DOI: 10.6060/snt.20248004.00017
УДК: 004.4

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ «МЕДБЛОКНОТ: ЗДОРОВЬЕ В ЦИФРАХ»

Наумова Е.М., Метелев В.А., Ксенофонтова О.Л., Смирнова Н.В.

Наумова Елена Михайловна (0009-0000-4759-3366), Метелев Владислав Александрович (0009-0004-8999-0764), Ксенофонтова Ольга Леонидовна 0000-0002-4122-6220),
Смирнова Наталия Владимировна (0000-0002-1873-5334)
Ивановский государственный химико-технологический университет,
г. Иваново, Россия. 153000, Ивановская область, г. Иваново, пр. Шереметевский, д. 7.
E-mail: sonyableid.mk@gmail.com, bckf2012@yandex.ru, olga_izvolova@mail.ru, nataliasmirnova_@mail.ru

В эпоху повсеместного внедрения цифровых технологий управление здоровьем переходит в цифровую плоскость. Разрабатывается все больше инструментов, направленных на расширение возможностей людей в вопросах управления своим здоровьем. В данной статье представлены результаты исследования по разработке веб-приложения, предназначенного для хранения и отслеживания динамики медицинских показателей. Основная цель работы заключалась в создании системы для самостоятельного контроля здоровья. Для достижения этой цели была проведена оценка целесообразности создания веб-приложения с помощью пользовательского интервью, сформированы функциональные требования, спроектирована архитектура приложения, сделан выбор инструментальных средств разработки, реализована разработка. В ходе исследования были применены современные инструменты разработки, такие как PostgreSQL, Prisma, TypeScript, NestJS, React, Chart.js, что позволило создать удобное в использовании приложение по мнению пользовательского тестирования. В заключение авторы подчеркивают перспективу развития продукта.

Ключевые слова: разработка, веб-приложение, здоровье, медицинские показатели, мониторинг здоровья, хранение данных, динамика, графики, визуализация данных, безопасность данных, PostgreSQL, Prisma, TypeScript, NestJS, React, Chart.js

DEVELOPMENT OF THE WEB-APPLICATION «MEDICAL NOTEBOOK: HEALTH IN NUMBERS»

Naumova E.M., Metelev V.A., Ksenofontova O.L., Smirnova N.V.

Naumova Elena Mikhailovna (0009-0000-4759-3366), Metelev Vladislav Aleksandrovich (0009-0004-8999-0764), Ksenofontova Olga Leonidovna (0000-0002-4122-6220),
Smirnova Natalia Vladimirovna (0000-0002-1873-5334)
Ivanovo State University of Chemical Technology,
Ivanovo, Russia. 153000, Ivanovo region, Ivanovo, Sheremetevsky ave., 7.
E-mail: sonyableid.mk@gmail.com, bckf2012@yandex.ru, olga_izvolova@mail.ru, nataliasmirnova_@mail.ru

In the era of widespread implementation of digital technologies, health management is moving into the digital plane. More and more tools are being developed to empower people to manage their health. This article presents the results of a study on the development of a web-application designed to store and track the dynamics of medical indicators. The main goal of the work was to create a system for self-monitoring of health. To achieve this goal, an assessment of the feasibility of creating a web-application was carried out using a user interview, functional requirements were formed, the application architecture was designed, development tools were selected, and the development was implemented. During the study, modern development tools were used, such as PostgreSQL, Prisma, TypeScript, NestJS, React, Chart.js, which made it possible to create an easy-to-use application according to user testing. In conclusion, the authors emphasize the prospects for product development.

Keywords: development, web application, health, medical indicators, health monitoring, data storage, dynamics, charts, data visualization, data security, PostgreSQL, Prisma, TypeScript, NestJS, React, Chart.js

ВВЕДЕНИЕ

Актуальность разработки веб-приложений связана со стремительным развитием цифровых технологий и ростом потребности пользователей в доступных и удобных решениях. Веб-приложения обеспечивают возможность работы с данными в реальном времени и оперативно принимать решения. Это важный аспект, например, для бизнеса, здравоохранения, образования [1, 2]. Они позволяют интегрировать различные сервисы и инструменты, улучшая взаимодействие с клиентами и повышая эффективность процессов. Кроме того, с увеличением числа мобильных устройств веб-приложения становятся важным инструментом для достижения аудитории, обеспечивая кросс-платформенность и доступность. В условиях быстро меняющегося рынка разработка современных веб-приложений становится ключевым фактором конкурентоспособности компаний [3].

Актуальность создания медицинских веб-приложений обусловлена необходимостью улучшения доступа к медицинским услугам, оптимизации управления данными пациентов и повышения качества обслуживания. Такие приложения обеспечивают телемедицину, запись на прием, доступ к медицинским записям и консультациям, что способствует эффективному взаимодействию между врачами и пациентами в условиях современного здравоохранения.

