СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ 5

ВВЕДЕНИЕ 6

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ РАЗРАБОТКИ, СРАВНЕНИЕ С АНАЛОГАМИ 9

1.1 Описание предметной области с анализом существующих решений и их недостатков исходя из целей работы и критериев сравнения, которые должны соответствовать искомым решениям поставленных задач. 9

1.1.1 MAC 9

1.1.2 Исследование операций 11

1.2 Разработка возможных направлений проведения исследований и методов решений отдельных задач, обоснование выбора оптимального варианта 12

1.3 Обоснование применяемых технологий и инструментов. . . . 12

1.4 Уточненная постановка задачи и требования к прототипу решения. 15

1.5 Выводы. 17

2 ПРОЕКТИРОВАНИЕ 18

2.1 Проектирование архитектуры прототипа системы решения, включая разработку структуры (компоненты и связи между ними), внутреннего и внешнего функционирования. 18

2.1.1 Анализ доступных средств платформы .Net 18

2.1.2 Определение алгоритмов, по которым будет работать программный модуль 19

2.1.3 Выбор типа базы данных и её провайдера. 20

2.1.4 Проектирование архитектуры 21

2.2 Разработка прототипов алгоритмов (как отдельных компонентов и их взаимосвязей, так и в целом системных), структур данных. 23
2.2.1 Типы данных Google OR-Tools 23
2.2.2 Определение требований к модели 25

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

ПО – программное обеспечение

EAM – Enterprise Asset Management - система управления активами

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Makespan – период времени, который проходит от начала работы до ее завершения.

Job-shop – оптимизационная проблема в информатике и исследовании операций. Это вариант оптимального планирования заданий. В общей проблеме планирования заданий нам дается n заданий $J_1, J_2, ..., J_n$ с различным временем обработки, которые должны быть запланированы на m машинах с различной вычислительной мощностью, при этом мы пытаемся минимизировать время выполнения (makespan) - общую длину расписания.

ВВЕДЕНИЕ

Актуальность.

EAM-системы - чрезвычайно важная часть производственного процесса предприятий. Это инструмент управления производительностью, рисками и затратами, связанными с физическими (производственными) активами организаций.

Для современных производственных процессов актуальными являются задачи оптимизации: оптимизация маршрутов логистики, рабочего времени, производственных затрат и т. д. Масштаб таких оптимизационных задач может быть непосилен для решения сугубо человеческими ресурсами, даже при использовании EAM-систем, поэтому необходимо включать в область решения данных задач программно-аппаратные комплексы, решающие подобные задачи.

С 2011 года мировыми конгломератами ставится вопрос о четвертой промышленной революции - индустрии 4.0. Отходя от её утопических идей, индустрия 4.0 это гигантский шаг в сторону модернизации всех аспектов производства.

Большие данные, интернет вещей, блокчейн, искусственный интеллект, цифровые двойники - все эти технологии, которые уже нашли те или иные способы применения в различных областях науки и техники, только и ждут своего часа применительно к производству и производственным процессам. Уже сейчас самые передовые производства невероятно трудно представить без всех этих технологий.

Так новый передовой завод Intel (ссылка на видос Линуса) - полностью автоматизированное цифровое производство, где все самые важные и ответственные роли исполняют именно роботизированные механизмы или автономные дроны с интеллектом роя.

Умная фраза в заключение актуальности

Степень разработанности темы, научно-техническая проблема.

На текущий момент на рынке представлено достаточно много решений, связанных с EAM и ERP системами, как отечественные:

– ”TRIM”
– 1C ERP
– ”Галактика ERP”

так и зарубежные:

– SAP ERP
– IBM Maximo

Все представленные EAM-системы (как отдельные, так и в составе ERP) позволяют в полуавтоматическом режиме планировать работы, то есть указать какое количество ресурсов и когда понадобится для обслуживания тех или иных активов.

Во всех приведённых выше системах данное планирование происходит по потребности, то есть в долгосрочной перспективе не учитывает внезапные изменения, которые могут произойти с ресурсами, и их всевозможные коллизии - ситуации, когда в один момент времени один и тот же ресурс может быть задействован в разных работах.

Исходя из этого можно сказать, что в таких системах стоит до сих пор нерешенная проблема актуализации ресурсного планирования. Именно решение этой проблемы является целью данной работы.

Объект работы

В качестве объекта работы выступают данные, полученные из системы класса EAM, структурированные определенным образом. В общем виде - база данных с набором сущностей вида ”работа”, ”тип работы, ”исполнитель” и так далее.

Цель работы и решаемые задачи исходя из темы.

Цель данной работы - создание открытого мультиплатформенного решения, предназначенного для ресурсного планирования в системах управления физическими активами.

В данной работе решается задача ресурсного планирования, то есть распределение заданий по исполнителям в заданном временном интервале с учетом ограниченного количества имеющихся ресурсов.

Планируемый н/т и практический уровень разработки, связь данной работы с другими.

