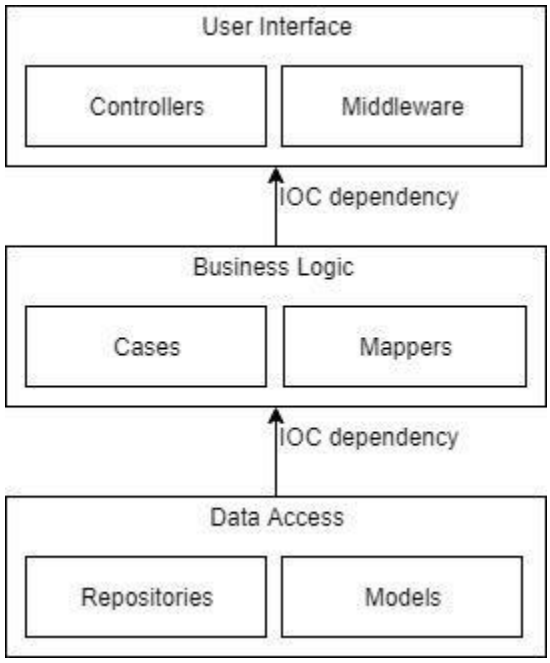

Рисунок 1 – Архитектура

Слой доступа данных включает в себя модуль модели таблиц базы данных (Models) и модуль репозиториев (Repositories). Модели представляют собой сущности, которые отображают данные хранящиеся в базе данных. Репозитории выполняют доступ к базе данных.

Слой бизнес логики включает модули Cases и Mappers. Задача модуля Mappers выполнять преобразования моделей базы данных в более удобный формат для отображения в браузере пользователя. Задача модуля Cases выполнение бизнес логики приложения используя слой доступа к данным.

Слой пользовательского интерфейса включает модули Controllers и Middleware. Модуль Controllers включает себя контролеры, которые выполняют обработку входящих запросов и осуществляет взаимодействие со слоем бизнес логики. Модуль Middleware добавляет дополнительную логику обработки для каждого запроса, например, логика проверки прав доступа.

Связь между слоями осуществляется с помощью подхода внедрения зависимостей (Dependency Injection). Внедрение зависимости (Dependency Injection, DI) — процесс предоставления внешней зависимости программному компоненту. Является специфичной формой «инверсии управления» (Inversion of control, IoC), когда она применяется к управлению зависимостями. В полном соответствии с принципом единственной обязанности объект отдаёт заботу о построении требуемых ему зависимостей внешнему, специально предназначенному для этого общему механизму[6].

Инструментарий

Важным решением перед разработкой программного обеспечения является выбор стека технологий, с помощью которого будет осуществляться разработка. При разработке вебприложения задача выбора удваивается, так как необходимо выбрать стек технологий для двух независимых частей: backend-a и frontend-a. Разработка frontend части в современном мире ведется практически всегда на языке

JavaScript (TypeScript). Однако, в последние годы разработали технологию WebAssembly, которая позволяет выполнять код в браузере на других языках (например, C#/Blazor), но так как данная технология появилась недавно (данная технология и библиотеки разрабатываемы на основе данной технологии еще слишком «молоды») авторами однозначно выбран JavaScript(TypeScript).

Из наиболее мощных и распространённых frontend фрэймворков есть три кандидата: Angular, ReactJS, VueJs. В данном случае выбор авторов пал на ReactJS, из-за философии разработки с использованием данного фрэймворка. Одна из особенностей React – это предлагаемый им процесс мышления при создании приложений. Данный процесс описан в официальной документации [7]. Основываясь на данном подходе можно разрабатывать приложения любых масштабов.

Для backend рассматривались два языка программирования Python и JavaScript. Язык Python включен в рассмотрение, так как в команде авторов есть опыт разработки на данном языке, а также обучение нейронных сетей выполняется на данном языке. Использовании JavaScript является удобным выбором, так как разработка frontend части выполняется на JavaScript. Таким образом требования к разработчикам на знание технологий уменьшается и это упрощает и ускоряет разработку, так как стек технологий требуемый для разработки веб-приложения велик. В данной разделе рассматриваются только самые необходимые технологии. Другие языки программирования рассматривать считается излишним, так как введение нового языка усложнит жизнь команде разработчиков.