Мониторинг динамики медицинских показателей является одним из важнейших аспектов медицины. Он позволяет отслеживать состояние здоровья, выявлять отклонения от нормы на ранних стадиях и своевременно принимать необходимые меры [4, 5].

Главным недостатком традиционного ручного метода мониторинга является сложность анализа и визуализации больших объемов данных. В связи с этим, возрастает потребность автоматизировать процессы сбора, хранения, анализа и визуализации динамики медицинских показателей.

Разрабатываемое приложение полезно как для пользователей, которые могут самостоятельно контролировать свое здоровье и своевременно принимать необходимые меры, так и для медицинских работников, которые могут выявлять и предупреждать ряд заболеваний у пациентов на ранних стадиях, и оптимизировать процесс принятия решений на основе анализа динамики медицинских показателей.

Таким образом, приложение способствует более эффективному взаимодействию пользователя с самим собой и пациента с врачом.

ОЦЕНКА ЦЕЛЕСООБРАЗНОСТИ СОЗДАНИЯ ВЕБ-ПРИЛОЖЕНИЯ

Пользовательское интервью является важным методом исследования в процессе разработки продукта, поскольку оно предоставляет разработчикам глубокое понимание предпочтений и потребностей будущих пользователей.

Результаты анализов и заключения врачей хранят 89% опрошенных (рис. 1).

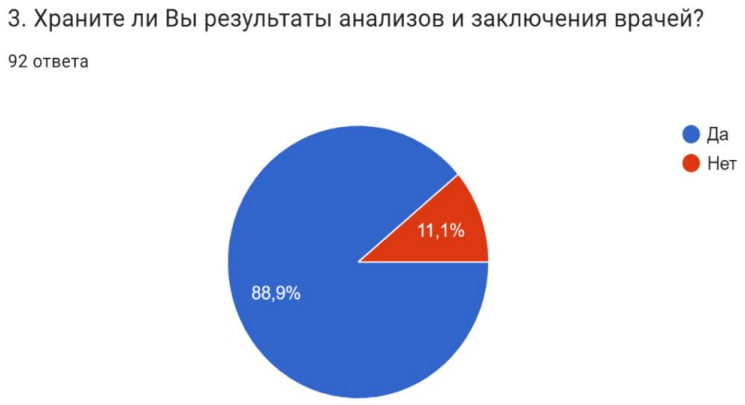

Рис. 1. Статистика ответов на вопрос
Fig. 1. Statistics of answers to the question

На вопрос «С какими трудностями Вы столкнулись при организации их хранения?» респонденты ответили следующее:

1. Потеря и повреждение бумажных и электронных носителей;
2. Долгие поиски актуального медицинского документа;
3. Нет постоянного доступа к бумажным носителям;
4. Бумажные носители занимают много места, а электронные много памяти;
5. Небезопасность хранения медицинских документов в поликлинике, так как бумажные медицинские карты выдаются пациентам без подтверждения личности документами.

Рис. 2. Статистика ответов на вопрос
Fig. 2. Statistics of answers to the question

100% опрошенных считают, что концепция и предполагаемый функционал веб-приложения «Медблокнот: Здоровье в цифрах» решают их проблемы организации хранения медицинских документов (рис. 2).

Таким образом, участники пользовательского интервью определяют целевую аудиторию разрабатываемого веб-приложения.

Поэтому в нашем случае целевой аудиторией являются лица мужского и женского пола, достигшие 18 лет (так как до совершеннолетия ответственность за здоровье ребенка несут родители) и не достигшие 58 лет у женщин и 63 лет у мужчин (так как с наступлением пенсионного возраста ответственность за здоровье лиц пожилого возраста ложиться на их детей). А также за здоровьем чаще следят лица, имеющие высшее образование, работающие в самых различных сферах деятельности.

Согласно пользовательскому интервью, сильными сторонами разрабатываемого решения являются: возможность самостоятельного ведения медицинской карты, решение проблемы фрагментации данных, хранение медицинских документов в одном месте, удобство отслеживания динамики медицинских показателей, быстрый доступ к медицинским документам, возможность делиться с врачом результатами динамики медицинских показателей и возможность получения уведомлений с персональными рекомендациями по улучшению здоровья.

ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Функциональные требования регламентируют функциональность системы. Это тип требований, который описывает, что должна делать система, ее ожидаемые функции и возможности.

