DOI 10.37882/2223–2966.2023.04.16

МЕТОД ПОСТРОЕНИЯ МОДУЛЬНОЙ МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ НА ПРИМЕРЕ РАЗРАБОТКИ ПРОГРАММЫ ПОДБОРА ЛЕКАРСТВЕННОГО ПРЕПАРАТА

THE METHOD OF BUILDING A MODULAR MEDICAL INFORMATION SYSTEM ON THE EXAMPLE OF THE DEVELOPMENT OF A DRUG SELECTION PROGRAM

I. Kiryakov
S. Molodyakov

Summary. The issues of implementing modular medical information systems are considered, that supports to connect new modules and replace old ones with new more advanced modules. The software of the medical system is developed using a micro-service architecture. The architectural scheme of the system is presented. The analysis of data exchange technologies and testing was carried out, also the technology of remote procedure call gRPC was highlighted. A template and the first modules related to the task of drug selection have been developed.

Keywords: information system, micro service architecture, data exchange, drug, API.

Киряков Иван Михайлович
Аспирант, Санкт-Петербургский политехнический университет Петра Великого
kiryakov.i@edu.spbstu.ru
Молодяков Сергей Александрович
Д.т.н., профессор, Санкт-Петербургский политехнический университет Петра Великого
molodyakov_sa@spbstu.ru

Аннотация. Рассматриваются вопросы построения модульных медицинских информационных систем, в которых обеспечивается возможность не только подключения новых, но и замена старых модулей новыми более совершенными модулями. Программное обеспечение медицинской системы разрабатывается с использованием микросервисной архитектуры. Представлена архитектурная схема системы. Проведен анализ технологий обмена данными, проведено тестирование, выделена технология удаленного вызова процедур gRPC. Разработаны шаблон и первые модули, связанные с задачей подбора лекарственного препарата.

Ключевые слова: информационная система, микросервисная архитектура, обмен данными, лекарственный препарат, API.

Введение

Современная медицина отличается тем, что в ней широко применяются информационные системы. Такими системами являются системы определения заболеваний, поиска медицинской информации, анализа большого массива данных из истории болезней и другие. Часто в медицинских информационных системах (МИС) применяются методы искусственного интеллекта (ИИ), в частности методы, связанные с использованием нейронных сетей [1]. Особенностью современных информационных технологий является непрерывное совершенствование методов и алгоритмов используемых в МИС. Так за последние два года сделан существенный шаг в развитии средств и методов определения заболеваний легких по рентгеновским снимкам. В результате возникает необходимость добавления и/или замены в МИС отдельных алгоритмов на другие. Такая замена возможна, если каждый алгоритм упаковать в отдельный модуль, а МИС будет представлять собой модульную систему.

В настоящее время наиболее современный подход для построения модульной системы связан с использованием микросервисной архитектуры для построения программного обеспечения. В этом случае приложение представляет собой совокупность слабосвязанных сервисов. Разработка каждого сервиса может вестись независимо от других, возможна замена отдельных сервисов [2, 3].

Настоящая работа является первым этапом создания модульной МИС по автоматизации рабочих процессов, совершаемых врачами в медицинских организациях. Программное обеспечение (ПО) разрабатывается с использованием микросервисной архитектуры. Разработаны первые модули, связанные с задачей подбора лекарственного препарата при помощи инструментов на основе ИИ. Рассмотрены технологии обмена данными, которые используются в МИС.

Концепция разрабатываемой системы

В мире разработки ПО преимущественно используются две популярные архитектуры разработки приложений: монолитная и микросервисная. Выбор конкретного архитектурного решения зависит от многих факторов, в частности, назначения разрабатываемой