В результате работы планируется получить готовое программно-техническое решение, которое может взаимодействовать с имеющимися EAM-системами и обеспечивать для них решение задачи ресурсного планирования с заданными граничными условиями.

Методы или методология проведения работы.

Убираем этот блок совсем.

Структура работы.

Работа состоит из Введения, N глав и Заключения, содержит список литературных источников из M наименований и Приложения/ий.

1 Анализ предметной области разработки, сравнение с аналогами

1.1 Описание предметной области с анализом существующих решений и их недостатков исходя из целей работы и критериев сравнения, которые должны соответствовать искомым решениям поставленных задач.

Одной из областей в операционных исследованиях является составление и оптимизация расписаний. Когда количество работ, человеческих ресурсов и различных механизмов невелико, то составление расписания вполне посильная для человека задача. Но когда количество объектов, учитываемых при планировании, возрастает, например, до уровня предприятия, то задача перестаёт быть тривиальной даже для целого отдела планирования. В таких случаях становится необходимым использование средств автоматизации планирования, примером которых являются мультиагентные системы или программные комплексы комбинаторной оптимизации - отдельные или в рамках EAM-систем.

В целом можно разделить существующие методы построения и оптимизации подобных моделей на два основных подтипа - Мультиагентные системы (MAC) и Исследование операций (Operations research - далее OR) в рамках целочисленного программирования (программирования в ограничениях).

1.1.1 MAC

Мультиагентная система – система, состоящая из множества взаимодействующих друг с другом агентов. Каждый агент в такой системе независим и не имеет представления о системе в целом, но знает свои задачи и потребности, а также задачи и потребности агентов на одном с ним уровне. Для простоты понимания можно условно сказать, что агенты представляют собой

нечто схожее с NPC (Неигровой персонаж (от англ. Non-Player Character)) в компьютерных играх.

Мультиагентные системы начали широко развиваться в девяностых годах и достигли своего пика к середине двухтысячных годов, после чего парадигма создания агентных сетей как чего-то, требующего отдельных подходов умерла в связи развитием и упрощением программирования в целом.

Согласно статьи Virginia Dignum и Frank Dignum, [1] несмотря на существование целых организаций по изучению мультиагентных систем, ни за время существования этих организаций, ни за время их работы, МАС не смогли проявить себя ни в одной из заявленных областей, кроме как моделирования поведения персонажей в компьютерных играх [2]. Исследователи связывают неудачу МАС со слабой способностью конкурировать с другими компьютерными технологиями того времени, в первую очередь вызванной тем, что из-под пера ассоциации или заинтересованных исследователей так и не вышло платформы МАС, которую могли бы использовать все желающие за пределами «закрытого» сообщества. Второй проблемой при попытке применить исследования агентов на практике, по их мнению, стала неспособность охватить или учесть существенные характеристики проблемы, для решения которой используются агенты. Аналогичные выводы были представлены Городецким и Скобелевым в статье 2017 года «МНОГОАГЕНТНЫЕ ТЕХНОЛОГИИ ДЛЯ ИНДУСТРИАЛЬНЫХ ПРИЛОЖЕНИЙ: РЕАЛЬНОСТЬ И ПЕРСПЕКТИВА» [3].

В целом всё-таки можно сказать, что МАС – достаточно перспективное направление, второе дыхание которому могут дать современные разработки в области искусственного интеллекта, нейросетей и программирования.

Главным плюсом (из которого в общих случаях следует и главный минус) мультиагентных систем является скорость реакции на изменения в системе и внешние факторы. Предполагается, что такая система фактически моментально должна реагировать на «происшествия» и сразу же выдавать удовлетворяющее решение с приемлемым качеством. Данное свойство мультиагентных систем делает их максимально пригодными для использования в

качестве систем решения задач реального времени или в качестве модели-двойника.

Под моделью-двойником представляется виртуальный клон реальной системы, где все объекты взаимодействия являются агентами. Предполагается, что такие модели будут использоваться, например, для симулирования внештатных ситуаций в системе и оценки реагирования системы и агентов на эти ситуации.

Так как МАС предлагают в целом «приемлемые решения», то в областях, где важна каждая мелочь и требуется наиболее полный точный результат, такой подход не подойдет. МАС вполне можно использовать на этапах конкретной эксплуатации, где как раз необходимы оперативные принятия решений, а вот в задаче установки оптимальных начальных условий такой подход скорее всего будет неприменим.

1.1.2 Исследование операций

Для получения наиболее точных и оптимальных решений применяется дисциплина под названием «Исследование операций». Согласно Википедии [4], «ИО - дисциплина, занимающаяся разработкой и применением методов нахождения оптимальных решений на основе математического моделирования, статистического моделирования и различных эвристических подходов в различных областях человеческой деятельности. Иногда используется название математические методы исследования операций.»

Методы исследования операций являются основными в решениях таких задач как планирование, транспортные потоки, маршрутизация, упаковка и задачи о назначениях. В варианте, представленном в статье, предлагается использовать методы комбинаторной оптимизации, а именно – целочисленное программирование в ограничениях.