Функциональные требования к разрабатываемому веб-приложению:

1. Веб-приложение должно предоставлять пользователю возможность регистрации аккаунта;
2. Веб-приложение должно предоставлять пользователю возможность авторизации в аккаунте;
3. Веб-приложение должно обеспечивать идентификацию и аутентификацию пользователей;
4. Веб-приложение должно предоставлять пользователю возможность восстановления пароля от своего аккаунта;
5. Веб-приложение должно предоставлять пользователю возможность управления настройками аккаунта;
6. Веб-приложение должно предоставлять пользователю возможность добавления, просмотра, редактирования и удаления своих пользовательских данных;
7. Веб-приложение должно содержать множество медицинских показателей;
8. Веб-приложение должно предоставлять пользователю возможность просмотра подробной информации о каждом медицинском показателе;
9. Веб-приложение должно обеспечивать безопасность и удобство хранения медицинских документов;
10. Веб-приложение должно предоставлять пользователю возможность самостоятельного ведения медицинской карты путем добавления, просмотра, редактирования и удаления своих медицинских данных;
11. Веб-приложение должно предоставлять пользователю возможность скачивания записей из своей медицинской карты;
12. Веб-приложение должно предоставлять пользователю возможность сортировки записей в своей медицинской карте;
13. Веб-приложение должно предоставлять поль-

зователю возможность фильтрации записей в своей медицинской карте;
14. Веб-приложение должно графически отображать динамику медицинских показателей пользователя с течением времени;
15. Веб-приложение должно предоставлять пользователю возможность экспорта динамик медицинских показателей в форматах JPG и PDF;
16. Веб-приложение должно информировать пользователя об отклонениях результатов медицинских показателей от нормы;
17. Веб-приложение должно информировать пользователя персонализированными рекомендациями по улучшению здоровья;
18. Веб-приложение должно предоставлять пользователю возможность связаться с поддержкой клиентов.

КОНЦЕПТУАЛЬНАЯ АРХИТЕКТУРА

Концептуальная архитектура представляет собой высокоуровневое описание структуры и взаимодействия компонентов системы, общей концепции работы системы.

Веб-приложения работают на основе архитектуры клиент-сервер. Это означает, что есть один или несколько серверов, которые предоставляют определенные услуги (например, хранение данных, обработка информации и т.д.), и есть клиенты, которые обращаются к этим серверам за этими услугами (рис. 3) [6].

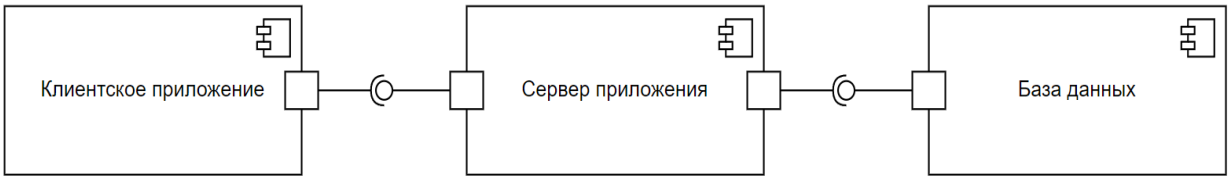

Рис. 3. Концептуальная архитектура веб-приложения
Fig. 3. Conceptual architecture of a web-application

Рассмотрим принцип работы «клиент-сервера».

Инициация запроса на клиенте: клиент, используя браузер, инициирует запрос к серверу, например, отправляя HTTP-запрос на сервер. Запрос обычно содержит информацию о том, какую услугу или какие данные хочет получить клиент.

Получение запроса: сервер принимает запрос от клиента через сетевое соединение.

Разбор запроса: сервер обрабатывает запрос, определяет тип запроса, параметры, какую именно услугу или данные хочет получить клиент в данном случае и другие детали.

Обработка запроса: на основе полученной информации сервер выполняет необходимые операции или вызывает соответствующие сервисы для обработки запроса.

Доступ к данным: при необходимости сервер может обращаться к базе данных или другим источникам данных для получения или обновления информации.

Генерация ответа: после обработки запроса сервер генерирует ответ, который будет отправлен обратно клиенту.

Отправка ответа: сервер через сетевое соединение отправляет обратно клиенту сформированный ответ в формате, который был указан в запросе.

Получение ответа на клиенте: клиент получает ответ от сервера и обрабатывает его, например, отображая данные пользователю или выполняя другие действия. Ответ может содержать информацию, которую запросил пользователь, или сообщение об ошибке, если что-то пошло не так.

ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ РАЗРАБОТКИ

В качестве среды разработки выбрана Visual Studio Code. Преимущество VS Code в том, что она не привязана к определенному языку программирования, поэтому с ее помощью в ходе реализации продукта можно создавать логику, интерфейсы, работать с базой данных и тестировать продукт [7].

Система управления базами данных обеспечивает хранение, управление, запросы, целостность, безопасность, масштабируемость и оптимизацию данных для эффективного управления информацией в базе данных, но способы выполнения этих действий будут значительно отличаться в различных СУБД, так как различные СУБД оптимизированы для разных типов данных и структур. Учитывая, что наши предполагаемые хранимые данные строго структурированы и связаны друг с другом, то реляционная СУБД, такая как Postgre SQL, должна быть хорошим выбором [8].

