Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/05 Современные интеллектуальные программно-аппаратные комплексы**

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине «Современные технологии разработки программного обеспечения»

НА ТЕМУ:

Распределенная система эволюционного моделирования

Студент	ИУ6-33М		В.Д. Шульман
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Руководитель			М.В. Фетисов
		(Подпись, дата)	(И.О. Фамилия)

РЕФЕРАТ

Расчётно-пояснительная записка с. 30, рис. 15, табл. 5, источников 8, приложений 2.

РАСПРЕДЕЛЕННАЯ СИСТЕМА, МИКРОСЕРВИС, ОБНАРУЖЕНИЕ СЕРВИСОВ, МОНИТОРИНГ, БАЗА ДАННЫХ, РЕПЛИКАЦИЯ ШАРДИРОВАНИЕ, НЕПРЕРЫВНАЯ ИНТЕГРАЦИЯ, ЭВОЛЮЦИОННОЕ МОДЕЛИРОВАНИЕ, АГЕНТ, ГЕН, ПОКОЛЕНИЕ, ЦЕЛЕВАЯ ФУНКЦИЯ

Объектом проектирования и разработки является распределенная система эволюционного моделирования.

Целью работы является проведение анализа предметной области, выделение основных бизнес-процессов и сущностей, проектирование и реализация распределенной информационной подсистемы, отладка и тестирование, разработка инструкции пользователя.

Результатом работы является спроектированный и реализованный программный продукт «Распределенная система эволюционного моделирования».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **6**

1 Анализ предметной области эволюционного моделирования. Формирования перечня решаемых задач **7**

1.1 Метод эволюционного моделирования 7
1.2 Генетический алгоритм 8
1.3 Генетического алгоритма для решения задач оптимизации 10
1.4 Муравьиный алгоритм и задача коммивояжера 12

2 Проектирование распределенной системы эволюционного моделирования **13**

2.1 Анализ требований к распределенной системе 13
2.2 Структура распределенной информационной системы 15
2.3 Проектирования алгоритмов функционирования 18
2.4 Проектирование базы данных 21

3 Реализация программных компонентов системы. Отладка и тестирование **25**

3.1 Выбор средств программной реализации 25
3.2 Реализация БД. Разработка скриптов создания объектов 26
3.3 Реализация интерфейсов. Разработка API 28
3.4 Проработка репликации и шардирования 29

4 Инструкция системного программиста **30**

4.1 Подготовка программного окружения 30
4.2 Первичная конфигурация 30
4.3 Запуск 31
4.4 Администрирование 31

ЗАКЛЮЧЕНИЕ **32**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **33**

ПРИЛОЖЕНИЕ А **34**

ПРИЛОЖЕНИЕ Б **40**

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Распределенная информационная система – совокупность взаимодействующих программных подсистем, каждая из которых может рассматриваться как программный модуль, исполняемый в рамках отдельного процесса

Микросервис - это небольшой программный компонент, решающий одну определенную задачу

Предметная область базы данных – часть реального мира, о которой база данных хранит, собирает и анализирует информацию

Система управления базами данных – совокупность программных средств, обеспечивающих управление, создание и использование БД

Go – императивный язык программирования

SQL – декларативный язык программирования

PostgreSQL – свободная объектно-реляционная СУБД

Шардирование – метод горизонтального масштабирования записи

Репликация - метод горизонтального масштабирования чтения

Эволюционное моделирование – подход, в основе которого лежат принципы и понятийный аппарат, заимствованный из популяционной генетики, а также ряд компьютерных методов

Генетический алгоритм – метод эволюционного моделирования, построенный на принципах естественного отбора и генетической рекомбинации

Целевая функция – это функция нескольких переменных, подлежащая оптимизации в целях решения некой задачи

Агент – эволюционная единица, представляющая одно конкретное решение задачи из всего множества возможных

Ген – элементарная структурная единица агента, подвергающаяся модификации в ходе процесса эволюционного моделирования

Поколение – совокупность агентов на конкретной итерации работы генетического алгоритма

Генотип – пространство поиска решений

Фенотип – совокупность найденных решений

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

БД – база данных

СУБД – система управления базами данных

ПП – программный продукт

ИС – информационная система

ИИС – интеллектуальная информационная система

ЦПУ – центральное процессорное устройство

ПЗУ – постоянно запоминающее устройство

ОЗУ – оперативно запоминающее устройство

ЭМ – эволюционное моделирование

ЭП – эволюционный процесс

ГА – генетический алгоритм

МА – муравьиный алгоритм

ГК – генетический код

ЦФ – целевая функция

ВВЕДЕНИЕ

Представленная работа является результатом анализа предметной области эволюционного моделирования. Сама же спроектированная и разработанная информационная система предназначена для обработки и хранения данных процессов эволюционного моделирования с учётом специфик рассмотренной предметной области.

Анализ предметной области, приводимый в данной расчётно-пояснительной записке, является информативной выжимкой из научно-исследовательской работы. Результатом анализа является набор бизнес-процессов, сущностей и связей между ними.

В процессе проектирования большое внимание уделялось не только правильному отображению сущностей предметной области в схеме базы данных (БД), но и проектно-техническим моментам, влияющим на возможность горизонтального масштабирования всей системы, как на запись (шардинг), так и на чтение (репликация).

Подразумевается, что разработанная подсистема должна войти в состав будущего дипломного проекта магистра. Саму же разработанную подсистему можно поделить на 2 части – на бизнес-логику, представленную микросервисами Golang и на множество баз данных под управлением систем управления базами данных (СУБД) PostgreSQL.

Для координации микросервисов используется реестр служб Consul.

Для сбора метрик используется системы мониторинга Prometheus.

1 Анализ предметной области эволюционного моделирования. Формирования перечня решаемых задач

1.1 Метод эволюционного моделирования

Эволюционное моделирование (ЭМ) – это группа эвристических методов, которые заимствуют принципы и понятийный аппарат у популяционной генетики (рис. 1). Эти методы позволяют осуществлять поиск решения для задач оптимизации [1].

Метод ЭМ является эвристическим. То есть он применяется для решения тех задач, которые по какой-то причине невозможно решить аналитически [2]. В этом плане метод ЭМ наравне с другими методами приближенного решения входит в класс «мягких вычислений» [3].

Можно сказать, что ЭМ – это подход, возникающий на стыке двух наук – эволюционной биологии и информатики (рис. 1).

Рисунок 1 – Эволюционное моделирование

Подход ЭМ может использоваться в:

– системах технического проектирования;

– системах автоматического управления и регулирования;

– коммуникационных и транспортных системах;

– и т. д. и т. п.

1.2 Генетический алгоритм

В ЭМ применяются различные методы, но наиболее часто используемый – это генетический алгоритм (ГА) (рис. 2). В рамках курсовой работы при реализации системы использовался именно генетический алгоритм, так что вся дальнейшая речь в РПЗ будет идти применительно именно к этому подходу ЭМ.

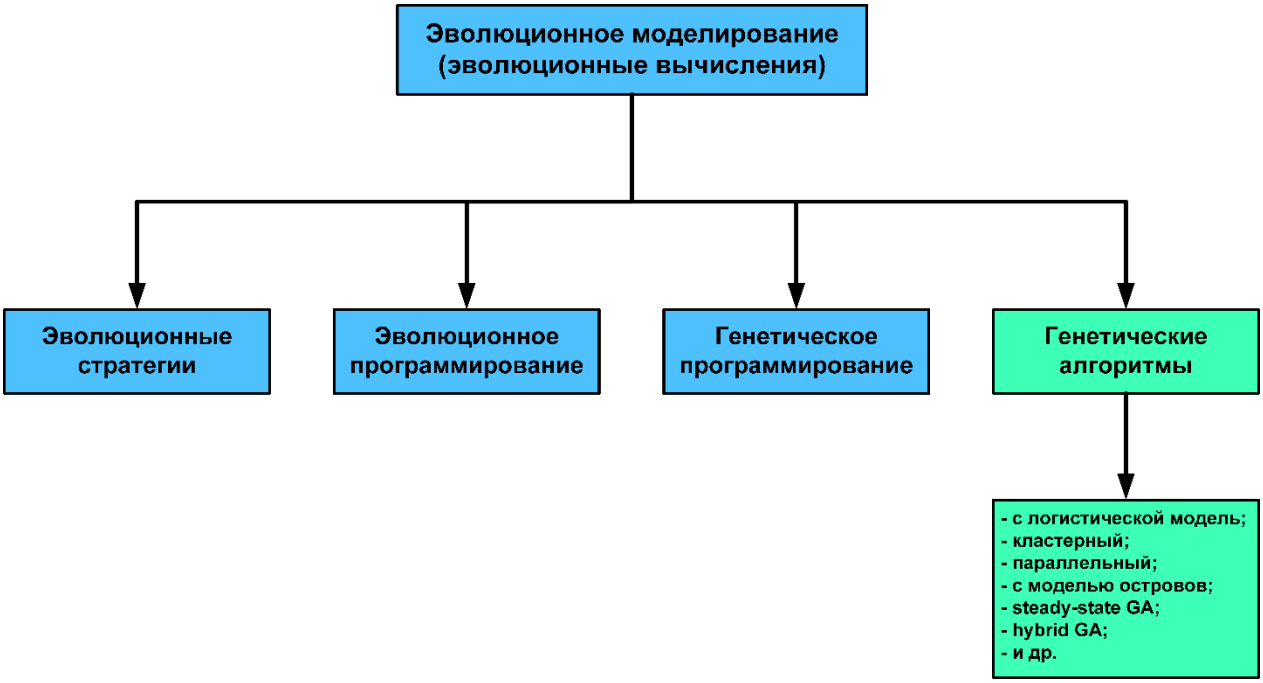

Рисунок 2 – Классификация методов эволюционного моделирования

Специфика работы ГА позволяет накапливать и использовать знания об исследованном пространстве поиска и, следовательно, позволяет проявлять способность к самообучению, что свойственно для интеллектуальных систем [].

В самом же процессе поиска в ГА используется значение целевой функции, а не её приращение (градиент), как традиционно принято в методах машинного обучения.