Backend часть является наиболее сложной частью, соответственно требования к данной части веб-приложения более высокие. Авторами были выделены основные требования к «экосистеме»

(библиотеки и фрэймворки реализованные в языке программирования) языков программирования:

- Высокая поддержка, хорошая документация и простота использования основных фрэймворком необходимых для реализации веб-приложения – вебфрэймворк выполняющий обработку, входящий HTTP запросов и поддерживающий написание Middleware для входящих запросов, фрэймворк выполняющий работу с базой данных на основе подхода ORM;
- Поддержка асинхронного программирования и наличие синтаксиса async/await. Асинхронность в программировании — выполнение процесса в неблокирующем режиме системного вызова, что позволяет потоку программы продолжить обработку [8];
- Поддерживаемая, легкая и удобная в использовании реализация подхода Dependency Injection.

Язык JavaScript подходит по все трем требованиям. Фрэймворки JavaScript для веб-разработки: среда исполнения (интерпретатор) языка программирования JavaScript NodeJS [9], веб-фрэймворк Express, ORM-фрэймворк TypeORM. Синтаксис async/await поддерживает во всех библиотеках и языка программирования JavaScript. Фрэймворк InversifyJS реализующий подход Dependency Injection прост в использовании и удобна обладает огромной документацией, описывающей все методы на все случаи жизни.

В отличие от выше упомянутого языка, язык Python имеет поддержку синтаксиса асинхронного программирования async/await только единицах фрэймворках. Реализация подхода Dependency Injection сложна и неудобна в использовании.

На основе вышесказанного, авторами был выбран следующий инструментарий:

- Язык программирования для, frontend-а и backend-а – JavaScript (TypeScript);
- Frontend-фрэймворк – ReactJS;
- CSS-фрэймворк – UIkit;
- Сборщик клиентской части приложения – Webpack;
- Веб-фрэймворк – Express (NodeJS);
- ORM-фрэймворк – TypeORM;
- Реализация IoC – InversifyJS.

Заключение

В данной работе авторами спроектированы и проанализированы основные столпы, требуемые для разработки веб-портала, предназначенного для анализа заболеваний лёгких. Спроектированы и проанализированы следующие моменты:

- Функционал – описано, что разрабатываемое приложение должно
- делать, какие задачи выполнять;
- Архитектура – описано, что собой представляет веб-приложение, из каких частей состоит и какая часть приложения за что отвечает;
- Инструментарий – описан выбранный стек технологий.

Работа является фундаментом разрабатываемого авторами веб-портала, позволяющего проводить анализ заболеваний лёгких, что позволит оптимизировать и облегчить работу медицинского персонала.

Список использованных источников

1. Jerry Cao Ultimate Guide to Prototyping [Электронный ресурс] / Блог о дизайне и проектировании пользовательских интерфейсов Studio by UXPin – https://www.uxpin.com/studio/blog/new-free-ebook-ultimate-guide-prototyping/ (дата обращения 10.01.2020).
2. Scott W. Ambler User Interface Prototyping Tips and Techniques [Электронный ресурс] / Блог Скота Амблера о эффективных подходах реализации программного обеспечения – URL: http://www.ambysoft.com/essays/userInterfacePrototyping.html (дата обращения 10.01.2020).
3. Software for prototyping user interfaces [Электронный ресурс] / Официальный сайт Marvel – URL: https://marvelapp.com/ (дата обращения 11.01.2020).
4. Веб-приложение [Электронный ресурс] / Статья на сайте Wikipedia – URL: https://ru.wikipedia.org/wiki/Веб-приложение#Архитектура_веб-приложений (дата обращения 15.01.2020).
5. Общие архитектуры веб-приложений [Электронный ресурс] / Статья на сайте Microsoft – URL: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/common-web-applicationarchitectures (дата обращения 16.01.2020).
6. Martin Fowler Inversion of Control Container and Dependency Injection pattern [Электронный ресурс] / Блог Мартина Фаулера – URL: https://www.martinfowler.com/articles/injection.html (дата обращения 16.01.2020).
7. Философия React [Электронный ресурс] / Официальный сайт ReactJS – URL: https://ru.reactjs.org/docs/thinking-in-react.html (дата обращения 20.01.2020).
8. Борисов И. Асинхронность в программировании [Электронный ресурс] / Новостной портал о программировании Tproger – URL: https://tproger.ru/articles/asynchronous-programming/ (дата обращения 21.01.2020).
9. Главная страница Node JS [Электронный ресурс] / Официальный сайт NodeJS – URL: https://nodejs.org/en/ (дата обращения 22.01.2020).