Для реализации backend-части веб-приложения выбран стек технологий TypeScript + NestJS [9, 10]. TypeScript является одним из самых распространенных языков программирования для fullstack-разработки веб-продуктов [11].

Выбирая один язык программирования для реализации как frontend-части, так и backend-части продукта, можно использовать общие типы данных и интерфейсы как на клиентской, так и на серверной стороне, использовать одни и те же утилиты и функции.

Это способствует повышению читаемости кода и облегчает его поддержку. Поэтому для frontend-части также выбираем TypeScript, а в качестве фреймворка – React [12].

Для построения графиков динамики медицинских показателей выбираем Chart.js - это библиотека JavaScript для создания интерактивных графиков и диаграмм на веб-страницах. Chart.js обладает интуитивным API, что упрощает создание и настройку графиков.

С ее помощью можно добавлять элементы управления, подсказки и другие интерактивные функции для улучшения пользовательского опыта. Можно настраивать цвета, шрифты, стили линий и многие другие аспекты графиков, чтобы соответствовать уникальному дизайну, разрабатываемого веб-продукта. Графики, созданные с помощью Chart.js, легко адаптируются под различные устройства и экраны [13].

РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ

Внедрим в проект Prisma - современное объектно-реляционное отображение (Object Relational Mapping, ORM) для Node.js и TypeScript, инструмент, позволяющий работать с базами данных с помощью JavaScript или TypeScript без использования SQL [14].

Инициализируем Prisma-проект с помощью команды «yarn prisma init». Выполнение данной команды приводит к генерации файла schema.prisma, определяющего подключение к БД, генератор, используемый для генерации клиента Prisma, и схему БД, а также файла .env с переменной среды окружения DATABASE_URL, значением которой является строка, используемая Prisma для подключения к БД. Определим модели в файле schema.prisma (рис. 4).

Далее сгенерируем файлы миграции SQL и напрямую запустим их в базе данных с помощью команды «yarn prisma migrate dev --name init». Автоматически запускается установка клиента Prisma в качестве производственной зависимости. Это инструмент для автоматической генерации типобезопасных запросов, который можно использовать для программного чтения и записи данных в базу данных.

Откроем приложение pgAdmin 4 и проверим сгенерированную схему базы данных (рис. 5).

schema.prisma

```
Generate
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model User {
  id        Int      @id @default(autoincrement())
  createdAt DateTime @default(now()) @map("created_at")
  updatedAt DateTime @updatedAt @map("updated_at")

  email     String @unique
  password  String

  surname     String
  name        String
  middleName  String? @map("middle_name")

  sex         String
  dateOfBirth DateTime
  avatarPath  String @default("/uploads/default-avatar.png") @map
  ("avatar_path")

  inputResult InputResult[]
}

model Analysis {
  id        Int      @id @default(autoincrement())
  createdAt DateTime @default(now()) @map("created_at")
  updatedAt DateTime @updatedAt @map("updated_at")

  name        String  @unique
  description String?

  category    Category @relation(fields: [categoryId], references:
  [id])
  categoryId  Int @map("category_id")

  unit   Unit @relation(fields: [unitId], references: [id])
  unitId Int @map("unit_id")

  inputResult     InputResult[]
  referenceValues ReferenceValues[]
}

```

Рис. 4. Файл schema.prisma
Fig. 4. File schema.prisma

Все 7 таблиц схемы базы данных связаны идентифицирующими связями. То есть первичный ключ дочерней сущности содержит внешний ключ, идущий от родительской сущности. Это гарантирует, что каждая запись в дочерней таблице связана с ровно одной записью в родительской таблице. Также это обеспечивает целостность данных, гарантируя, что данные в дочерней таблице всегда ссылаются на существующую запись в родительской таблице.

РЕАЛИЗАЦИЯ BACKEND

Предположим, что мы хотим войти в свой аккаунт. Веб-приложению нужно понять, что мы владеем этим аккаунтом. Для этого нам нужно предоставить корректные учетные данные (логин и пароль). Веб-приложение проверяет эти данные, и если они верны, то мы входим в свой аккаунт. Этот процесс называется аутентификацией.

После успешной аутентификации мы заходим на страницу «Профиль пользователя» и отправляем запрос на редактирование пользовательских данных. Веб-приложение проверяет, что запрос пришел от владельца профиля, и позволяет нам изменять данные. Процесс проверки прав на осуществление действий называется авторизацией.