Таким образом, ГА обладает преимуществами:

– независимость от вида функций;

– независимость от области определения и типов переменных;

– применимость к широкому кругу задач без нужды в модификации.

Далее будет использоваться терминологий из таблицы 1.

Таблица 1 – Соответствие терминов эволюционной и мат. моделей

Эволюционная модель	**Математическая модель**
Агент	Решение, объект, строка, последовательность
Ген	Переменная, параметр, характеристика, признак
Fitness-функция	Целевая функция
Популяция	Множество решений
Поколение	Итерация работы эволюционного алгоритма

Последовательность работы ГА представлена на рисунке .

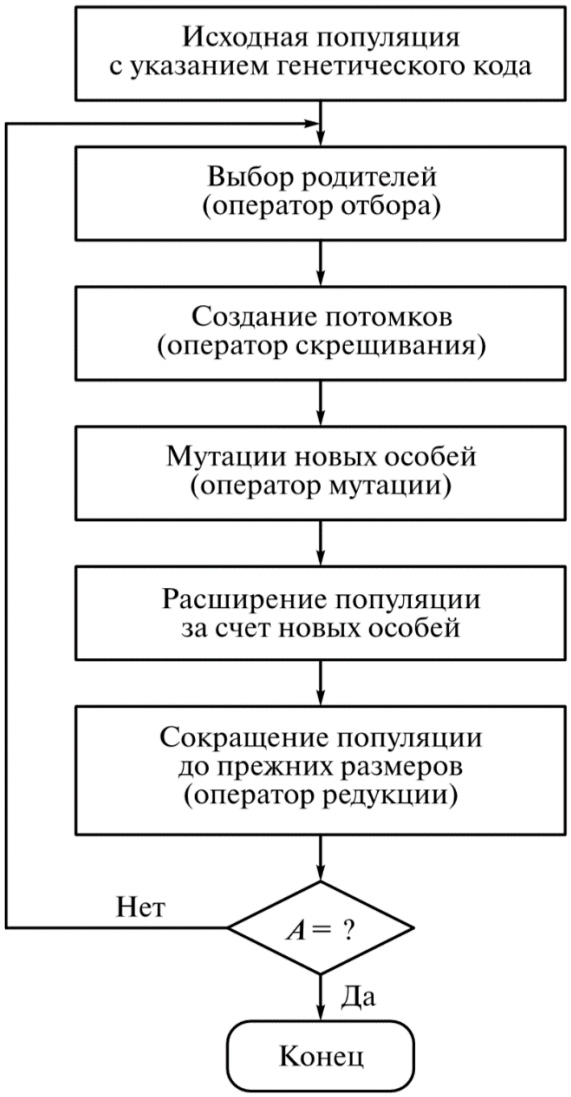

Рисунок 3 – Последовательность работы генетического алгоритма

1.3 Генетического алгоритма для решения задач оптимизации

Наиболее распространенное применение ГА – это решение оптимизационных задач. Чаще всего ГА не решает оптимизационную задачу напрямую, а используется в качестве алгоритма оптимизации (настройки) уже существующего специализированного эвристического алгоритма (рис.)

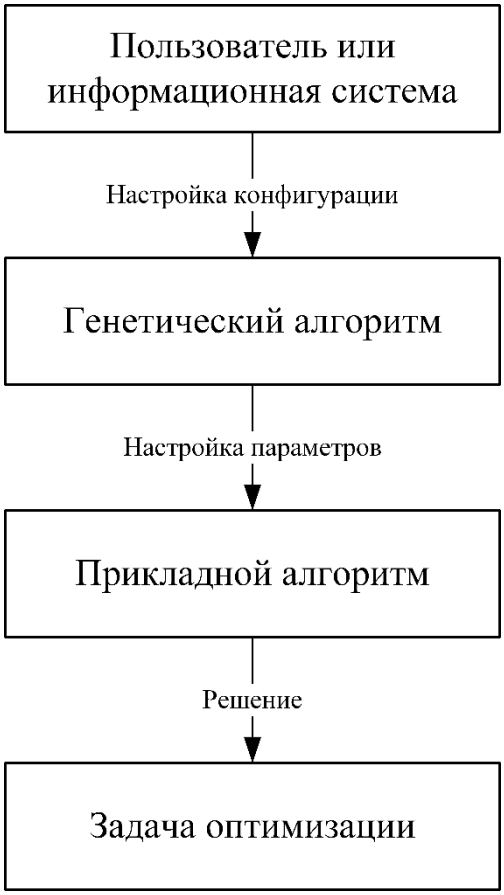

Рисунок 4 – Иерархическая структура работы алгоритмов

Чтобы использование такого подхода было оправдано, необходимо, чтобы оптимизируемый алгоритм обладал достаточным перечнем параметром (обычно хватает от 3-х и более) для того, чтобы было что варьировать средствами ГА с целью избегания полного перебора.

В данной курсовой работы для демонстрации состоятельности такого иерархического подхода предлагается в качестве решаемой оптимизационной задач использовать графовый полносвязный вариант задачи Коммивояжера, в качестве прикладного алгоритма использовать муравьиный алгоритм, а в качестве ГА использовать классический ГА и 3 операторами.

Структура классического 3-операторго ГА представлена на рисунке . Из него видно, что ГА состоит из нескольких основных компонентов: набора операторов, последовательно применяемых к каждому агенту в поколении, популяцию агентов и блок с оценкой приспособленности агентов.

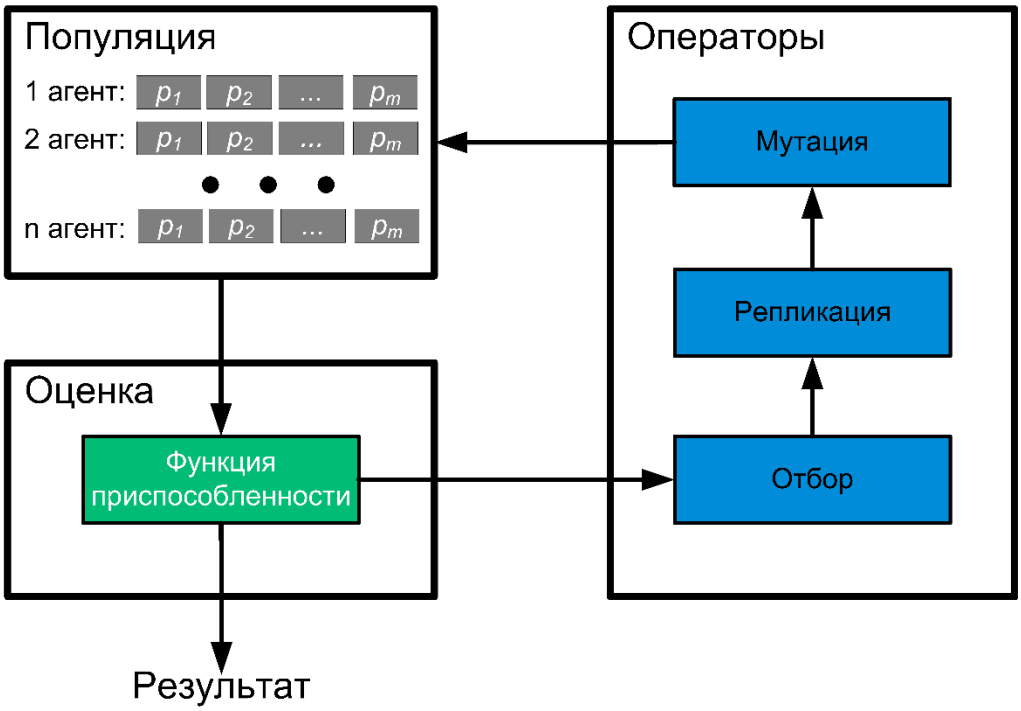

Рисунок 5 – Структура генетического алгоритма

Результатом работы ГА является некоторая наиболее лучшая конфигурация агента, позволившая достигнуть наилучшего результата во время решения задачи оптимизации. Как правило, ГА используется там, где невозможно использовать алгоритмы с полиномиальной сложностью. ГА позволяют достичь некоторого неплохого результата, однако не дают гарантии того, что найденное решение является наилучшим.

ГА весьма гибки, каждый из компонентов поддается настройке. Существует множество вариантов стратегий отбора, репликации и мутаций агентов. В данном курсом проекте предлагается использовать простые реализации этих операторов: табличный отбор, репликация на основе случайного выбора агента, мутация случайного гена с заранее заданной вероятностью.

1.4 Муравьиный алгоритм и задача коммивояжера

В данной курсовой работе в качестве демонстрации предлагается реализовать решение задачи коммивояжера (нахождения цикла минимальной длины) (рис.) с помощью муравьиного алгоритма (рис.).