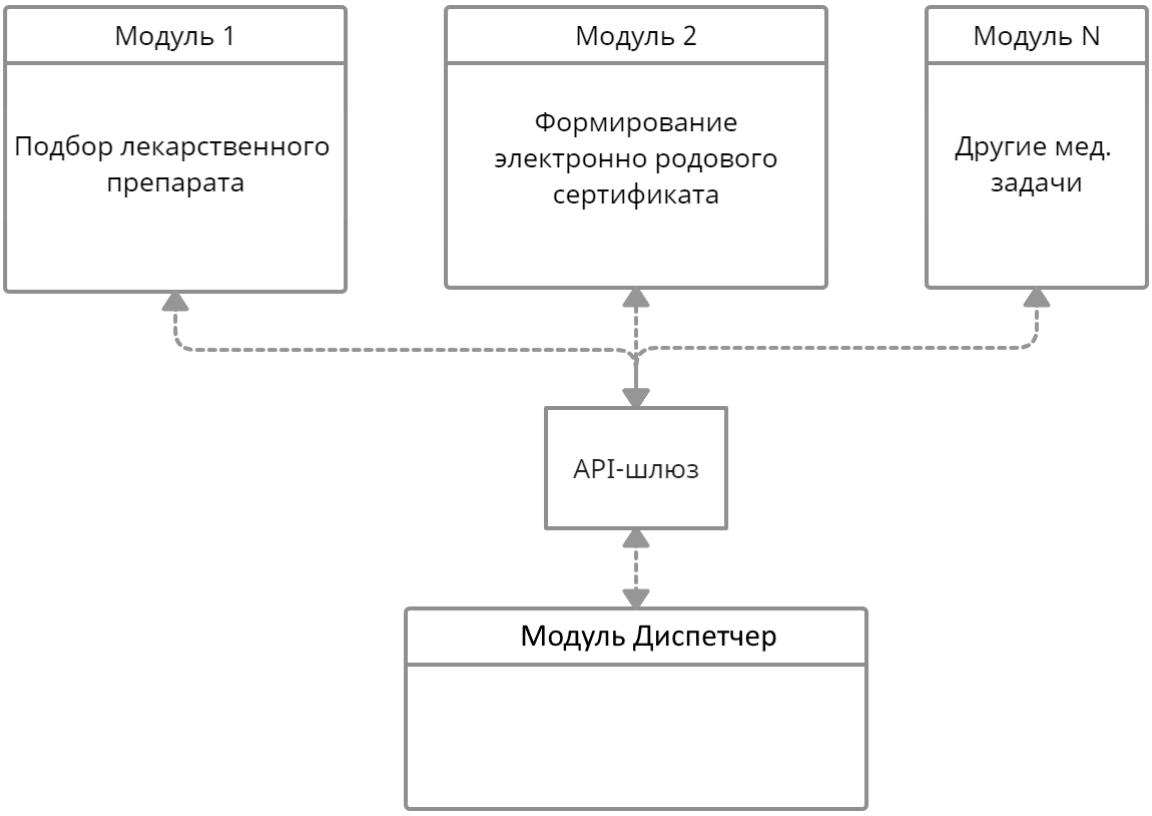

Рис. 1. Архитектура системы

системы. В рамках современных проектов, нуждающихся в легкой масштабируемости, в последнее время предпочтение отдается микросервисам [2]. Поэтому основой разработки МИС является микросервисная архитектура. Каждый сервис работает в своем процессе и взаимодействует с другими сервисами при помощи API-шлюза т.е. советующей технологии обмена данными.

На рис. 1 представлена схема медицинского модульного комплекса, который, предназначенный для автоматизации процесса медицинского обслуживания пациента.

Данный модульный комплекс будет иметь возможность динамического приседания и отсоединения нужных модулей, которые могут быть использовано как последовательно, так и на прямую по отдельности. Каждый модуль представляет самостоятельную единицу, которая в полной мере выполняет свою задачу и имеет возможность передавать нужные данные по API «следующему модулю». В качестве «следующего модуля» может выступать модуль Диспетчер, который будет поддерживать интерфейс пользователя, организовывать вызов других модулей. В данной работе излагается разработка конкретного модуля «Подбор лекарственного препарата», также подобным образом может быть преобразована в модуль разработанная нами программа обнаружения COVID-19 [4].