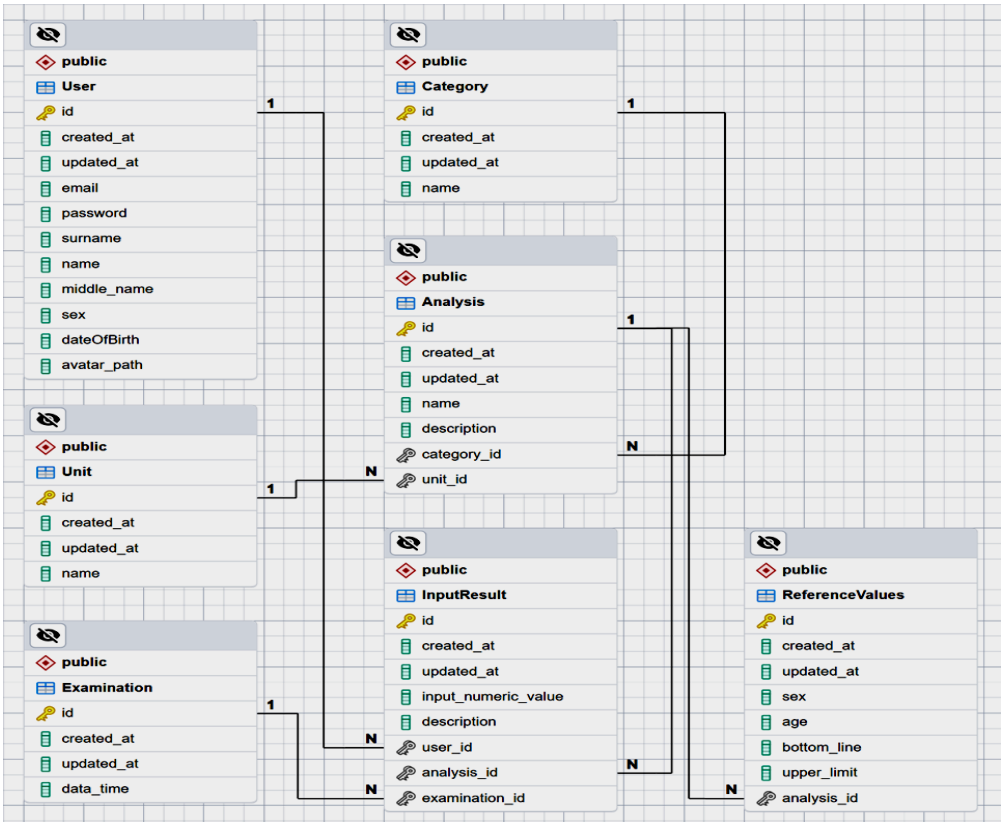

Рис. 5. Схема базы данных
Fig. 5. Database schema

Безопасность коммуникации между веб-браузером и веб-приложением обеспечивается на backend'е с помощью JWT-токенов и заключается в том, что токены генерируются и подписываются только со стороны веб-приложения. Злоумышленник не сможет подделать токен, так как не знает секретный ключ, который используется для подписи токена. Подпись токена происходит с помощью шифрования. С помощью подписи веб-приложение проверяет, что токен действительно был сгенерирован им.

Процессы аутентификации и авторизации между веб-браузером пользователя и веб-приложением с использованием JWT-токенов происходят следующим образом:

1) веб-браузер отправляет запрос веб-приложению с логином и паролем;

2) веб-приложение проверяет логин и пароль, и если они верны, то генерирует JWT-токен и отправляет его веб-браузеру. При генерации JWT-токена веб-приложение ставит подпись секретным ключом, который хранится только в веб-приложении;

3) веб-браузер сохраняет JWT-токен и отправляет его вместе с каждым запросом в веб-приложение;

4) веб-приложение проверяет JWT-токен, и если он верный, то выполняет действие от имени авторизованного пользователя [15].

РЕАЛИЗАЦИЯ FRONTEND

Основной функционал веб-приложения доступен пользователю после регистрации, авторизации и обязательного заполнения пользовательских данных.

На странице «Показатели» веб-приложение предоставляет пользователю возможность прочтения библиотеки медицинских показателей, разбитых по категориям. Нажав на конкретный лабораторный или не лабораторный показатель, можно ознакомиться с правилами его измерения и подготовкой к его замерам (рис. 6).

Веб-приложение предоставляет пользователю возможность самостоятельного ведения электронной медицинской карты, путем добавления в нее результатов анализов, лабораторных и не лабораторных исследований и заключений врачей (рис. 7). Веб-приложение предоставляет пользователю возможность просмотра, редактирования, удаления и скачивания всех добавленных обследований (рис. 8).