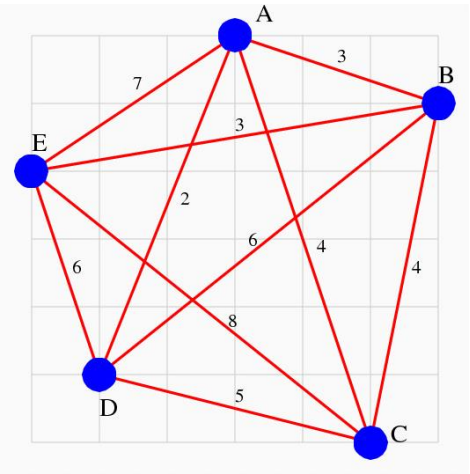

Рисунок 6 – Графовое представление задачи коммивояжера

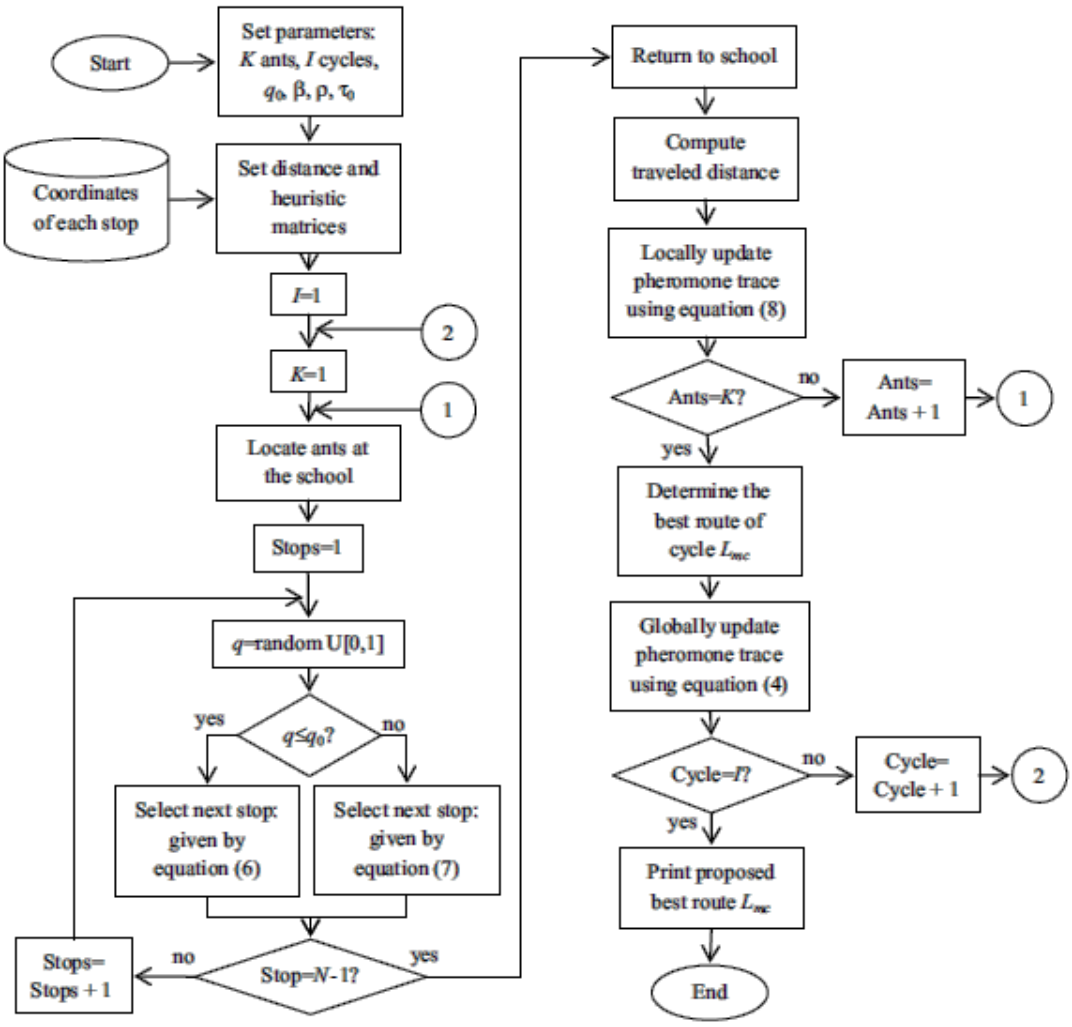

Рисунок 7 – Муравьиный алгоритм

2 Проектирование распределенной системы эволюционного моделирования

2.1 Анализ требований к распределенной системе

На основании требований к функциональности системы можно составить диаграмму вариантов использования для пользователя (рис.).

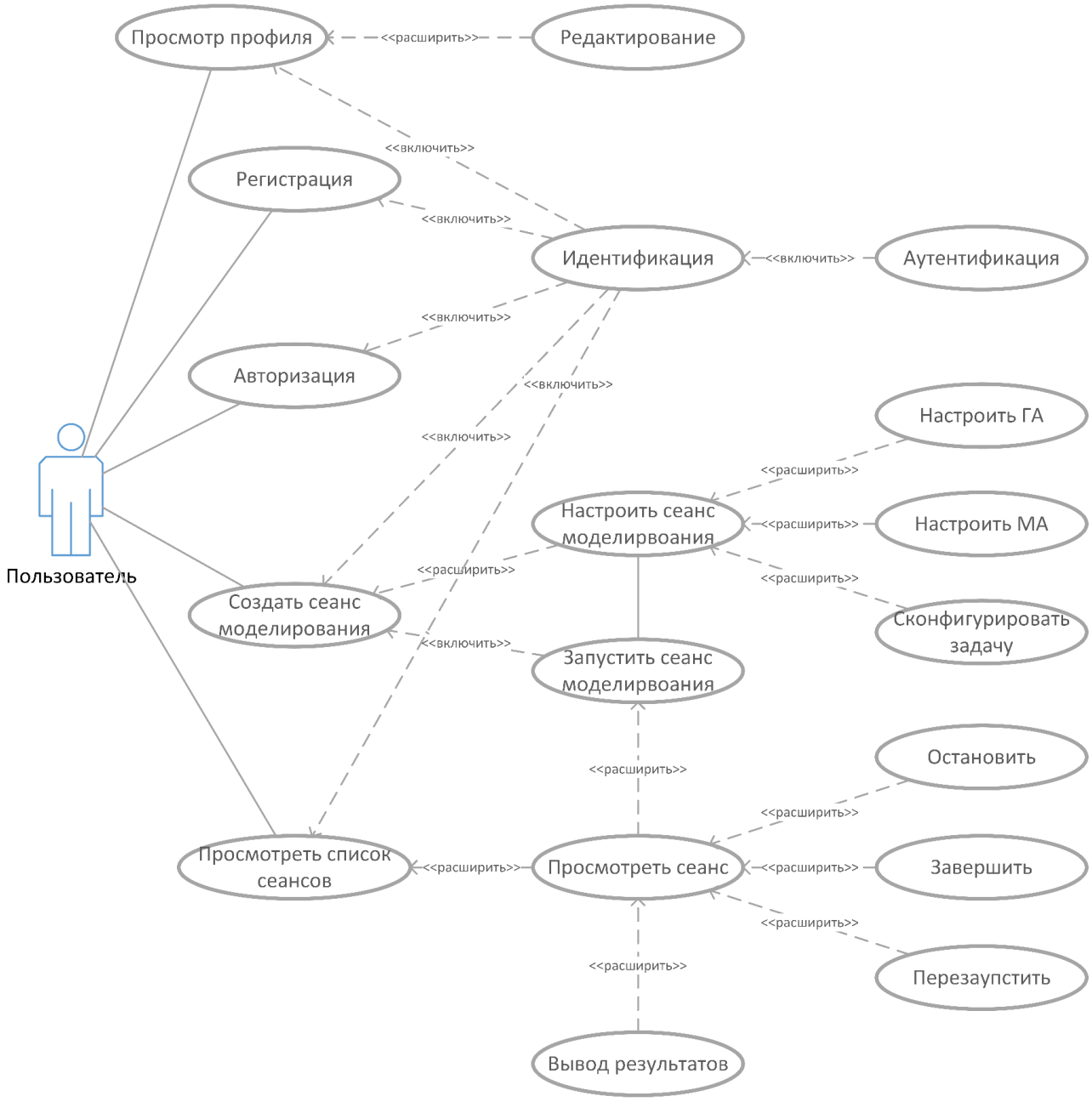

Рисунок 8 – Диаграмма вариантов использования

Из диаграммы видно, что основные функции, которые должна выполнять система для пользователя – это регистрация, авторизация, создание сеансов моделирования и просмотр результатов их работы.

Для нового пользователя процесс получения необходимых ему результатов представлен в диаграммы последовательности (рис.).

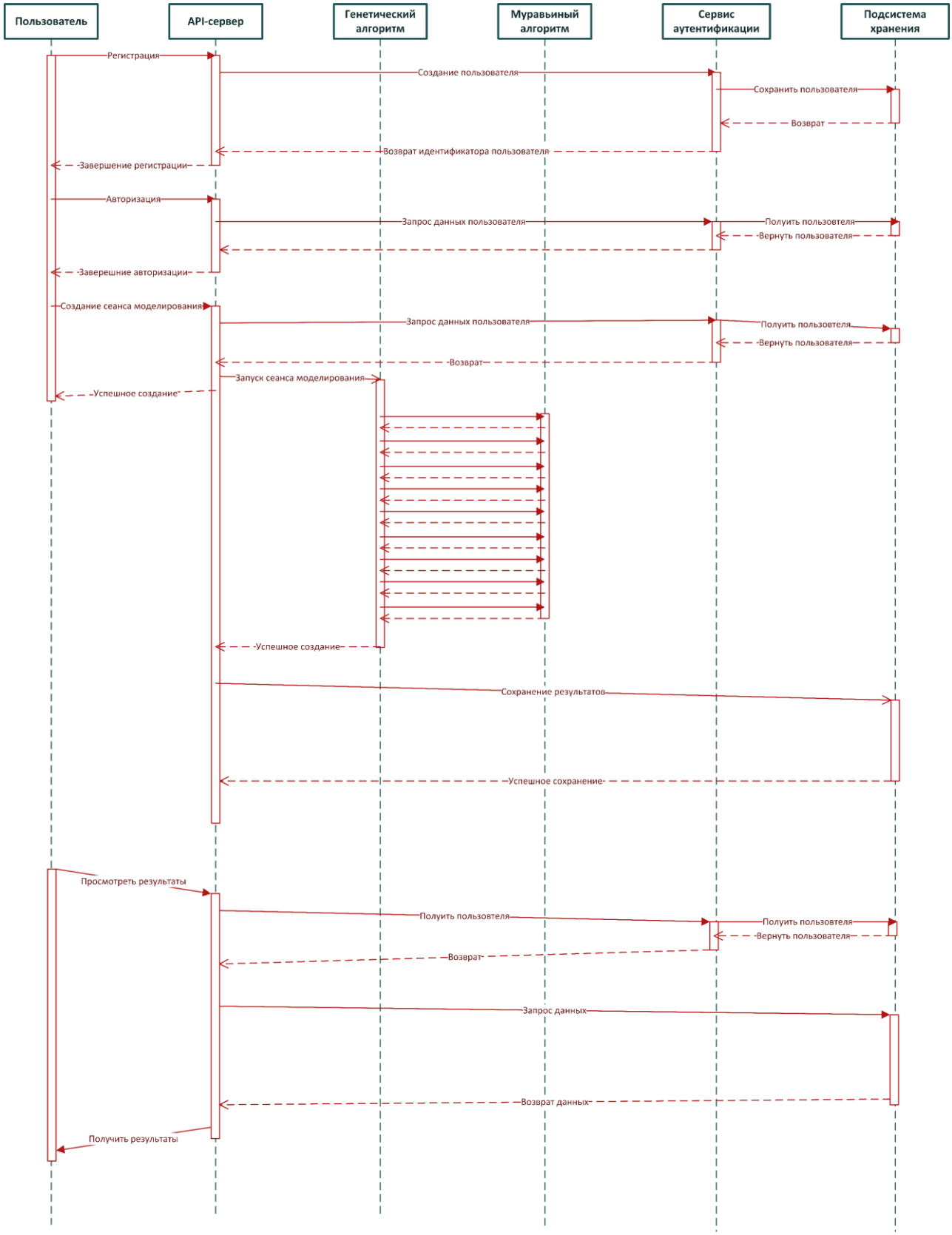

Рисунок 9 – Диаграмма последовательности

2.2 Структура распределенной информационной системы

Если не вдаваться в подробности взаимосвязей внутренних компонентов проектируемого программного продукта, то описать его структуру и способ взаимодействия с внешним миров можно с помощью рисунка .