Программирование в ограничениях — это теория решения комбинаторных задач, опирающаяся на большое количество знаний из областей искусственного интеллекта, информатики и исследования операций. В програм-

мировании в ограничениях декларативно задаются ограничения для набора переменных входных данных. Ограничения не определяют последовательность действий для получения искомого результата, а указывают свойства и особенности для нахождения конечного результата.

Оптимизация с ограничениями, или программирование с ограничениями (англ.- constraint programming, далее CP), — это название, данное идентификации выполнимых решений из очень большого набора возможных решений. CP основывается на выполнимости (нахождении выполнимого решения), а не на оптимизации (нахождении оптимального решения) и фокусируется на ограничениях и переменных, а не на целевой функции. Фактически, задача CP может даже не иметь целевой функции - цель может заключаться просто в том, чтобы сузить большое множество возможных решений до более управляемого подмножества путем добавления ограничений к задаче.

1.2 Разработка возможных направлений проведения исследований и методов решений отдельных задач, обоснование выбора оптимального варианта

Рассказать про другой опенсорс, про другие программные методы реализации, еще какой метод

1.3 Обоснование применяемых технологий и инструментов.

Реализовывать различные методы комбинаторной оптимизации представляется достаточно сложной задачей, поэтому предполагается использование готовых библиотек и фреймворков, реализующих методы исследования операций.

Одними из самых популярных фреймворков для решения задач исследования операций являются открытые GLPK [5], LP_Solve [6], MiniZinc [7], Google OR-Tools [8], CHOCO [9] и проприетарные IBM CPLEX [10] и Gurobi [11].

Приведем частичные результаты бенчмарков между Google OR-Tools и IBM CPLEX (ссылка на таблицу).

Таблица 1.1 – Сравнение Or-Tools и CPLEX в задаче Job-shop (меньше - лучше)

Номер теста	Одно ядро		Четыре ядра	
	CPLEX msp (sec)	OR-Tools msp (sec)	CPLEX msp (sec)	OR-Tools msp (sec)
1	1234 (1.9)	1234 (1.8)	1234 (3.3)	1234 (1.6)
2	943 (0.7)	943 (0.7)	943 (1.4)	943 (0.4)
3	656 (1169.3)	660	565 (525)	661 (1200)
4	682	679	680	679
5	685	695	694	689
6	55 (0)	55 (0)	55 (0)	55 (0)
7	930 (3.8)	930 (5)	930 (5.9)	930 (2.9)
8	1165 (1.4)	1165 (5)	1165 (0.5)	1165 (3.4)
9	666 (0)	666 (0.1)	666 (0)	666 (0.1)
10	655 (0.3)	655 (0.3)	655 (0.5)	655 (0.1)

В тестах производительности между Google OR-Tools и IBM CPLEX стоял вопрос решения проблемы составления расписания работы в цехе (англ. Job-shop scheduling problem), которая приобрела особую известность благодаря своей простой формулировке, приводящей к трудно решаемым задачам.

Наиболее типичным критерием оптимизации является минимизация промежутка времени (англ. makespan), т.е. промежутка времени между началом первой операции и окончанием последней. Проблема представляется в виде набора заданий, которые должны быть обработаны набором машин. Каждое задание - это последовательность операций, каждая операция должна быть обработана определенной машиной и занимает определенное время обработки. Каждое задание имеет определенный порядок операций, который должен соблюдаться. Допустимым решением этой задачи является такая последовательность операций на каждой машине, при которой нет перекрытия

времени между двумя операциями на одной машине и соблюдается порядок операций.

Как видим из статьи, по результатам тестов на момент 2017 года фреймворк IBM CPLEX в целом чаще выигрывал у OR-Tools в размере makespan и быстроте получения решения, при этом фреймворк OR-Tools показывал себя лучше в многоядерных тестах большого масштаба.

В статье https://journals.lib.unb.ca/index.php/AOR/article/view/18595/20992 представлены уже результаты сравнения коммерческих IBM CPLEX и Gurobi. По результатам тестов последние версии обоих решателей в целом идут нога в ногу и выдают результаты очень близкие как по качеству, так и по скорости нахождения решений.

По результатам бенчмарков [12] [13] (таблица) можно сказать, что между фреймворками Google OR-Tools, IBM CPLEX и Gurobi (особенно в многоядерной конфигурации) существует условный паритет по скорости и качеству решения оптимизационных задач, и в целом предугадать, кто окажется впереди в условиях реальной задачи невозможно.

Одним из наиболее доступных фреймворков является Google Or-Tools. OR-Tools - это программное обеспечение с открытым исходным кодом для комбинаторной оптимизации, которая направлена на поиск наилучшего решения проблемы из очень большого набора возможных решений. Последние несколько лет именно OR-Tools среди всех открытых библиотек занимает первые места в соревнованиях MiniZinc Challenge по программированию в ограничениях [14]. Также, в отличие от большинства остальных фреймворков имеет большее количество официально написанных интерфейсов под разные языки программирования – C++ (изначально написан именно на нём), C#, Java, Python.