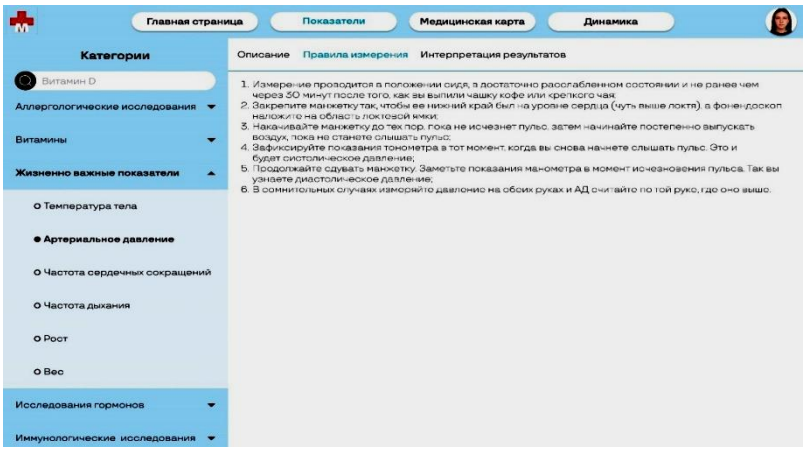

Рис. 6. Карточка показателя
Fig. 6. Summary of Indicator Information

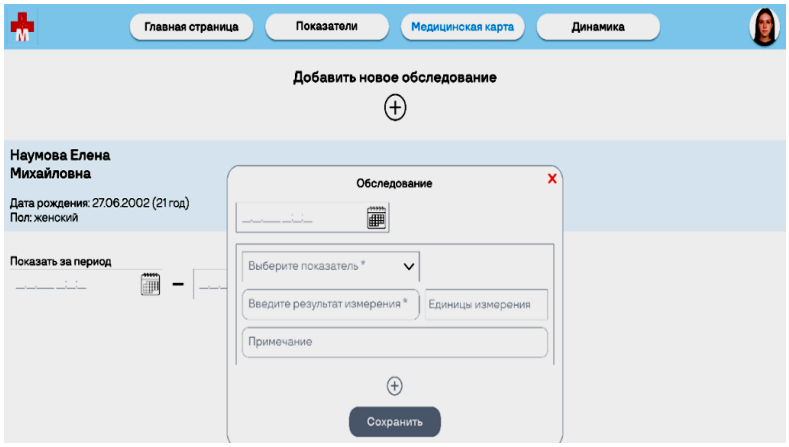

Рис. 7. Добавление нового обследования
Fig. 7. Adding a new examination

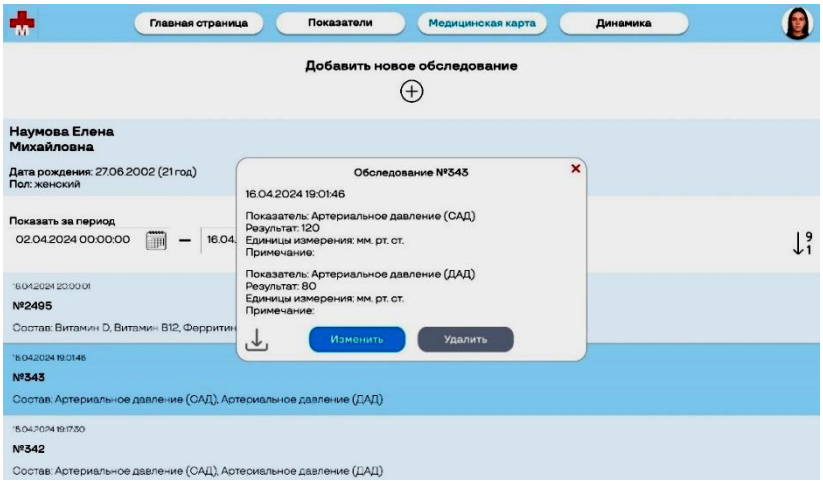

Рис. 8. Просмотр хранимого обследования
Fig. 8. Viewing a stored examination

Веб-приложение хранит все добавленные пользователем обследования, сортирует и отображает их по различным фильтрам. На рис. 9 задан фильтр на все добавленные пользователем обследования за последние две недели. Если веб-приложение не находит обследования по указанному фильтру, то пользователь получает сообщение об ошибке (рис. 10).