Анализ технологий обмена данными при микросервисной архитектуре

При проектировании приложения на микросервисной архитектуре одним из основных вопросов является взаимодействие между ее модулями. Рассмотрим известные решения и определим наилучшее решение для обмена данными между приложением-клиентом и приложением-сервером. Обмен данными между приложениями относится к элементу программного интерфейса API (Application Programming Interface), который может быть вызван или выполнен на различных уровнях абстракции в системе.

Для рассмотрения были выбраны следующие используемые технологии [5, 6]:

- протокол простого доступа к объектам (SOAP)
- передача репрезентативного состояния (REST)
- удаленный вызов процедур (gRPC)

Акцентируем внимание на таком критерии как особенности применения технологии обмена данными.

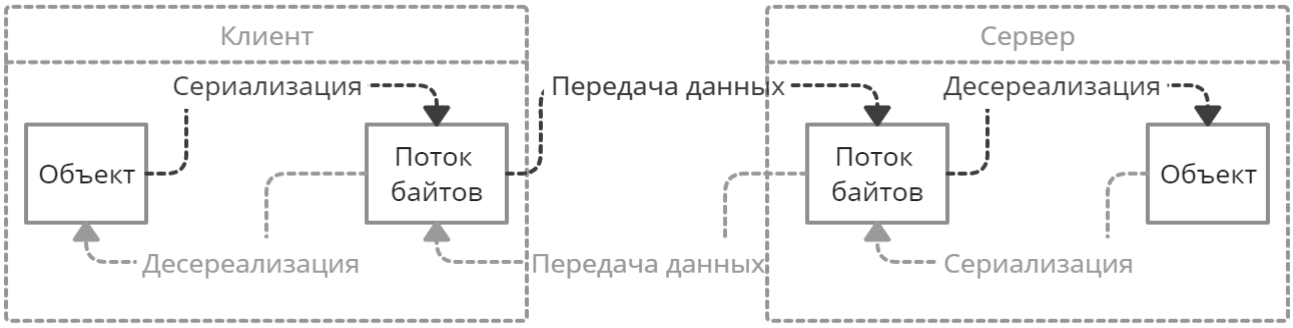

Рис. 2. Принцип обмена данными

Весь процесс обмена данными сводиться к принципу изображенный на рис. 2.

В настоящее время разработка приложений введётся на объектно-ориентированном языке программирования, например Java, C# и др. Соответственно данные, над которыми осуществляется манипуляция в алгоритмах приложения находятся в состоянии объекта, представляющий класс на языке программирования ООП. Для передачи объекта осуществляется её преобразование в соответствующий формат данных, т.е. сериализация, а при получении осуществляется её обратный процесс, т.е. десериализация. Самыми популярными форматами сериализации данных на сегодняшний день являются JSON (JavaScript Object Notation), XML (eXtensible Markup Language) и Protocol Buffers (Protobuf). Для передач данных в SOAP используется XML, для REST нет фиксированного формата передачи сообщений, а gRPC используется Protocol Buffers.

Реализация механизма обмена данными сопровождается некоторыми трудностями, которые необходимо преодолеть разработчикам и программистам, чтобы обеспечить согласованность между участниками обмена данными.

Опишем, как разрешается данная проблематика в приведённых механизмах:

SOAP использует WSDL (Web Services Description Language) — язык описания веб-сервисов и доступа к ним, основанный на языке XML. WSDL — это свод правил общения с сервисом, соблюдая которые можно осуществить взаимодействие с данным сервисом. Для языка C# имеются утилиты, способные осуществить генерацию необходимого кода на основания WSDL для осуществления обмена данными, включая этапы социализации и десериализации, но не для всех распространённых языков программирования имеются такие утилиты и весь механизм обмена придётся программисту реализовать своими силами используя библиотеки своего языка программирования.

