

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Алтайский государственный технический университет
им. И.И. Ползунова»

Факультет (институт) информационных технологий
Кафедра «Информатика, вычислительная техника и информационная безопасность»
Направление 09.04.01 Информатика и вычислительная техника
Направленность (профиль) Программно-техническое обеспечение автоматизированных систем
УДК 004.42

УТВЕРЖДАЮ
Заведующий кафедрой ИВТиИБ
А.Г. Якунин
подпись инициалы, фамилия
«_____» 2019 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

МД 09.04.01.12.000 ПЗ
обозначение документа

Разработка программного обеспечения для расчета параметров оптимизации
при выборе коммерческой клиники
тема магистерской диссертации

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Студент группы 8ИВТ-71 Сидоренко Евгения Александровна
№ группы фамилия, имя, отчество

Научный руководитель д.т.н., профессор каф. ИВТиИБ Л.И. Сучкова
должность, учёная степень инициалы, фамилия

Барнаул 2019

Реферат

Магистерская диссертация содержит 109 страниц, 43 рисунка, 40 используемых источников, 3 приложения.

Ключевые слова: многокритериальная оптимизация, сервис записи к врачу, метод главного критерия, метод аддитивной свертки критериев.

Объект исследования – сервис онлайн – записи в коммерческие клиники. Предмет исследования – методы решения задачи многокритериальной оптимизации. Цель работы: разработка программного обеспечения для расчета параметров оптимизации при выборе услуг коммерческой клиники.

Результатом работы является модуль, который позволяет оптимизировать процесс поиска предложений для записи к врачу. Программный продукт внедрен в сервис онлайн – записи к врачу TalonDo.

The abstract

Master's thesis contains 109 pages, 43 figures, 40 used sources, 3 applications.

Key words: multicriteria optimization, service appointment, the method of the main criterion, the method of additive convolution of criteria.

The object of the study is the service of online recording in commercial clinics. Subject of research – methods of solving the problem of multi-criteria optimization. Purpose: development of software for the calculation of optimization parameters in the selection of services of a commercial clinic.

The result of the work is a module that allows you to optimize the process of searching for proposals for an appointment with a doctor. A software product embedded in the online service appointment TalonDo.

Изм.	№ документа	Подпись	Дата	МД 09.04.01.12.000 ПЗ		
Разраб.	Сидоренко Е.А.			Разработка программного обеспечения для расчета параметров оптимизации при выборе коммерческой клиники	Лит	Лист
Пров.	Сучкова Л.И.				У	2
Н. контр.	Попкова А.И.					Листов
Утв.	Якунин А.Г.					109
АлтГТУ, ФИТ гр. 8ИВТ - 71						

Содержание

Перечень условных обозначений	5
Введение.....	6
1 Обзор и анализ методов оптимизации критериив для принятия решения.....	8
1.1 Процесс принятия решений	8
1.2 Постановка задачи многокритериальной оптимизации.....	11
1.3 Математическая модель многокритериальной оптимизации.....	12
1.4 Возможные проблемы, возникающие при использовании методов решения задач МКО.....	14
1.5 Оптимальность по Парето	15
1.6 Существующие методы решения задач многокритериальной оптимизации	17
2 Особенности реализации сервиса для выбора коммерческой клиники	28
2.1 Сервисы для выбора коммерческой клиники	28
2.2 Основные функции сервиса TalonDo.....	34
2.3 Структура web–сервиса.....	35
2.4 Структура базы данных сервиса TalonDo	36
2.5 Среда разработки и используемые технологии	58
3 Реализация модифицированных методов расчета параметров оптимизации	63
3.1 Метод аддитивной свертки	65
3.1.1 Описание алгоритма работы метода	65
3.1.2 Реализация метода	66
3.1.3 Пользовательский сценарий использования метода	70
3.2 Метод главного критерия.....	75
3.2.1 Описание алгоритма работы метода	75
3.2.2 Реализация метода	76
3.2.3 Пользовательский сценарий использования метода	77
4 Исследования разработанного модуля для расчета параметров оптимизации при выборе коммерческой клиники	82
Заключение	91