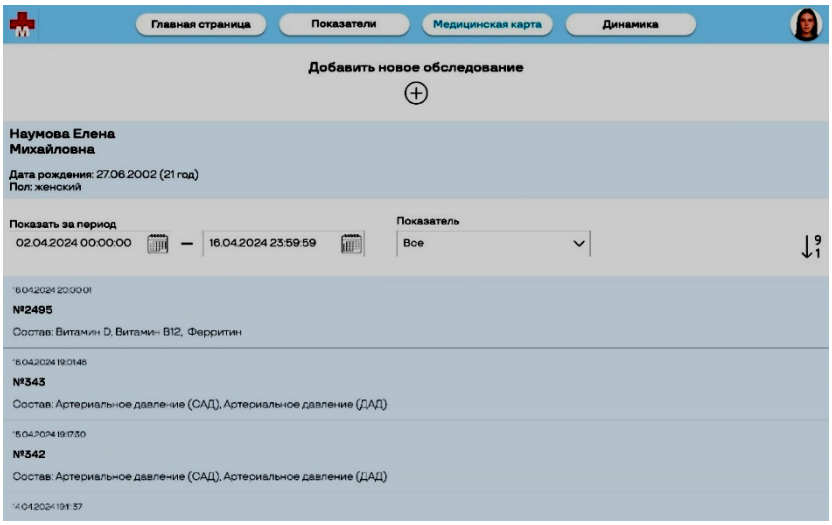

Рис. 9. Фильтрация хранимых обследований
Fig. 9. Filtering a stored examination

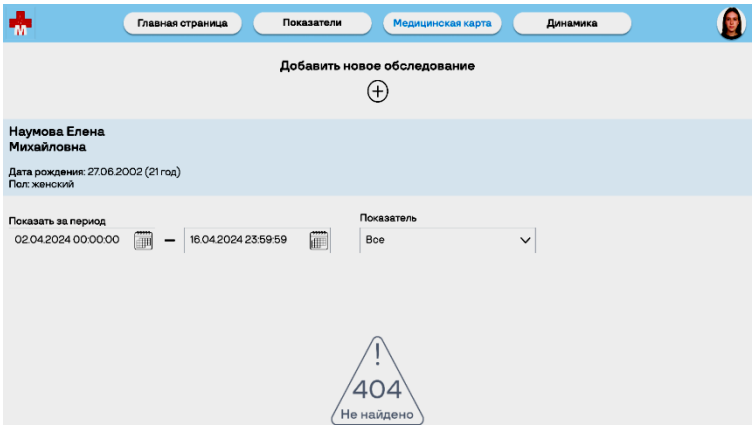

Рис. 10. Фильтрация хранимых обследований – ошибка
Fig. 10. Filtering a stored examination – error

Веб-приложение содержит функции для графического отображения динамики медицинских показателей с течением времени. Для построения графиков используется Chart.js. Её встроенная функция toBase64Image позволяет скачивать графики в виде изображения. На рис. 11 и рис. 12 заданы построения линейного и столбчатого графиков динамики артериального давления у пользователя за последние две недели. На графиках отображаются изменения показателя и область значений, соответствующих норме. Норма рассчитывается индивидуально на основании введенных пользователем данных о поле и возрасте в личном кабинете.

Рис. 11. Линейный график динамики артериального давления
Fig. 11. Linear graph of blood pressure dynamics

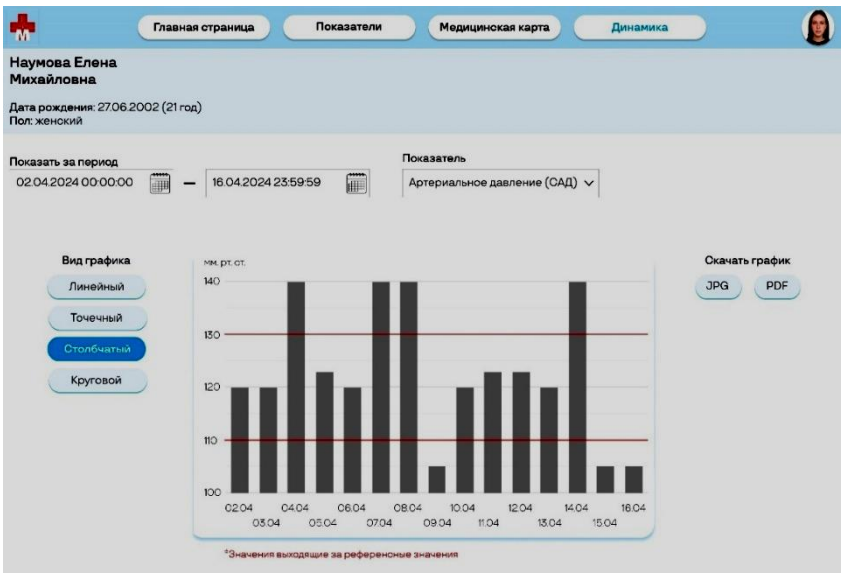

Рис. 12. Столбчатый график динамики артериального давления
Fig. 12. Columnar graph of blood pressure dynamics

ЗАКЛЮЧЕНИЕ

Разработанное веб-приложение демонстрирует потенциал цифровых технологий в улучшении процессов управления здоровьем человека. Отметим, что у самого приложения есть потенциал развития по следующим направлениям: расширение базы показателей и биомаркеров до 1500 единиц; возможность добавления в личный аккаунт профилей несовершеннолетних детей, пожилых родителей и других членов семьи; замена ручного ввода на парсинг медицинских данных из файлов форматов PDF и JPG (загруженный документ будет обрабатывать программа оцифровки и распознавания данных); возможность ведения дневников (настроения, питания, физической активности, приема лекарств и т.д.); интеграция с медицинскими и умными устройствами; интеграция с врачами частных клиник; интеграция с сервисами телемедицины для возможности проведения онлайн-консультаций с врачами.

Авторы заявляют об отсутствии конфликта интересов, требующего раскрытия в данной статье.

The authors declare the absence a conflict of interest warranting disclosure in this article.