REST это архитектурный стиль, ориентированный на использование HTTP в качестве транспортного протокола. В отличие от SOAP, для REST нет фиксированного формата передачи сообщений, также отсутствует и схема информационного обмена [5]. Для взаимодействия с сервером можно использовать XML, JSON или любой другой удобный формата. REST не имеет стандартного языка описания взаимодействия. Для того чтобы вернуть что-то нужное, требуется многократно вызывать REST. Управление данными происходит с помощью методов HTTP: GET, POST, PUT, DELETE. Данный механизм является весьма трудозатратным для реализации и несет издержки в будущем так как отсутствуют механизм позволяющий обеспечить согласованность между участниками обмена данными т.е схема информационного обмена. Основной плюс REST это высокое быстродействие, по сравнению с другими механизмами и более простое в использовании. Это связано с тем, что REST не «перегружен» различными протоколами и стандартами.

gRPC в качестве схемы информационного обмена используется proto-файл, который представляет механизм, позволяющий обеспечить согласованность между участниками обмена данными, и соответственно избавляет от издержек в разработке в будущем. gRPC предоставляет механизмы независимые от языка и платформы сериализации структурированных данных с прямой и обратной совместимостью. Данная технология дает возможность определить структуру данных, затем генерируется исходный код для записи и чтения структурированных данных, при этом могут использоваться различные языки программирования. Также можно выделить быстродействие данного механизма, форматы JSON и XML — текстовые, в отличие от Protobuf используемый в gRPC, который является бинарным и вызывает меньшую нагрузку в процессе преобразования данных.

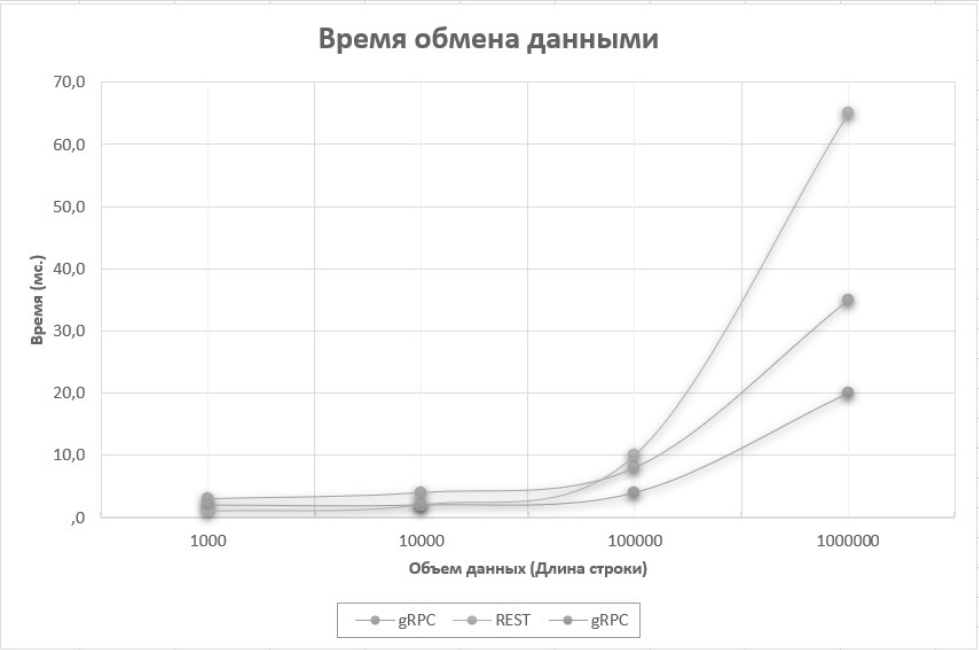

Рис. 3. График времени обмена данными.

Для оценки и сравнительного анализа быстродействия для каждой технологии обмена данными были реализованы программы, как клиентская, так и серверная часть. При реализации тестовых программ был соблюден принцип обмена данными, описанный выше и приведён на рис. 2. По окончанию обмена данными учитывалось время сериализации и десериализации данных. В качестве объекта используемый для обмена выступала структура данных, состоящая из 3-х полей и заполняемая строковыми данными различной длины в диапазоне от 1000 до 1000000 символов. Тестовым стендом служит ПК с характеристиками: ЦП — Intel Core i5–9600KF (6 ядер по 4.6 ГГц), ОЗУ — DDR4 16 Гб.