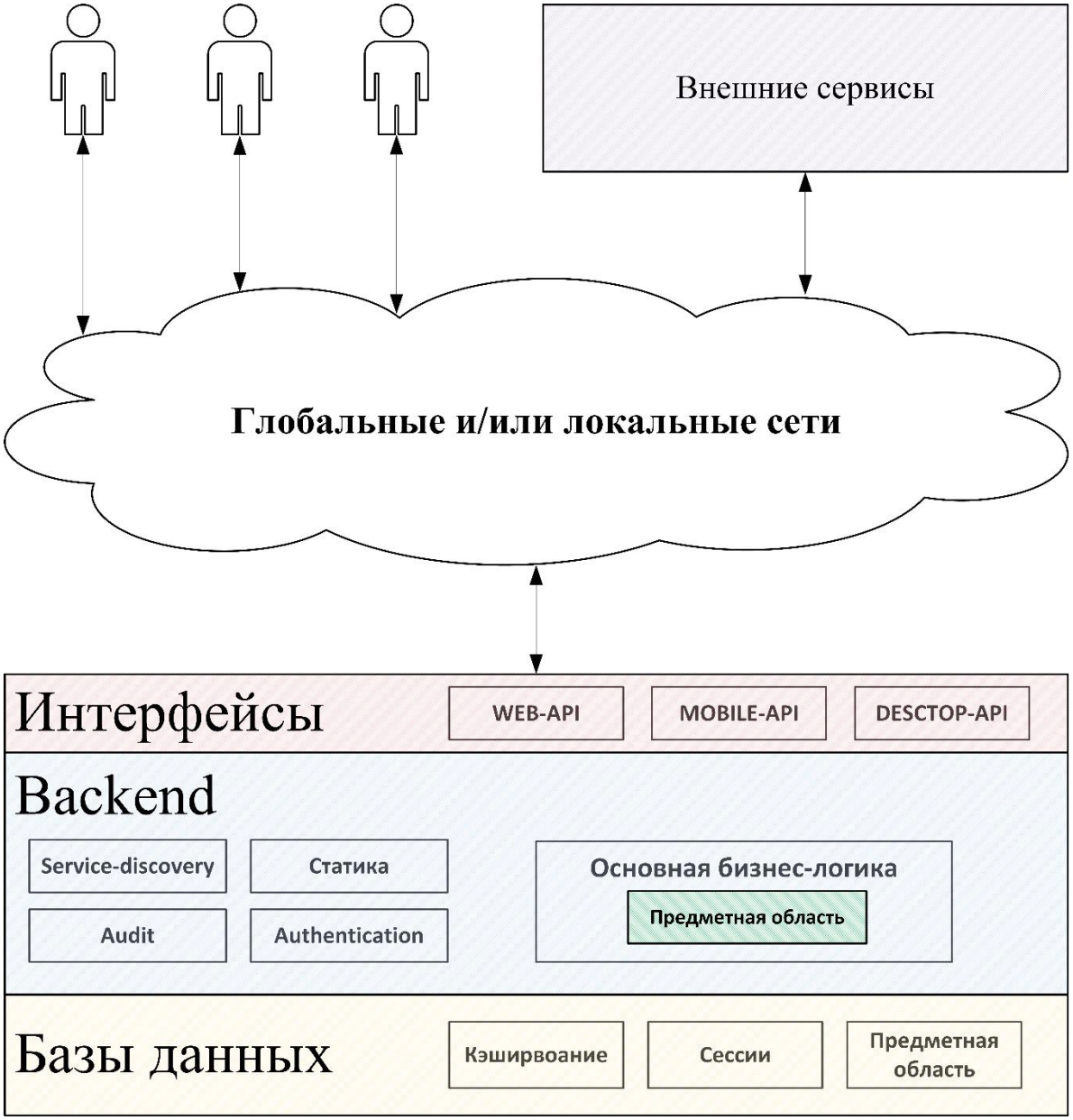

Рисунок 10 – Упрощенная структура информационной системы

Как видно из рисунка выше, вся бизнес-логика, связанная с особенностями предметной области, инкапсулирована. Она не должна влиять на более общие процессы, как аудит, service-discovery или аутентификация.

В действительности, в рамках курсового проекта реализуются не все компоненты. Это происходит по 2 причинам: либо программный компонент уже имеется в виде свободно распространяемого готового решения (например, для

service-discovery есть Consul), либо он является по своей сути дубликатом-аналогом уже имеющейся функциональности (для взаимодействия с системой достаточно одного desktop, web или mobile интерфейса).

В соответствии с требованиями ТЗ, все программный компоненты проектируются и реализуются как независимые самостоятельные программные компоненты, взаимодействующие друг с другом по сетевым протоколам (по большей части используя HTTP).

Каждый самописный программный компонент системы (микросервис) проектируется по многоуровневой архитектуре (рис.).

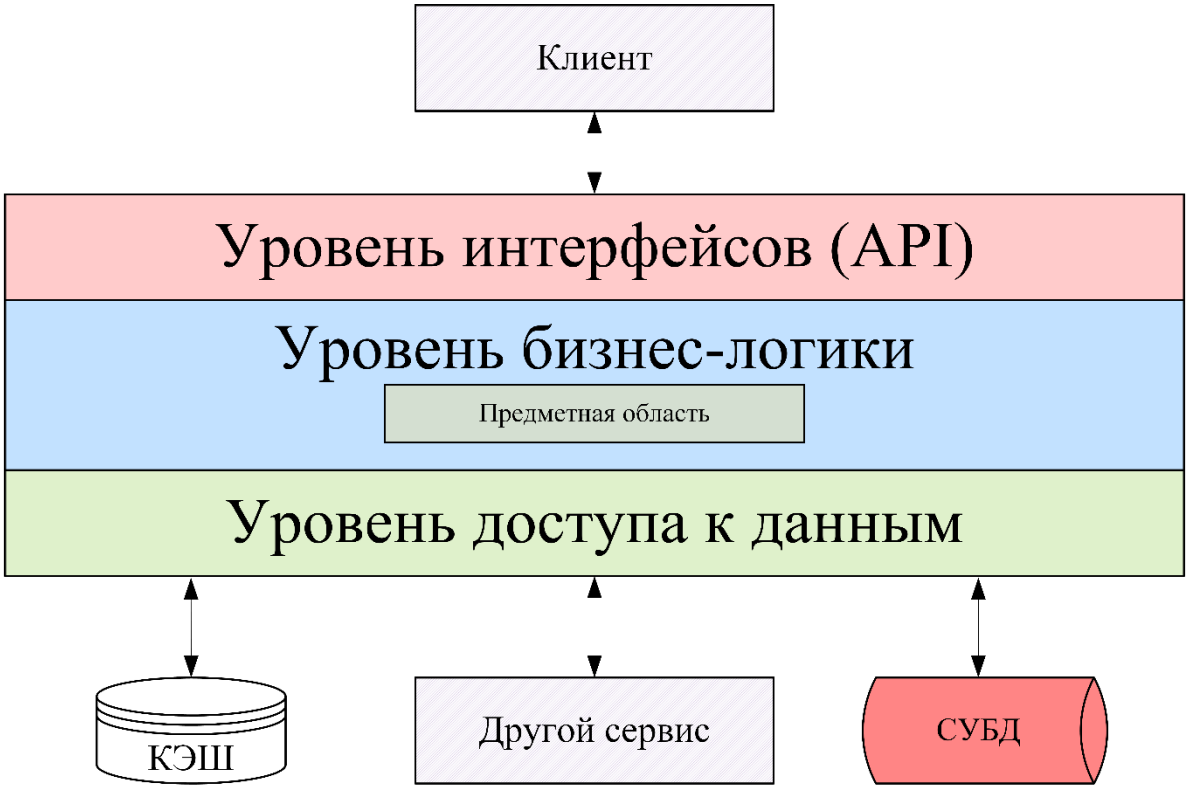

Рисунок 11 – Многоуровневая архитектура

В представленной архитектуре используется принцип вертикального управления. Каждый уровень использует нижестоящий уровень для выполнения своих функций, при этом нижестоящие уровни не должны ничего знать про вышестоящие.

В качестве клиентов системы могут выступать как живые люди, так и другие программы. Если в случае других систем для взаимодействия достаточно воспользоваться API (application programming interface) проектируемой в рамках

данной курсовой работы системы, то для обеспечения возможно взаимодействия пользователя-человека с системой необходима разработка отдельных графических интерфейсов.

Примеры сквозного взаимодействия пользователя-человека и пользователя-программы с распределенной системой показаны на рисунке .