ЛИТЕРАТУРА

1. **Ермолаев М.Б., Хомякова А.А., Белова А.Д., Серкова Ю.А.** Разработка алгоритма интеллектуальной поддержки принятия решений на базе системного подхода. *Известия высших учебных заведений. Серия: Экономика, финансы и управление производством.* 2022. № 1(51). С. 138-146. DOI 10.6060/ivecofin.2022511.594.
2. **Наумова Е.М., Ксенофонтова О.Л.** Разработка тематической социальной сети «pettypet». Сборник научных трудов вузов России "Проблемы экономики, финансов и управления производством". 2023. № 52. С. 76-80.
3. **Благой Н.А., Дуболазова Ю.А., Мокеева Т.В., Ефимов Е.А.** Оценка экономической эффективности стартапов в сфере киберспорта. *Известия высших учебных заведений. Серия: Экономика, финансы и управление производством.* 2022. № 3(53). С. 13-21. DOI 10.6060/ivecofin.2022522.610.
4. **Куленцан А.Л., Марчук Н.А.** Моделирование и прогнозирование заболеваемости населения г. Иваново. *Современные наукоемкие технологии. Региональное приложение.* 2022. № 4(72). С. 75-83. DOI 10.6060/snt.20227204.00011.

REFERENECES

1. **Ermolaev M.B., Khomyakova A.A., Belova A.D., Serkova Yu.A.** Development of an algorithm for intellectual decision support based on a systematic approach. *News of higher educational institutions. Series: Economics, Finance and Production Management.* 2022. N 1(51). P. 138-146. DOI 10.6060/ivecofin.2022511.594.
2. **Naumova E.M., Ksenofontova O.L.** Development of the thematic social network "pettypet". Collection of scientific papers of Russian universities "Problems of economics, finance and production management". 2023. N 52. P. 76-80.
3. **Blagoy N.A., Dubolazova Yu.A., Mokeeva T.V., Efimov E.A.** Assessment of the economic efficiency of startups in the field of esports. *Izvestia of higher educational institutions. Series: Economics, Finance and Production Management.* 2022. N 3(53). P. 13-21. DOI 10.6060/ivecofin.2022522.610.
4. **Kulentsan A.L., Marchuk N.A.** Modeling and forecasting morbidity among the population of Ivanovo. *Modern high-tech technologies. Regional application.* 2022. N 4(72). P. 75-83. DOI 10.6060/snt.20227204.00011.

5. **Ксенофонтова О.Л., Смирнова Н.В., Котова А.В.** Применение методов интеллектуального анализа данных в эпидемиологии. *Современные наукоемкие технологии. Региональное приложение.* 2023. № 2(74). С. 88-93. DOI 10.6060/snt.20237402.0009.
6. «Клиент-серверная архитектура в картинках» https:// habr.com/ru/articles/495698
7. **Johnson B.** Visual Studio Code: End-to-End editing and debugging tools for web developers. Indianapolis, Wiley. 2019. 192 p.
8. «Документация PostgreSQL» https://www.postgresql.org.
9. «Руководство по TypeScript» https://metanit.com/ web.
10. «Документация NestJS» https://docs.nestjs.com.
11. **Черный Б.** Профессиональный TypeScript. Разработка масштабируемых JavaScriptприложений. СПб. Питер, 2021. 352 с.
12. «Документация React» https://react.dev/learn
13. «Документация Chart.js» https://www.chartjs.org/ docs/ mast.
14. «Документация Prisma» https://www.prisma.io/docs.
15. «JWT-авторизация на сервере» https://ru.hexlet.io/ courses/go-web-development/lessons/auth/theory

5. **Ksenofontova O.L., Smirnova N.V., Kotova A.V.** Application of data mining methods in epidemiology. *Modern high-tech technologies. Regional application*. 2023. N 2(74). P. 88-93. DOI 10.6060/snt.20237402.0009.
6. "Client-server architecture in pictures" https://habr. com/ ru/articles/495698
7. **Johnson B.** Visual Studio Code: End-to-End editing and debugging tools for web developers. Indianapolis, Wiley. 2019. 192 p.
8. "PostgreSQL documentation" https://www.postgresql.org.
9. "TypeScript Manual".https://metanit.com/web/typescript
10. "NestJS documentation" https://docs.nestjs.com
11. **Black B.** Professional TypeScript. Development of scalable javascriptapplications. St. Petersburg: St. Petersburg, 2021. 52 p.
12. "React Documentation" https://react.dev/learn
13. "Chart documentation.js" https://www. chartjs.org/ docs/master
14. "Prisma Documentation" https://www.prisma.io/docs.
15. "JWT-authorization on the server" https://ru.hexlet.io /courses/ go-web-development/lessons/auth/theory

Поступила в редакцию 04.08.2024
Принята к опубликованию 12.11.2024
Received 04.08.2024
Accepted 12.11.2024