Фреймворк Google OR-Tools поддерживает решатели под задачи линейной оптимизации (решения проблемы, смоделированной как набор линейных зависимостей.), целочисленной оптимизации (Смешанное целочисленное программирование - Mixed Integer Programming – MIP; некоторые переменные обязательно представляют собой целые числа) и программи-

рование в ограничениях. Все эти решатели могут быть использованы для решения таких задач, как: решение судоку, диета Стиглера[15], шахматные задачи, планирование, составление оптимальных маршрутов и т.д.

В результате обзора и анализа методов решения поставленной задачи, на данный момент наиболее подходящим методом можно признать метод комбинаторной оптимизации и программирования в ограничениях, так как явно предоставляет набор устоявшихся практик и решений, показавших себя во множестве проектов.

Основным средством предлагается использовать фреймворк Google OR-Tools, решающий задачи комбинаторной оптимизации, так как данный фреймворк является бесплатным, открытым, может быть использован в коммерческой разработке и покрывает большую часть областей и задач исследования операций.

Мультиагентные системы на данный момент, напротив, по результатам анализа показали себя непригодными к повсеместному использованию, так как не предоставляют доступных средств для реализации и не были широко протестированы в реальных условиях.

1.4 Уточненная постановка задачи и требования к прототипу решения.

Имеются следующие начальные условия:

– Есть множество запланированных работ (ЗР) на заданном временном интервале. Каждая работа имеет расчетные плановую дату/время начала выполнения и плановую дату/время завершения выполнения (разница между этими датами определяет длительность работы).
– Каждая работа имеет перечень исполнителей и механизмов, необходимых для ее выполнения (каждый из этих перечней может быть пустым). Для каждого исполнителя и механизма задана потребность, выраженная в чел.*час. (либо, аналогично, машино*час.). В качестве исполнителя (механизма) может быть указан либо конкретный испол-

нитель (штатная единица (ШЕ) – для человека или конкретный механизм), либо требуемая квалификация (профессия) исполнителя или необходимый тип механизма.

– Имеется подразделение (цех, участок и т.п.), располагающее определенными людскими ресурсами и определенным набором механизмов. Состав людских ресурсов определяется штатным расписанием (оно имеет структуру), перечень доступных механизмов также задан в этой структуре. Каждая ШЕ (людской ресурс) в составе штатного расписания имеет свою профессию, а механизм – соответствующий ему тип механизма.

Задача:

На основании имеющихся исходных данных постараться построить такой порядок выполнения работ, чтобы в каждый день суммарная потребность в людских ресурсах и механизмах на выполнение всех работ не превышала объем располагаемых ресурсов и механизмов, заданных в штатном расписании.

Для того, чтобы найти такое решение, система может перемещать исходные ЗР во времени в заданных пределах. При этом должны соблюдаться следующие условия:

– длительность работ и потребность в ресурсах, указанные для каждой ЗР, должны сохраняться;
– порядок выполнения работ одного типа и для одного и того же объекта должен сохраняться;
– могут быть заданы ограничения по изменению сроков выполнения работ: например, нельзя изменять сроки более, чем на 7 дней в любом направлении, либо можно изменять не более, чем на 10 дней в сторону уменьшения, но не более, чем на 2 дня в сторону увеличения.

Результат:

В результате система должна предложить новые плановые даты выполнения каждой ЗР при условии соблюдения заданных ограничений. Если решения нет, то система должна сообщить об этом, желательно при этом

указать, какого ресурса или механизма (по мнению системы) недостаточно для выполнения плана.

1.5 Выводы.

Умные слова

2 Проектирование

Проектирование планировщика ресурсов предполагает собой разработку, состоящую из следующих этапов:

– анализ доступных средств платформы .Net;
– выбор типа базы данных и её провайдера;
– определение алгоритмов, по которым будет работать программный модуль;
– написание и отладка программного модуля.

2.1 Проектирование архитектуры прототипа системы решения, включая разработку структуры (компоненты и связи между ними), внутреннего и внешнего функционирования.

2.1.1 Анализ доступных средств платформы .Net

На данный момент платформу .Net можно условно поделить на две части - .Net Framework и .Net Core.

На данный момент платформа .Net Framework является поддерживаемой, но устаревшей, и по рекомендациям Microsoft 111https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server111 не следует начинать новые проекты на этой платформе. Дополнительно необходимо указать, что платформа .Net Framework не является кроссплатформенной и работает только в операционных системах семейства Windows, чего недостаточно в мире современной разработки на данный момент.

Открытый кроссплатформенный фреймворк .Net Core является преемником .Net Framework. Именно Core версию Microsoft рекомендует для всех новых проектов. Главным отичием новой платформы в первую очередь стала поддержка работы в Unix-системах (Linux, MacOS), соответственно все сервисы, написанные на .Net Core, без каких-либо ограничений могут быть

использованы в современной серверной парадигме: Linux серверы, Docker, контейнеризация и виртуализация.