На рис. 3 представлен график зависимости времени обмена от количества передаваемых данных. Можно сделать вывод, что REST является легковесной и быстрой технологией, но проигрывает по времени при передаче больших данных по причине того, что у него отсутствует встроенной системы сериализации и десериализации данных. SOAP и gRPC обладают встроенной системой сериализации и десериализации данных, но SOAP уступает gRPC по причине того, что gRPC использует более быстрый механизм конвертации данных на основе Protobuf.

Рассмотрев представленные технологии обмена, можно сделать вывод, что наилучшим вариантом при реализации приложения, gRPC является лучшим вариантом, особенно когда реализация алгоритма обмена данными в приложении является основной задачей, а для реализации её требуется надёжное и нетрудоёмкое в применении технология.

Описание разрабатываемой системы

Рассмотрим первый модуль, который планируется реализовать и внедрить в данную систему, предназначенный для подбора лекарственного препарата (ЛП) под названием «Подбор лекарственного препарата». В основе модуля лежать нейросетевые алгоритмы распознавания сведений.

Прием к врачу начинается с получения анамнеза, т.е. врач осуществляет процесс получения от пациента совокупности сведений, путем расспроса и медицинского обследования. Полученную информацию от пациента врач вносит в электронную медицинскую карту (ЭМК).

Модуль «Подбор лекарственного препарата» собрав всю доступную медицинскую информацию о пациенте из ЭМК, осуществляет анализ полученных данных и подбор ЛП. Проанализировав данные, распознав симптомы модуль может предложить несколько ЛП по приоритету. Результаты поиска подходящих ЛП могут быть основаны на огромном объеме информации, которые содержат даже самые малоизвестные нюансы по заболеванию. Модуль может анализировать симптомы и не просто подбирать ЛП по диагностированной болезнь, но и выбирает максимально безопасный и эффективный ЛП в зависимости от особенностей пациен-