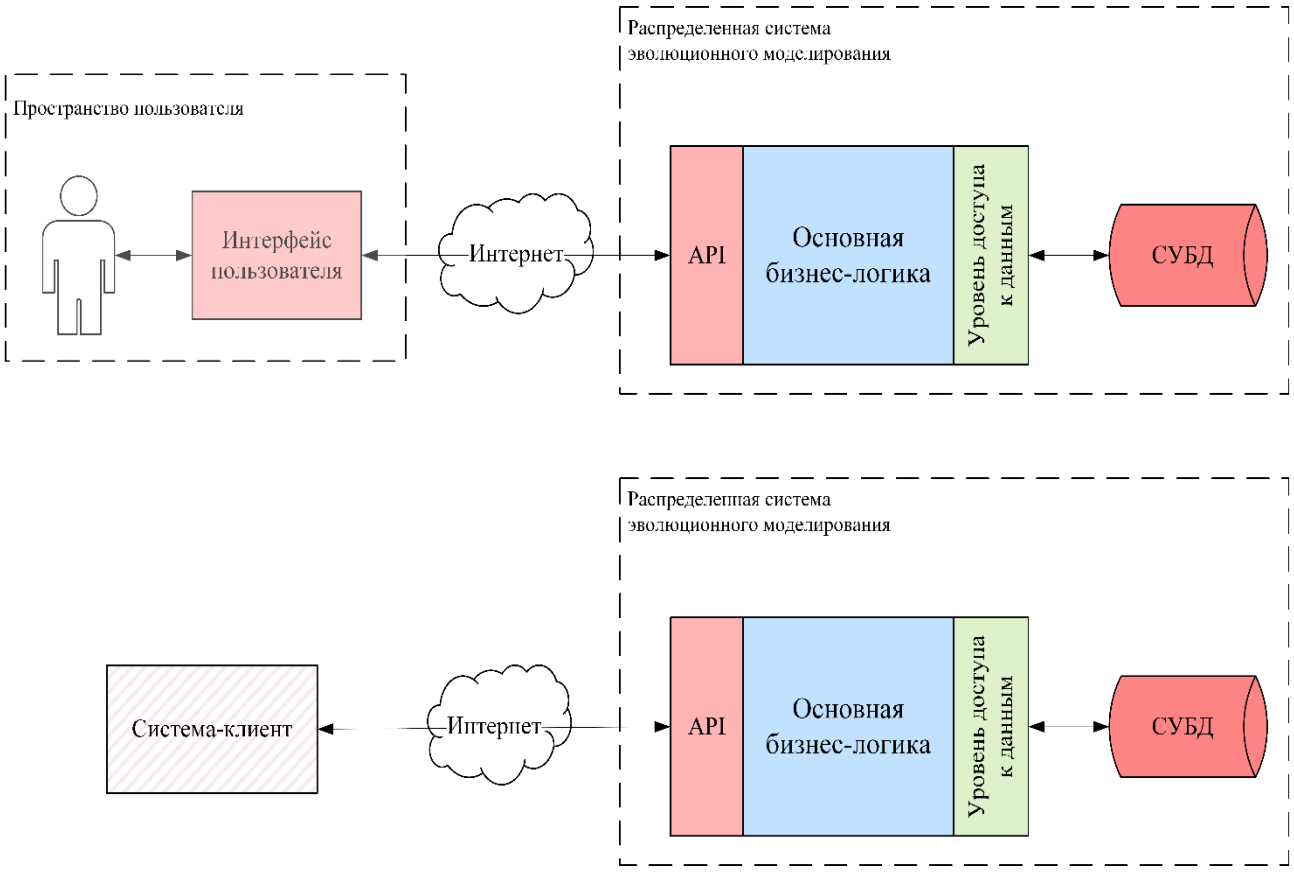

Рисунок 12 – Сквозное взаимодействие

Как видно из рисунка выше, для взаимодействия с системой пользователю необходимо получить соответствующее ПО с пользовательских интерфейсов. В случае веб-интерфейса пользователю достаточно перейти на соответствующий адрес.

Для мобильного или desktop интерфейса всё немного сложнее, однако соответствующее ПО может быть распространено с помощью сервисов цифровой дистрибьюции. В настоящей курсовой работе для демонстрации работоспособности системы предлагается использовать desktop интерфейс, о котором подробнее будет говориться в следующем разделе РПЗ.

2.3 Проектирования алгоритмов функционирования

При решении задачи оптимизации средствами проектируемой распределенной системы эволюционного моделирования чётко прослеживается иерархичность во взаимодействии имеющих место алгоритмов. Можно сказать, что имеет место наличие мета-алгоритмов.

Таким образом, по аналогии с многоуровневой архитектурой, вышестоящие алгоритмы используют нижестоящие для решения задачи, при этом нижестоящие алгоритмы ничего не знают вышестоящих.

Подобный процесс гибридизации теоретически может продолжаться бесконечно, однако в рамках данной курсовой работы было принято решение, что целесообразно остановиться на 2-компонентной гибридизации (использовать генетический и муравьиные алгоритмы) (рис.).

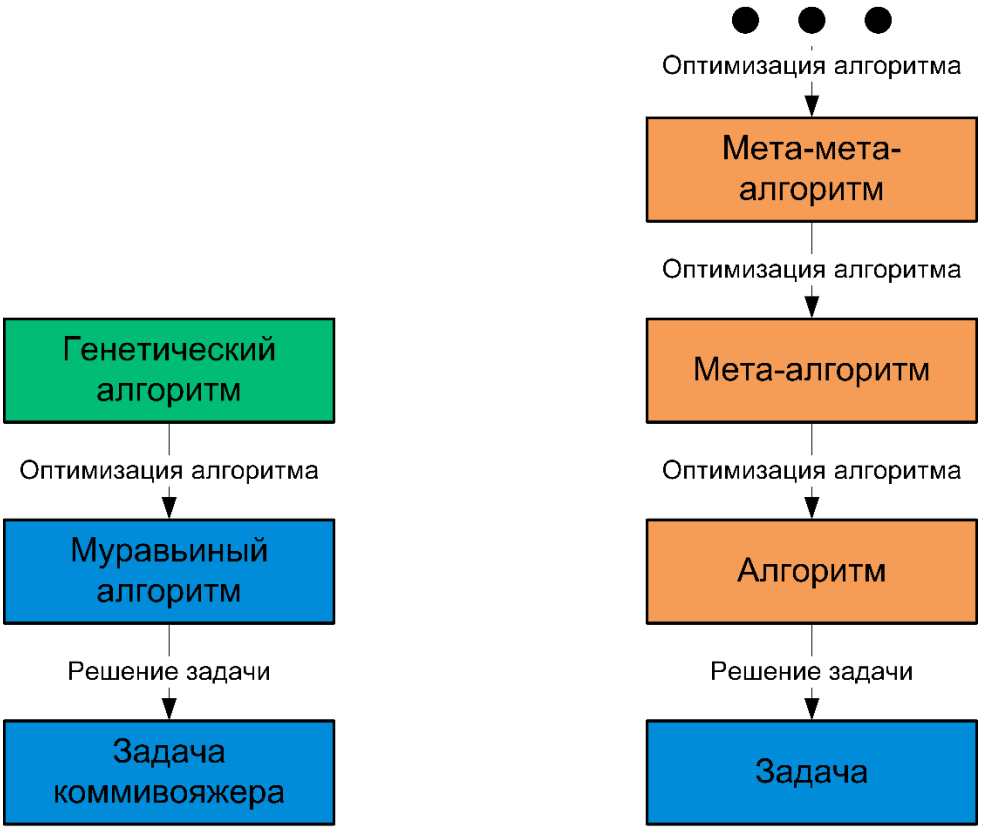

Рисунок 13 – Суть гибридизации

Таким, образом, если ГА является мата-алгоритмом оптимизации алгоритма решения задачи оптимизации, то для настройки самого ГА должен использоваться какой-то мета-мета-алгоритм, однако в рамках курсового проекта в роли настройщика ГА выступает пользователь системы.

Подбор конфигурации ГА и параметров для него – это нетривиальный процесс. Безусловно, конфигурация и параметры для ГА могут задаваться конечным пользователем, однако, если пользователь не является экспертом в предметной области задачи и предметной области эволюционного моделирования, это будет малоэффективно.

С точки зрения феномена интеллектуальных систем было бы желательно, если бы параметры и варианты конфигурации подбирались системой автоматически. Для этого могут применяться уже накопленные результаты предыдущих решений задач. В таком случае, следуя принципу правдоподобия, конфигурация для какой-либо задачи по умолчанию подбиралась бы на основе конфигурации для уже решенной максимально похожей задачи. Однако в случае отсутствия достаточного количества данных это может также не сработать.

Поскольку временные ресурсы ограничены, а ТЗ не подразумевает создание интеллектуальной системы, то вполне допустимым будет использование некоторых базовых зашитых в систему параметров ГА в комбинации с возможностью их варьирования со стороны пользователя. В качестве таких параметров могут выступать:

– размер популяции (количество агентов в поколении);

– процент отсеиваемых агентов;

– вероятность мутации агентов во время репликации;

– максимальное количество поколений (количество итераций ГА);

– критерий основа;

– формула расчёта функции приспособленности;

– интервалы возможных значений для генов;

– и т.д. и т.п.

Для муравьиного алгоритма также существует множество гиперпараметров для варьирования, однако подбор этих параметров входит в задачи генетического алгоритма, а не конечного пользователя системы. Одна из целей разрабатываемой системы – избежать полного перебора параметров, возложив эту функцию на ГА.

Разрабатываемая система проектируется таким образом, чтобы её потенциально возможно было переиспользовать и для других алгоритмов оптимизации и решаемых ими задач. В конце концов, если отбросить подробности внутренней работы, ГА занимается перебором параметров настраиваемого алгоритма.

ГА нет необходимо знать какую задачу решает оптимизируемый прикладной алгоритм. Это потенциально позволяет использовать один и тот же ГА для оптимизации (настройки) разных прикладных алгоритмов (рис.).