Для данного проекта была выбрана самая акутальная версия фреймворка .Net Core - .NET 6, так как она является LTS-версией, а так же соответствует всем требованиям проекта - открытости и кроссплатформенности.

В качестве платформы для взаимодействия с пользователем был выбран серверный вариант фреймворка Blazor. Blazor - это бесплатный и открытый фреймворк для написания веб приложений на языке C#, поддерживающий реактивность, то есть работающий напрямую с DOM-элементами, а не HTML страницы. Blazor делится на Blazor WebAssembly - полностью клиентское приложение и на Blazor Server - клиент-серверную реализацию, в которой пользовательская и серверная часть общаются между собой по протоколу SignalR. SignalR - это бесплатная библиотека с открытым исходным кодом для Microsoft ASP.NET, которая позволяет серверному коду отправлять асинхронные уведомления веб-приложениям на стороне клиента. Библиотека включает в себя серверные и клиентские компоненты JavaScript.

2.1.2 Определение алгоритмов, по которым будет работать программный модуль

За планирование и/или перепланирование в планировщике будет отвечать библиотека Google OR-Tools. Задача, изложенная выше в первой главе, однозначно подходит под определение задач целочисленного программирования в ограничениях, так как каждый объект виртуальной модели в данной задаче можно определить как целочисленную переменную или их множество, на которые в явном виде можно наложить ряд ограничений и целевых функций, соответствующих условиям задачи.

Для такого типа задач пакет Google OR-Tools предоставляет отдельный класс для решения задач целочисленного программирования в ограничениях - CP-SAT, где CP (от англ. Constraint Programming - Программирование в ограничениях) - непосредственно целочисленное программирование в ограниче-

ниях, а SAT (от англ. Satisfiability - Удовлетворимость) - указывает, что целью решения данного класса задач является не нахождение одного конкретного результата, а нахождение одного или нескольких решений, удовлетворяющих ограничениям, наложенным на входные данные.

2.1.3 Выбор типа базы данных и её провайдера.

В первую очередь при выборе типа баз данных встаёт вопрос об использовании реляционных или нереляционных баз данных. В целом в EAM-системах обычно используются именно реляционные базы данных, так как этому способствует необходимость в четкой иерархической структуре модели представления активов предприятия. Также в EAM-системах редко встречается необходимость в шардинге - то есть в горизонтальном расширении данных для обеспечения масштабируемости. Исходя из этого для данного проекта была выбрана именно реляционная база данных.

В качестве реляционной СУБД был выбран PostgreSQL. PostgreSQL - это бесплатная система управления реляционными базами данных с открытым исходным кодом, в которой особое внимание уделяется расширяемости и соответствию стандарту SQL.

В данный момент ввиду острой необходимости в импортозамещении и независимости от зарубежного ПО, сервисов и средств именно СУБД PostgreSQL видится лучшим вариантом. У российских разработчиков уже есть большой опыт в развитии своих доработок и форков PostgreSQL. Так, например, давно и успешно применяется форк Postgres Pro ССЫЛКА, СУБД ЛИНТЕР-ВС на базе PostgreSQL используется в ОС МСВС, а в дистрибутив операционной системы Astra Linux Special Edition (версия для автоматизированных систем в защищённом исполнении, обрабатывающих информацию со степенью секретности «совершенно секретно» включительно) включена СУБД PostgreSQL, доработанная по требованиям безопасности информации.

2.1.4 Проектирование архитектуры

Желаемая архитектура системы представлена на рисунке 1. Предполагается, что разрабатываемая система ресурсного планирования будет являться отдельным сервером планирования, работающим независимо от EAM-системы. Связь между EAM-системой и сервером планирования должна осуществляться посредством наличия общей базы данных и API для передачи команд и промежуточных объектов.