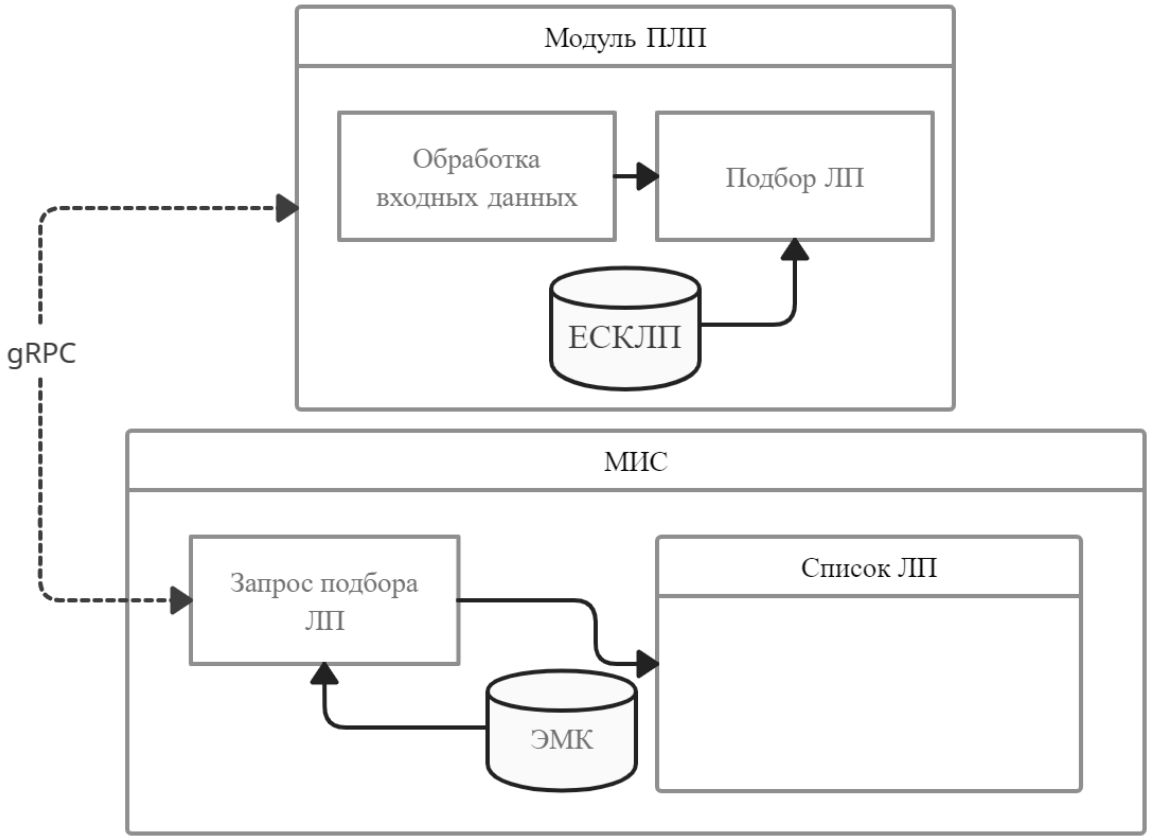

Рис. 4. Архитектура системы с использованием модуля ПЛП.

та. Список ЛП сформированные модулем, можно будет трактовать как второе мнение по результатам анализа.

На рис. 4 представлена архитектура разрабатываемой системы в котором отображена концепция подключения подбора лекарственно препарата (ПЛП) к медицинской информационная системе и использования его для расширения функциональных возможностей МИС.

В системе используется единый структурированный справочник лекарственных препаратов (ЕСКЛП) [6]. Данный справочник создан Минздравом и используется в процессе мониторинга движения лекарственных препаратов. Данный справочник рекомендуем для применения во всех МИС используемых в медицинских организация РФ. Данные о ЛП, такие как наименования, лекарственная форма, дозировка и т.д. заданные в данном справочнике приняты как стандарт данных используемых для передачи информации о пациенте в другие системы Минздрава РФ.

Модуль представляет из себя микросервис, реализованный по стандартам, позволяющий подключать его к МИС. Архитектуру системы на рис. 4 можно представить как шаблон проектирования, необходимый для расширения функционала МИС, в основе которого лежат следующие составляющие:

- МИС (с диспетчером);
- Модуль;
- API-шлюз.

Модуль, представляет из себя микросервис, выполняющий конкретную медицинскую задачу с применением ИИ для МИС. В данной работе микросервис реализуется на платформе ASP.NET Core.

API-шлюз, в данной системе это средство универсального подключения модуля к МИС. Он обеспечивает коммуникацию между клиентом (МИС) и сервисом (Модуль). В основе которого лежит фреймворк gRPC, который использует протокол RPC (Remote Procedure Call) для обмена сообщениями между клиентом и сервером.

gRPC использует подход contract-first для построения API. То есть некоторый контракт, которому сервер и клиент должны следовать. Этот контракт описывается с помощью файлов с расширением.proto.

Proto-файлы выполняют две основные задачи:

```
syntax = "proto3";
option csharp_namespace = "GrpcMedicationService";
package packMedicationService;

//Определение сервиса
service MedicationService
{
  // Метод получения информации о ЛП
  rpc GetMedicationName (MedicationInfoRequest) returns (MedicationInfoReply);
}

// Отправляемое сообщение
message MedicationInfoRequest { string DiagnosInfo = 1; }

// Получаемый ответ
message MedicationInfoReply
{
    string MedicationName = 1;
    string MedicationCode = 2;
    string MedicationInfo = 3;
}
```

Рис. 5. Proto-файл микросервиса «модуль ПЛП»

```
// Создание канала для обмена сообщениями с сервером
GrpcChannel channel = GrpcChannel.ForAddress("https://localhost:7147");

// Создаем клиента
var client = new MedicationService.MedicationServiceClient(channel);

// Обмен сообщениями с сервером
MedicationInfoReply reply = await client.GetMedicationNameAsync(new MedicationInfoRequest()
{
    DiagnosInfo = diagnosInfo
});
```

Рис. 6. Взаимодействие клиента с микросервисом «модуль ПЛП»

- Описание сервиса gRPC;
- Описание формата сообщений, которыми обмениваются клиент и сервер.

Для описания сервиса gPRC и сообщений proto-файле используется специальный синтаксис proto.