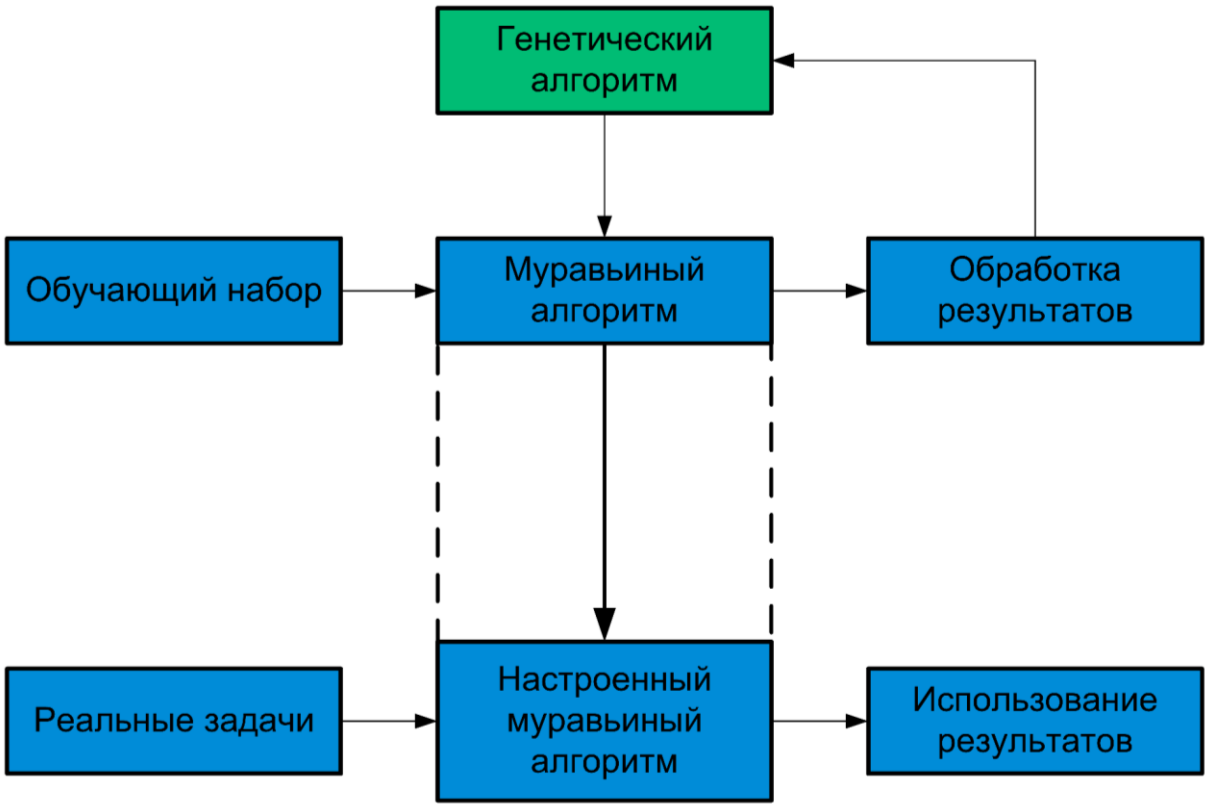

Рисунок 14 – Иллюстрация практической пользы системы

2.4 Использование микросервисного подхода

Наличие правил вертикального управления как с точки зрения многоуровневой архитектуры, так и с точки зрения алгоритмов функционирования позволяют легко разбить программные модули на микросервисы (рис.).

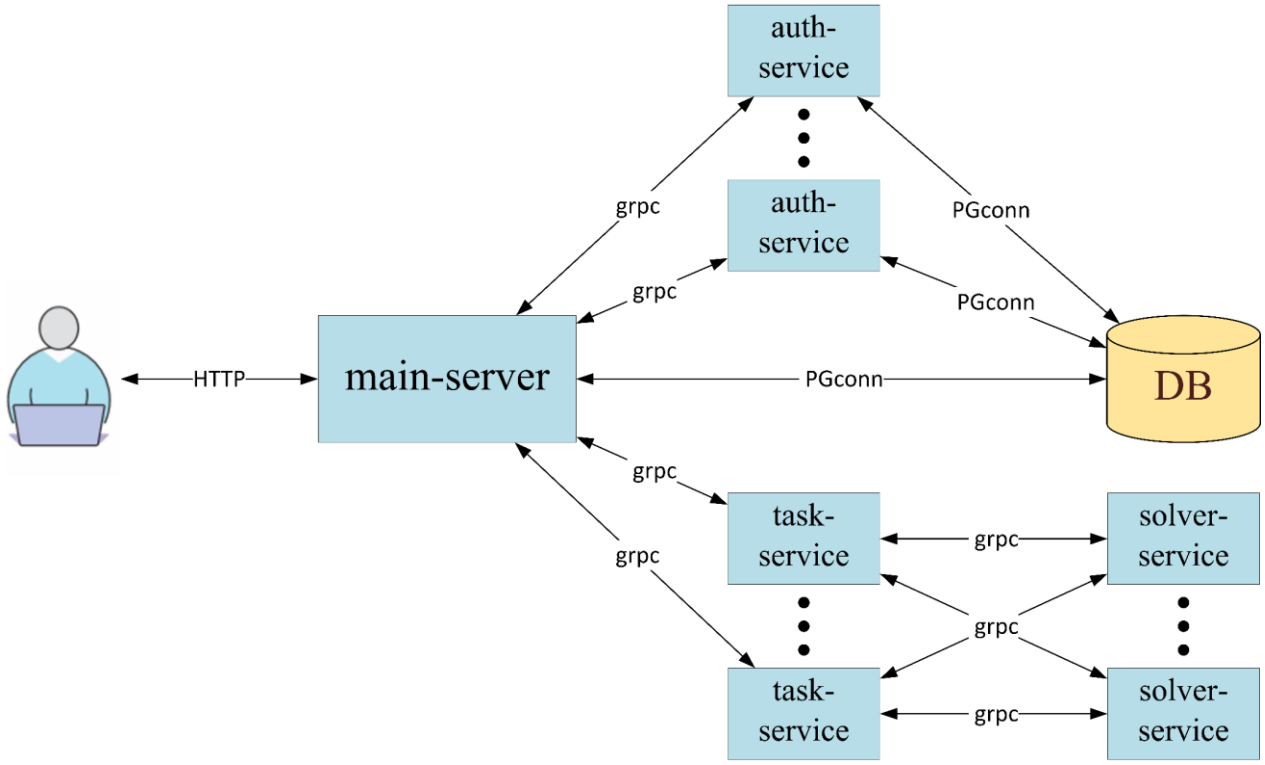

Рисунок 15 – Микросервисный подход

Исходя из рисунка выше может показаться, что main-server является узким местом, однако это не так. Ожидается, что большая часть вычислений (95%+) будет приходиться на task и solver сервисы. Main-server играет роль точки входа, т.е. всё, чем он должен заниматься – это трансфер данных между пользователями, сервисами и базами данных, который происходит не столь часто, т.к. прежде, чем сохранить данные какого-либо сеанса моделирования, этот сеанс должен полностью отработать, что (выяснено экспериментально) занимает от пары до несколько десятом минут.

В случае необходимости в дальнейшем main-server может быть также масштабирован, а между пользователем и main-server поставлено специализированное программное решение в виде nginx.

2.5 Проектирование базы данных

БД описывается совокупностью сервисных, пользовательских и сущностей предметной области и их связей, представленную в виде таблиц 2 и 3.

Таблица 2 – Сущности в БД без ссылочных атрибутов

Сущность	Атрибуты	Описание
Пользователь		Конечный пользователь системы, который использует её в целях решения интересующих его задач
Сессия пользователя		
Агент	ID, значение ЦФ	Вся работа ЭМ сводится к генерации агентов с как можно большим значением ЦФ
Ген	ID, позиция, значение	Параметр агента, исп. для вычисления значения ЦФ
Поколение	ID, порядковый номер	Набор агентов на одной итерации работы процесса ЭМ
Сеанс	ID, название, дата создания, описание	Связующая и разграничивающая структурная единица
Генетический алгоритм	ID, название, описание, параметры	Мета-алгоритм оптимизации
Прикладной алгоритм		Алгоритм оптимизации
Задача оптимизации		

Таблица 3 – Связи между сущностями БД

Связь	Тип	Описание
Поколение – Агент	Один-ко-многим	В рамках одного поколения существует множество агентов
Агент – Ген	Один-ко-многим	Агенты кодируются произвольным кол-вом генов
Агент - Агент	Один-ко-многим	У одного агента может быть множество предков и потомков
Сеанс – Поколение	Один-ко-многим	За один сеанс генерируется произвольное кол-во поколений
ГА – Сеанс	Один-ко-многим	Один и тот же ГА может использоваться в разных сеансах

Во второй таблице можно заметить одну связь вида «Многие-ко-многим». В рамках схемы БД эта связь реализуется через промежуточную таблицу.

Некоторые атрибуты сущностей, которые носят такие имена, как «свойства, параметры, показатели, настройки, схема и т.д.» не являются строго типизированными. Это означает, что в зависимости от используемых методов и алгоритмов, реализация программных модулей которых находятся за пределами разрабатываемой подсистемы, данные атрибуты могут иметь разное представление.

Использование нестрого типизированных атрибутов позволяет повысить гибкость и универсальность схемы БД. В противном случае приходилось бы использовать множество дополнительных сущностей и полей.

Совокупность связей и сущностей, описанных в таблицах выше, может быть представлено инфологической модель БД на рисунке 16.