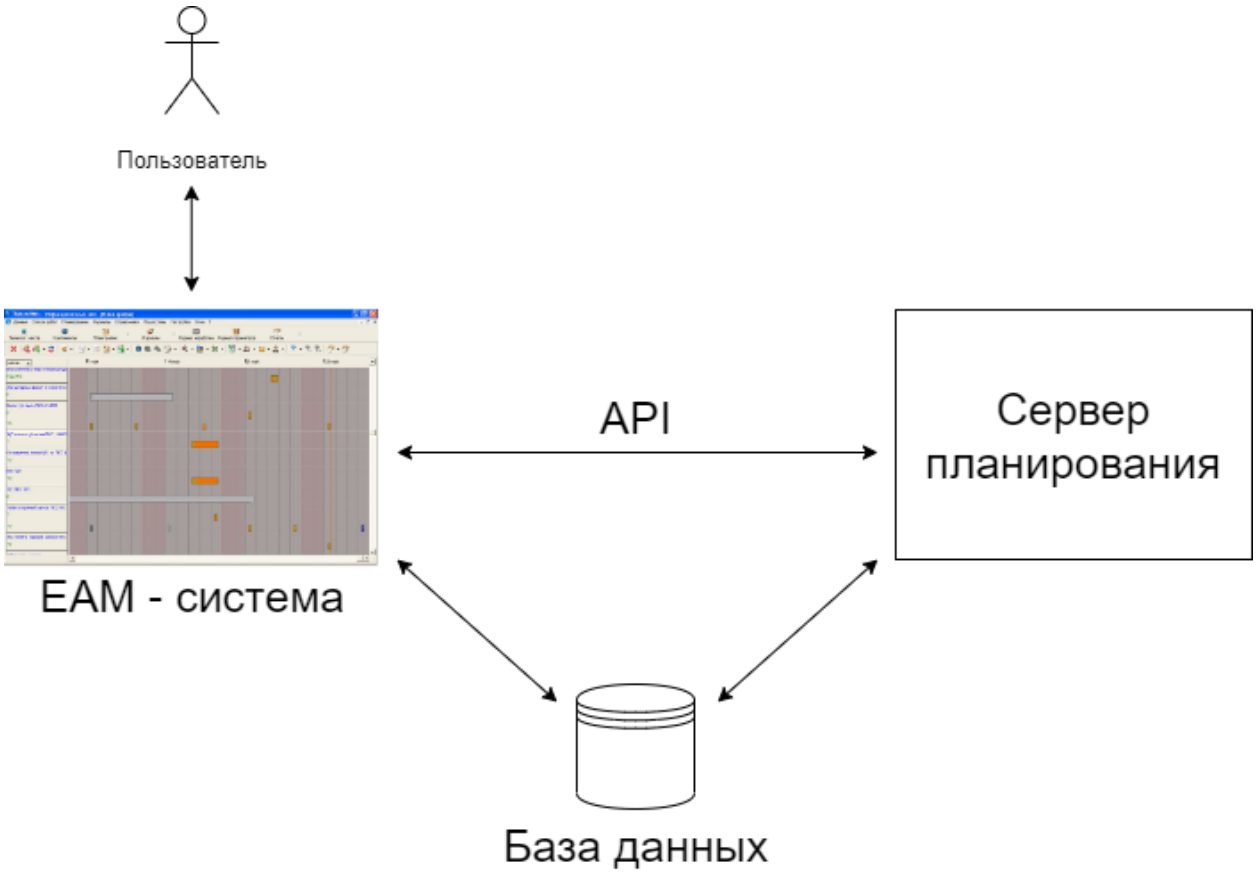

Рисунок 1 – Архитектура системы.

Архитектуру самого сервера планирования (рис. 2) можно представить как API, который посредством контроллера вызывает сервис перепланирования, использующий библиотеку OR-Tools.