При помощи gRPC обеспечивается легкое подключения модуля к МИС, следящим образом. Например, возьмем файл «MedicationService.proto», который был написан для микросервиса «модуль ПЛП», представлен на рис.5.

Для подключения микросервиса «модуль ПЛП» к МИС, разработчику МИС, необходимо сделать 4 шага:

1. Добавить файл «MedicationService.proto» в состав проекта МИС;
2. Указать в конфигурационном файле проекта МИС путь к proto-файлу: *<Protobuf Include=»Protos\MedicationService.proto» GrpcServices=»Client» />*;
3. Установить Nuget пакет Grpc.Tools, соответствующей версии для проекта МИС. Grpc.Tools содержит инструменты для поддержки protobuf-файлов и осуществит автоматически генерацию необходимых классов по файлу «MedicationService.proto» на языке программирования проекта МИС. В составе сгенерированных классов присутствуют методы взаимодействия, с классами запроса и ответа, которые строго соответствуют контракту;
4. Реализовать взаимодействие с микросервисом на примере рис. 6, с использованием сгенерированных классов. На рис. 6 предоставлен код, в котором создаётся клиент для взаимодействия микросервисом «модуль ПЛП», вызов метода сервиса «*GetMedicationName*», в котором передаётся запрос «*MedicationInfoRequest*» с инфор-

мацией о диагнозе пациента, на который получает ответ «*MedicationInfoReply*» с информацией о подобранном лекарственном препарате.

Подобных proto-файлов как «MedicationService.proto» МИС может подключить несколько. Таким образом, МИС может внедрить в свою систему несколько необходимых модулей, расширить функционал и всего за пару шагов обновить МИС.

Заключение

В данной работе был рассмотрен метод построения модульных медицинских информационных систем, связанный с использованием микросервисной архитектуры. Он предполагает постепенное наращивание возможностей информационных систем путем добавления новых модулей и замены старых модулей новыми более совершенными. Причем в качестве модулей можно использовать программы, в которых используются методы искусственного интеллекта.

Проведен анализ технологий обмена данными SOAP, REST, gRPC. После проведенного тестирования, выделена технология удаленного вызова процедур gRPC, которая используется в системе. Представлена архитектурная схема системы с модулем подбора лекарственного препарата. Рассмотрен механизм работы с модулем. Определен шаблон и механизм взаимодействия с модулями.

Будущие направления разработок и исследований связаны с разработкой интерфейса-диспетчера для работы с врачами, расширением возможностей модуля подбора лекарственного препарата, созданием других модулей.

ЛИТЕРАТУРА

1. Мазаев В.П., Рязанова С.В. Основные направления развития искусственного интеллекта в медицине. // Научное обозрение. Медицинские науки. 2020.— № 5.— С.33–40. doi: 10.17513/srms.1141.
2. А. Э. Порфильева, Р.Ф. Шайхутдинов, Нуриева Г.А. Эффективная разработка приложений при микросервисной архитектуре // Электронные библиотеки.— 2018.— Т. 21.— № 3–4.— С. 357–368.— EDN XVASHB.
3. Chris Richardson (2017). Pattern: Microservice Architecture. URL: http://microservices.io/patterns/microservices.html
4. Думаев Р.И., Киряков И.М., Молодяков С.А. Особенности предобработки и сегментации изображений в задаче обнаружения COVID-19 по рентгеновским снимкам // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и Технические Науки.—2022.—№ 09.— С. 88–95. doi: 10.37882/2223–2966.2022.09.08
5. Различия REST и SOAP [Электронный ресурс].— URL: https://habr.com/ru/post/483204/ (19.01.2023)
6. Авельцов, Д.О. Применение протокола сериализации структурированных данных Protobuf в микросервисной архитектуре / Д.О. Авельцов // Проблемы автоматики и управления.— 2022.— № 3 (45).— С. 185–196.— EDN SJWCHJ.
7. Единый структурированный справочник лекарственных препаратов (ЕСКЛП) [Электронный ресурс].— URL: esklp.egisz.rosminzdrav.ru

Журнал «Современная наука: актуальные проблемы теории и практики»