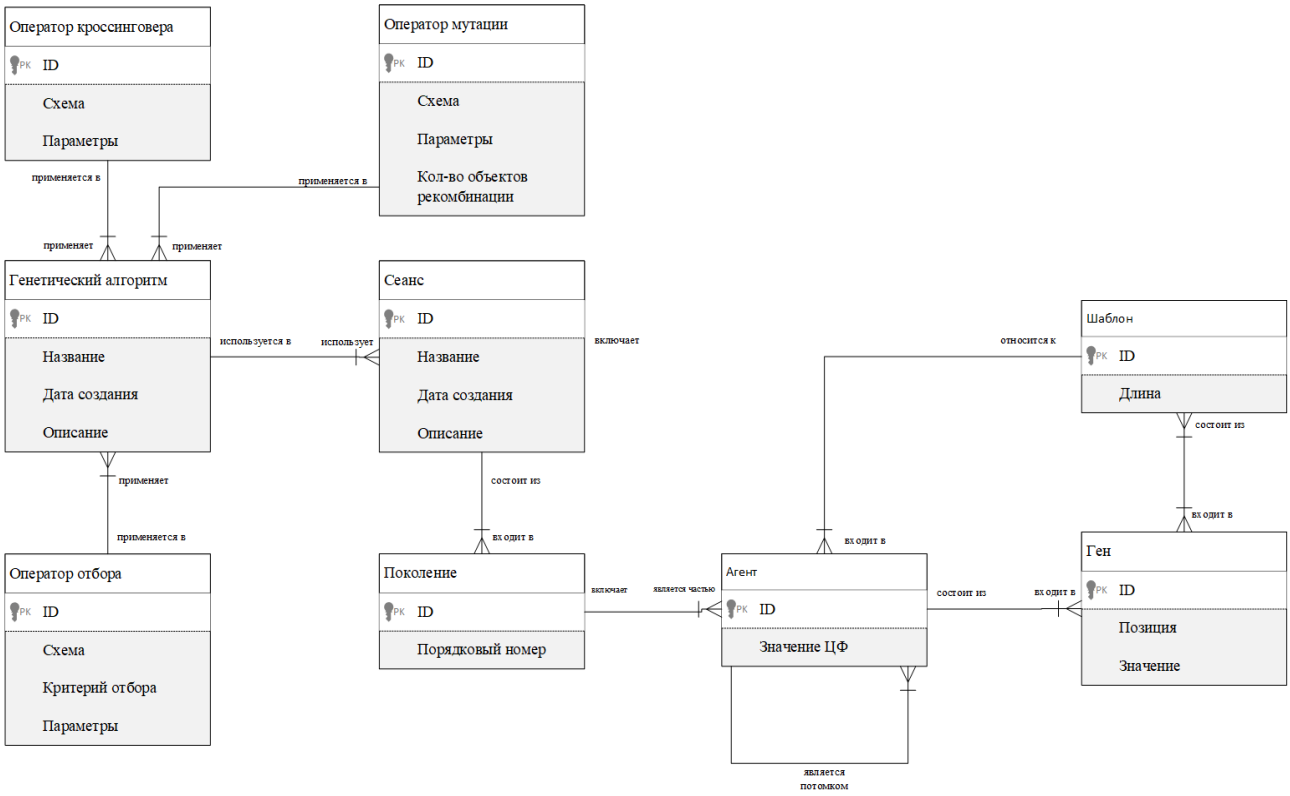

Рисунок 16 – Инфологическая модель БД

Инфологическая модель позволяет представить совокупность данных и связей между ними до того, как мы выберем конкретную модель хранения данных (реляционную, документоориентированную и т.д).

Из схемы выше видно, что сущности имеют между собой сильную связанность. Не наблюдает явной иерархичности (схема не вырождается в

дерево). Полагаясь на эти наблюдения, будет вполне оправдано принять решение о применении реляционной модели хранения данных.

3 Реализация программных компонентов системы. Отладка и тестирование

3.1 Выбор средств программной реализации

Для реализации распределенной системы эволюционного моделирования было принято решение использовать следующие технические компоненты:

– микросервис бизнес-логики (обработка данных);

– реляционная СУБД (чтение и запись данных);

– сервис оркестровки (шардирование, репликация, координация);

– сервис мониторинга (сбор метрик и их аналитики).

Выбранные программные реализации перечисленных технических компонентов представлены в таблице 4.

Таблица 4 – Выбранные программные решения

Компонент	**Программная реализация**	**Описание**
Миксросервис	Golang	Компилируемый многопоточный язык программирования, который широко используется при написании сервером с параллельной обработкой запросов.
СУБД	PostgreSQL	Одна из наиболее популярных свободно распространяемых реляционных СУБД
Оркестровка	Consul	Предоставляет ряд функций, обеспечивающих доступность информации об инфраструктуре
Мониторинг	Prometheus	Система мониторинга с богатым графическим интерфейсом и полностью открытым исходным кодом.

Consul и Prometheus являются готовыми решениями, которые не требуют написания программного кода, а нуждаются лишь в настройке в соответствии с решаемыми задачами. Микросервис Golang же является рукописным.

Таблицы, связи, ограничения, процедуры, триггеры и другое в PostgreSQL реализуются написанием SQL кода.

3.2 Реализация БД. Разработка скриптов создания объектов

Схема БД является реализацией инфологической модели с добавлением атрибутов, реализующих реляционную модель (внешние ключи и таблица для связей M-M). Итоговая схема представлена на рисунке 17.

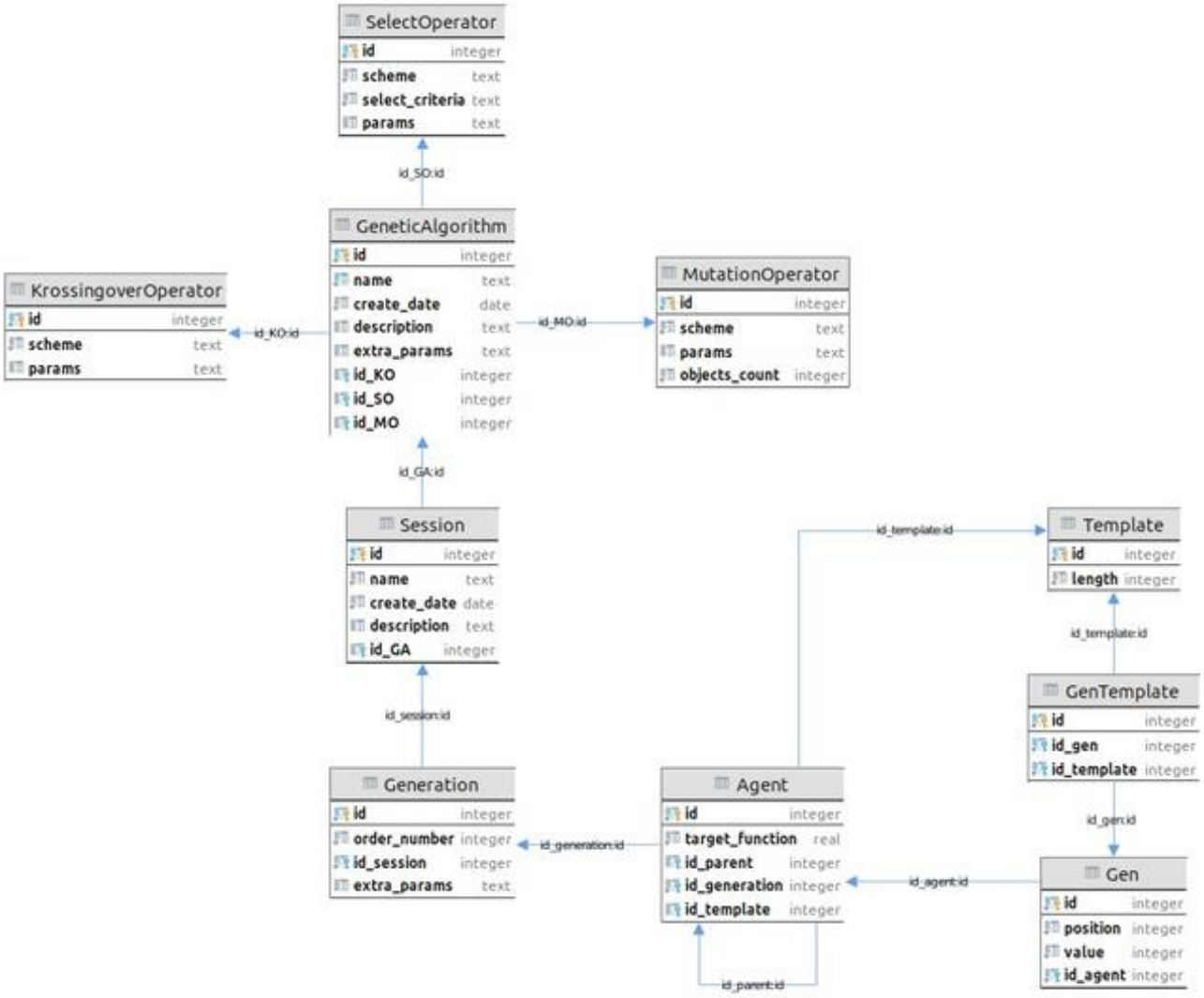

Рисунок 17 – Схема БД в PostgreSQL

Общее количество таблиц равно 10. Из них одна таблица не является отдельной сущность с позиции инфологической модели, а представляет из себя связующее звено, реализующее связь «многие-ко-многим».

Схема представленной БД можно условно разделить на 2 части – «конфигурационную» и «сессионную».

Конфигурационная часть часть состоит из таблиц: GeneticAlgorithm, KrossingoverOperator, MutationOperator, SelectOperator. Эти таблицы несут в себе

информацию о конфигурации модели генетического алгоритма, используемого в эволюционном методе.

Остальные таблицы несут в себе информацию касаемо результатов работы ГА при конкретных настройках модели.

Предполагается, что большая часть объемов данных будет приходиться на таблицы:

1) Session;

2) Generation;

3) Agent;

4) Template;

5) Gen;

6) GenTemplate.

Эмпирический опыт подсказывает, что их доля будет превышать 90% и со временем работы ИС будет только увеличиваться.

Поскольку количество записей в различных таблицах БД будет расти неравномерно, то имеет смысл разнести эти данные по разным узлам СУБД.

Предлагается использовать major узел с таблицами, содержащими конфигурационные данные, и minor узлы, которые в случае необходимости можно будет шардировать (рис. 18).