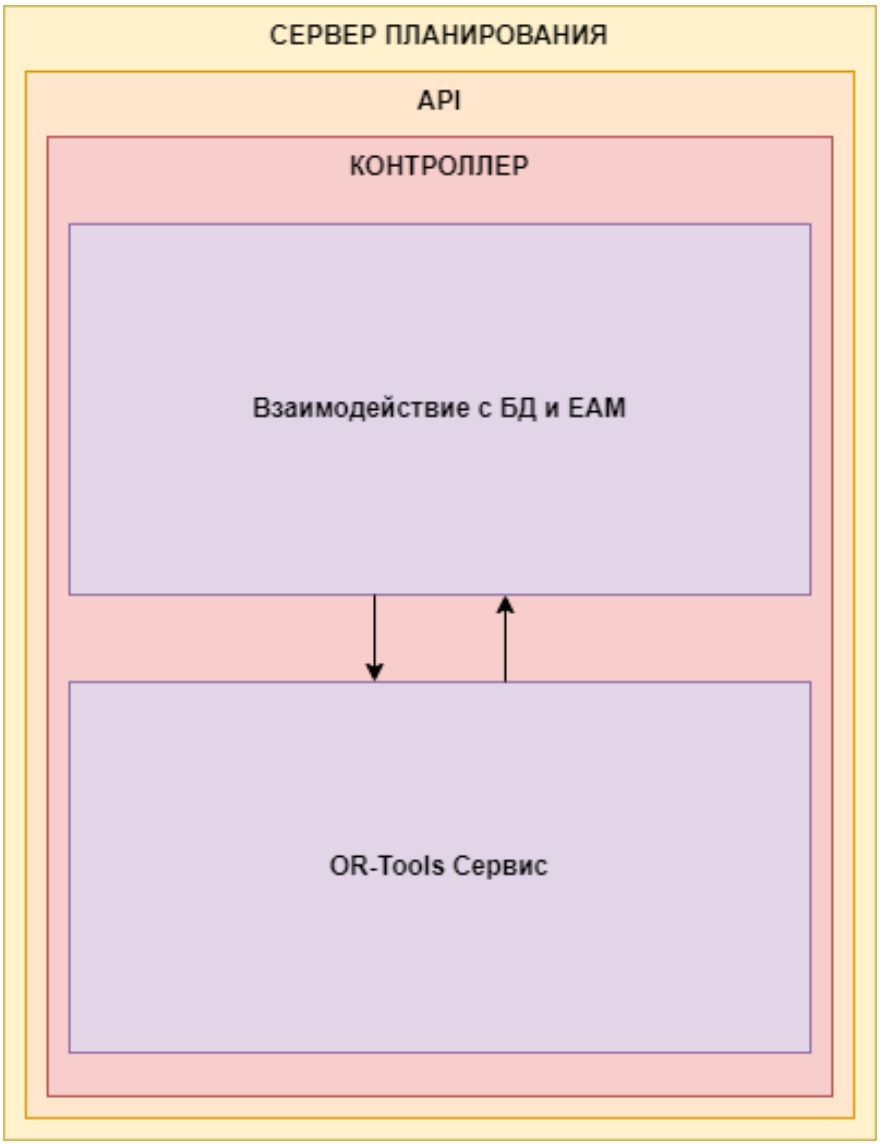

Рисунок 2 – Архитектура сервера планирования.

Рассмотрим представленную на рисунке 2 архитектуру сервера планирования более подробно.

Фронтенд (EAM-система) отправляет команду через API, в котором в привязанном к роуту контроллере выполняются определенные действия - инициализация перепланирования выбранных через фронтенд работ или их же, но уже перепланированных, отправка обратно во фронтенд (EAM-систему) на отображение и подтверждение.

Фактически самой главной частью представленного сервера планирования является сервис OR-Tools. Именно данный сервис отвечает за основную часть - непосредственно процесс планирования предварительно задан-

ных работ. В данном сервисе происходит инициализация модели, задание ограничений для этой модели, а также задается целевая функция, то есть функция, относительно которой будут искаться все удовлетворяющие ограничениям решения для данной модели.

2.2 Разработка прототипов алгоритмов (как отдельных компонентов и их взаимосвязей, так и в целом системных), структур данных.

2.2.1 Типы данных Google OR-Tools

Все объекты, для которых проводится процесс оптимизации, должны быть представлены в фреймворке Google OR-Tools одним из следующих типов данных:

- IntVar;
- BoolVar;
- IntervalVar;
- OptionalIntervalVar.

Тип данных IntVar это объект, который может принимать любое целочисленное значение в определенных диапазонах, такой диапазон называется доменом. В коде объявляется как

```
var integer_variable = model.NewIntVar(0, 100, "Integer Varible
    ");
```

где 0 и 100 это домен переменной, а "Integer Variable" - произвольное имя переменной, задаваемое пользователем.

Тип данных BoolVar представляет собой переменную для хранения булевых значений. Внутри фреймворка BoolVar аналогичен типу IntVar, но с доменом $[0, 1]$. Выделен в отдельный тип данных для удобства программирования.

Пример объявления:

```
var boolean_variable1 = model.NewBoolVar("Boolean Variable");
```

В качестве параметров при объявлении передается только произвольное имя переменной, так как домен по умолчанию лежит в пределах $[0, 1]$

Тип IntervalVar представляет собой интервальную переменную. Интервальная переменная - это и ограничение, и переменная. Она определяется тремя целочисленными переменными: start, size и end.

Она является ограничением, потому что внутри она обеспечивает выполнение того, что start + size == end.

Это также переменная, поскольку она может задаваться в определенных методах ограничения планирования, таких как: NoOverlap, NoOverlap2D, Cumulative.

Объявляется так:

```
var interval_variable = model.NewIntervalVar(start, size, end,
    "Interval Variable");
```

в которой start - предполагаемый момент начала интервала, size - размер (длина) интервала - обычно фиксированное целое число, end - предполагаемый момент окончания интервала.

В качестве значений start и end могут приниматься либо целые числа - в таком случае, например, суть оптимизации такой переменной заключается в нахождении её места на горизонте планирования, либо переменные типа IntVar - тогда оптимизация данной переменной будет зависеть и от также неизвестных переменных определенного домена.

Последним из доступных типов является OptionalIntervalVar - опциональный интервал. Этот отличапется от IntervalVar литералом is_present, который указывает, активна (то есть фактически присутствует в модели и участвует в процессе оптимизации) данная переменная или нет.

Пример:

```
var optional_interval_variable = model.NewOptionalIntervalVar(
    start, size, end, is_presernt "Optional Interval Variable");
```

в которой is_present представляет собой булевую переменную типа BoolVar.

Представленные выше типы данных определяются как часть модели. Модель это некторое условное пространство имён, которое содержит в себе

переменные-объекты модели, методы для работы с объектами, ограничения для них, а также методы для задания целевой функции модели.

Решение модели эквивалентно нахождению для каждой переменной единственного значения из множества начальных значений (называемого начальной областью), такого, что модель выполнима, или оптимальна, если вы предоставили целевую функцию.

Основными ограничениями для модели можно выделить:

– Add;
– AddNoOverlap;
– AddAllDifferent;
– AddImplication.

Метод Add добавляет ограничение в виде линейного выражения вида $x + y = z$, где x, y, z - целые числа или переменные типа IntVar.