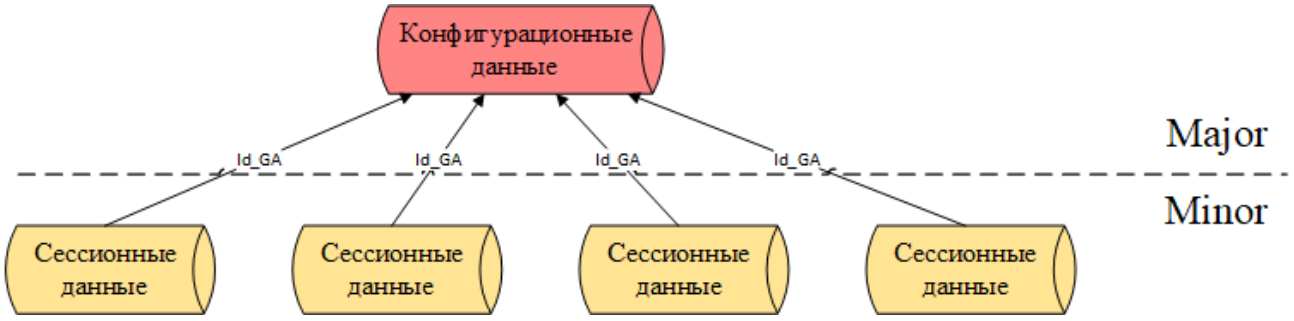

Рисунок 18 – Major и minor базы данных

Скрипты создания всех объектов (таблицы, первичные и внешние ключи, ограничения на поля, индексы и т.д.) БД представлены в приложении Б.

3.3 Реализация интерфейсов. Разработка API

Интерфейс подсистемы реализуется в виде набора протоколов, допустимых форматов передачи данных, процедур и функций, образующих API приложения.

При разработке микросервисов соблюдались соглашения об именовании функций и процедур API для каждого уровня в соответствии с таблицей 5.

Таблица 5 – Соответствие между видами операций и их именованием на каждом из уровней

Действие	**Уровень представления**	**Уровень бизнес-логики**	**Уровень доступа к данным**
Добавление	POST	Add	Insert
Выборка	GET	Read	Select
Модификация	PUT	Change	Update
Удаление	DELETE	Remove	Delete

Реализация API микросервисов Golang осуществляется в соответствии с набором интерфейсных ограничений, вытекающих из набора принципов REST:

– каждый ресурс обозначен уникальным идентификатором;

– все операции с ресурсами осуществляются через представление;

– самодостаточные сообщения в рамках одного запроса;

– использования гипермедиа для определения состояния клиента.

Основным форматом данных в реализуемой подсистеме является JSON. Передача структуры JSON осуществляется средствами протокола HTTP в составе тела запроса или ответа.

Ресурс, path и query параметры задаются с помощью URL (листинг 1)

Листинг 1 – Код модуля «Dialog»

```
http://localhost:1234/session/:id/generations?sort=ordNum
```

где localhost – хост, 1234 – порт, session – ресурс, id – path параметр, sort=ordNum – get параметр

3.4 Проработка репликации и шардирования

Принимая во внимание особенности предметной области, можно сделать вывод, что операции записи будут превалировать над операциями чтения. Исходя из этого факта, было принято решение предусмотреть возможность масштабирования на чтение за счёт механизма шардирования.

Разрабатываемая подсистема должна быть масштабируема как по каскаду микросервисов, так и по узлам СУБД.

Шардирование микросервисов предлагается осуществлять через регистрации микросервисов в сервисе Consul.

Шардирование СУБД предлагается осуществлять за счёт разделения пространства первичных ключей таблицы Session на фиксированные диапазоны, соответствующие определенным шардам.

В случае необходимости репликация данных может быть осуществлена в рамках каждого шарда встроенными средствами СУБД PostgreSQL.

Вышеописанное может быть проиллюстрировано рисунком 19.

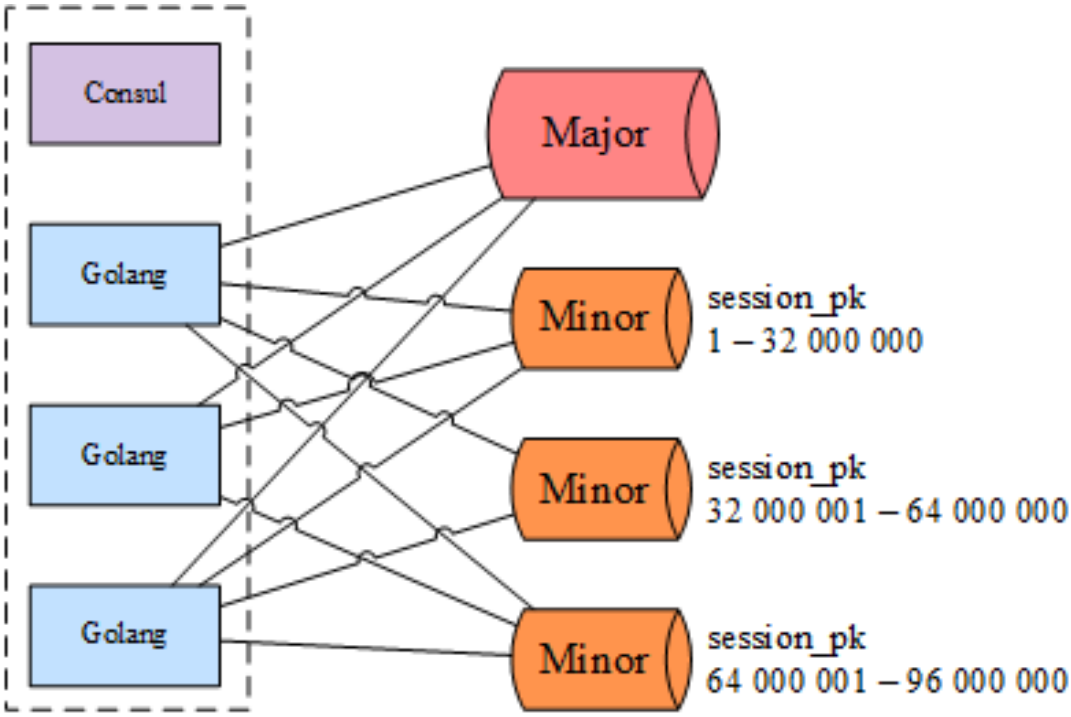

Рисунок 19 – Распределение диапазонов первичных ключей между шардами

4 Инструкция системного программиста

Инструкция системного запуска включает в себя последовательность действий для подготовки программного окружения, первичной конфигурации подсистемы, запуска и администрирования.

4.1 Подготовка программного окружения

Для корректного запуска и конфигурации подсистемы на локальной машине (рекомендуется использовать Ubuntu 18.04 LTS или docker контейнеры) необходимо установить следующие программные компоненты:

1) Система контроля версий git;

2) Docker (Ubuntu) версии 20.x и выше;

3) компилятор языка программирования Golang версии 1.15 или выше;

4) сервер СУБД PostgreSQL версии 10.X;

5) систему оркестровки Consul;

6) систему мониторинга Prometheus.

Все перечисленные компоненты распространяются в виде пакетов из официального репозитория Ubuntu.

4.2 Первичная конфигурация

Для начала работы необходимо скачать проект из репозитория, расположенного на сервисе github по ссылке https://github.com/ValeryBMSTU/rbd (доступ осуществляется с помощью публичных ключей ssh).

Воспользовавшись файлом с sql кодом, расположенном в папке db, необходимо с помощью утилиты pg_dump или консоли psql создать все необходимые объекты БД.

В конфигурационных файлах микросервиса Golang необходимо прописать значения с учётом особенностей программной среды (в случае тестового запуска можно оставить значения по умолчанию). Необходимо задать значения для следующих параметров:

– ip адрес микросервиса;

– прослушиваемый порт;

– ip адрес и порт службы Consul

– ip адрес и порт службы Prometeus;

– ip адрес и порт СУБД PostgreSQL;

– логин, пароль, БД и схема для запросов СУБД.

4.3 Запуск

В случае использования docker необходимо собрать docker-образ с помощью соответствующего файла и запустить.

В случае запуска подсистемы непосредственно на локальной машине необходимо запустить каждый из компонентов отдельно. Для корректности работы рекомендуется запускать компоненты в следующей последовательности:

1) сервер PostgreSQL;

2) служба Consul;

3) служба Prometheus;

4) сервер Golang.

Более подробную инструкцию по конфигурации и запуску можно найти в readme проекта.

4.4 Администрирование

Метрики и логи собираются централизованно службой Prometheus. Служба Prometheus может быть гибко настроена как с точки зрения собираемых данных, так и с точки зрения их анализа и представления (диаграммы, гистограммы, графики, speedometers и т.д.).

В случае необходимости запуска дополнительных микросервисов Golang они автоматически регистрируются в службе Consul. Дополнительных манипуляций не требуется.

В случае добавления шардов СУБД необходимо добавить ip-адрес нового шарда в конфигурационный файл микросервиса Golang и последовательно перезапустить все его экземпляры.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была создана ИС «Подсистема хранения данных процессов эволюционного моделирования».

В процессе выполнения курсовой работы был проведен анализ предметной области, выделены основные сущности предметной области и бизнес-процессы. На основе выделенных сущностей и инфологической модели была спроектирована и реализована схема базы данных. На основе выделенных бизнес-процессов были обозначены решаемые кейсы и выбраны проектно-технические решения.

Для спроектированной системы были выбраны инструменты программной реализации. Все использованные инструменты являются свободно распространяемым программным обеспечением.

Итоговая ИС выполняет поставленные перед ней задачи в достаточно мере. Так как разработанная ИС является подсистемой, то проверить правильность и быстродействие работы невозможно из-за отсутствия для того нужных программных компонентов остальной системы. Несмотря на это, в подсистеме продуманы механизмы горизонтального масштабирования, в частности, механизмы шардирования и репликации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аверченков В. И., Казаков П. В. Эволюционное моделирование и его применение. – 2012.

2. Виноградов Д. В. Области применения эволюционных алгоритмов //Вестник МГУП имени Ивана Федорова. – 2016. – №. 2. – С. 18-19.

3. Zadeh L. A. Soft Computing and Fuzzy Logic // IEEE Software. 1994. № 6 (11). С. 48–56.

4. Методология IDEF0. Стандарт. Русская версия.

5. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка //СПб.: Питер. – 2018.

6. The Golang 1.15 Manual [Электронный ресурс] // The Go Programming Language. URL: https://golang.org/doc // (дата обращения 30.04.2021).

7. The Echo. Go web framework manual [Электронный ресурс] // Labstack. URL: https://echo.labstack.com/guide (дата обращения 04.05.2021).

ПРИЛОЖЕНИЕ А

Техническое задание

Листов 5

ПРИЛОЖЕНИЕ Б

Исходный текст программных модулей

Листов 18