Ограничение AddNoOverlap гарантирует, что все выбранные для него интервалы не пересекаются во времени.

AddAllDifferent заставляет все переменные иметь разные значения.

Метод AddImplication обеспечивает ограничение вида ЕСЛИ, ТО - $a => b$.

Класс LinearExpr позволяет производить над типами данных OR-Tools различные математические операции, такие как сумма, скалярное произведение, домножение на коэффициент - Sum(), ScaleProd(), Term().

Для задания целевой функции фреймворк предлагает два метода - maximize и minimize. Данные методы позволяют максимизировать или минимизировать некоторые необходимые параметры модели.

2.2.2 Определение требований к модели

Так как при планировании работ периоды занятости ресурсов могут накладываться друг на друга независимо от качества планирования, то ожидается, что не все работы смогут быть запланированы. Соответственно, цель

работы планировщика заключается в максимизации количества запланированных работ и выражается формулой:

$$\max \sum_{0}^{N} (n_0, ..., n_N) \tag{2.1}$$

где N — количество работ;

n — переменная домена $[0, 1]$, означающая присутствие работы.

Так как работы одного типа и работы, привязанные к одному человеку, никак не должны пересекаться, то должно так же выполняться следующее ограничение:

$$\forall p \in I \ \ \forall q \in I : p \neq q \implies V_p \cap V_q \neq \emptyset \tag{2.2}$$

где V_i — семейство интервалов;

p, q — интервал множества V.

для каждого рабочего колво работ запланированных <= необходимому

Добавить

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ (СТАРЫЙ)

1. Видеозапись Blender used at VFX studio, [Электронный ресурс], URL: https://www.youtube.com/watch?v=ZpfUJDxEfz4 (дата обращения 17.05.2020).

2. Видеозапись Blender 2.8 Facial motion capture tutorial, [Электронный ресурс], URL: https://www.youtube.com/watch?v=uNK8S19OSmA (дата обращения 17.05.2020).

3. Сайт программного обеспечения захвата движений FaceCap [Электронный ресурс], URL: https://apps.apple.com/us/app/face-cap-motion-capture/id1373155478 (дата обращения 20.05.2020).

4. Сайт программного обеспечения захвата движений iClone7 [Электронный ресурс]. URL: https://www.reallusion.com/iclone/ (дата обращения 20.05.2020).

5. Chris Conlan, The Blender Python API: Precision 3D Modeling and Add-on Development. Изд-во Apress, 2017. – 150с.

6. Сайт Blender 2.8 API reference [Электронный ресурс], URL: https://docs.blender.org/api/current/ (дата обращения 23.05.2020).

7. Сайт библиотеки компьютерного зрения OpenCV [Электронный ресурс], URL: https://opencv.org/ (дата обращения 23.05.2020).

8. Сайт библиотеки компьютерного зрения DLib [Электронный ресурс], URL: http://dlib.net/ (дата обращения 23.05.2020).

9. Using the Facemark API [Электронный ресурс], URL: https://docs.opencv.org/master/d7/dec/tutorial_facemark_usage.html (дата обращения 23.05.2020).

10. DLib Face landmark detection [Электронный ресурс], URL: http://dlib.net/face_landmark_detection.py.html (дата обращения 30.05.2020).

11. Видеозапись работы втроенных в Blender средств захвата движений [Электронный ресурс], URL: https://streamable.com/tm5s0 (дата обращения 30.05.2020).

12. Алгоритм Левенберга — Марквардта [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Алгоритм_Левенберга_—_Марквардта (дата обращения 01.06.2020).

13. Vahid Kazemi and Josephine Sullivan ”One Millisecond Face Alignment with an Ensemble of Regression Trees” KTH, Royal Institute of Technology Computer Vision and Active Perception Lab Teknikringen 14, Stockholm, Sweden 2014. – 8c.

14. Датасет Facial point annotations [Электронный ресурс], URL: https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/ (дата обращения 05.06.2020).

15. О введении в действие санитарно-эпидемиологических правил и нормативов СанПиН 2.2.2/2.4.1340-03 (в ред. от 03.09.2010) [Электронный ресурс], URL: https://www.eg-online.ru/document/regulatory/219156/ (дата обращения 05.06.2020).

16. Страница библиотеки opencv-contrib-python на сайте базы модулей языка Python PyPi [Электронный ресурс], URL: https://pypi.org/project/opencv-contrib-python/ (дата обращения 06.06.2020).

17. Страница библиотеки dlib на сайте базы модулей языка Python PyPi [Электронный ресурс], URL: https://pypi.org/project/dlib/ (дата обращения 06.06.2020).

18. Страница библиотеки numpy на сайте базы модулей языка Python PyPi [Электронный ресурс], URL: https://pypi.org/project/numpy/ (дата обращения 06.06.2020).

19. Сайт Trained model files for dlib example programs. [Электронный ресурс], URL: https://github.com/davisking/dlib-models (дата обращения 06.06.2020).