Список использованных источников	93
Приложение А Задание на выполнение магистерской диссертации	98
Приложение Б Акт о внедрении ПО	100
Приложение В Исходный код.....	101

Перечень условных обозначений

СНИЛС	страховой номер индивидуального лицевого счета застрахованного лица в системе персонифицированного учета Пенсионного фонда Российской Федерации
БД	база данных
ЛК	личный кабинет
ОМС	обязательное медицинское страхование
МО	медицинская организация
ФФОМС	федеральный фонд обязательного медицинского страхования
ОКМУ	общероссийский классификатор медицинских услуг
ИТ	информационные технологии
МИС	медицинская информационная система
ЛПУ	лечебно профилактическое учреждение
ПО	программное обеспечение
МКО	многокритериальная оптимизация
ГА	генетический алгоритм
ЛП	линейное программирование
ЛПР	лицо, принимающее решение
ТАП	талон амбулаторного пациента

Введение

В настоящее время интернет – технологии и различные интернет – сервисы широко распространены во всех сферах человеческой деятельности. Сфера медицины не осталась в стороне. Помимо программных комплексов для постановки диагнозов, обработки и хранения результатов анализов, диагностики различных заболеваний, популярность набирают и сервисы онлайн – записи на прием к специалистам частных клиник.

Использование электронного сервиса записи к врачу имеет множество преимуществ, таких как: экономия времени, конфиденциальность, отсутствие посредников в виде регистратуры, большой выбор специалистов и услуг.

Каждый человек время от времени оказывается в ситуации, когда достижение некоторого результата может быть осуществлено не единственным способом. Необходимая пациенту медицинская услуга также может быть оказана во многих клиниках, разными специалистами и по ценам, отличным друг от друга. Учитывая данный факт, онлайн сервис, целью которого является помочь клиенту в выборе медицинской услуги, должен содержать алгоритмы по нахождению оптимальных решений (вариантов услуг) для каждого конкретного пациента.

Однако в различных ситуациях для пациента наилучшими могут быть совершенно разные решения. Все зависит от выбранного или заданного критерия. Проблема оптимального выбора медицинской услуги является многокритериальной, так как варианты решения оцениваются при помощи частных критериев. Поскольку, как правило, каждый из критериев выделяет «свой» наилучший вариант, т.е. не бывает варианта, который одновременно является лучшим по каждому из критериев, то многокритериальные задачи принципиально сложнее однокритериальных и требуют для своего решения специальных методов и подходов. Выбор пациентом медицинского учреждения относится к классу многокритериальных задач.

Методы многокритериальной оптимизации более полно отражают задачи, ставящиеся перед реальными многосвязными системами. Многокритериальность объясняется тем, что при оценке действительно сложных ситуаций редко удается обойтись одним критерием. Перед лицом, принимающим решение, появляется проблема выбора наиболее подходящего метода многокритериальной оптимизации, и, как следствие, принятие допущений, условностей, границ теоретических аналогов реальных объектов [1].

Целью данной работы является разработка программного обеспечения для расчета параметров оптимизации при выборе услуг коммерческой клиники.

Объектом исследования является сервис онлайн – записи в коммерческие клиники.

Предмет исследования – методы решения задачи многокритериальной оптимизации.

Для достижения поставленной цели необходимо будет выполнить ряд задач:

- 1) провести аналитический обзор существующих методов решения задач многокритериальной оптимизации;
- 2) выполнить модификацию выбранных методов для адаптации к работе алгоритма сервиса;
- 3) спроектировать структуру основных модулей системы;
- 4) разработать программное обеспечение (модуль) для расчета параметров оптимизации при выборе услуг коммерческой клиники;
- 5) проанализировать эффективность применения разработанного модуля.

1 Обзор и анализ методов оптимизации критериев для принятия решения

1.1 Процесс принятия решений

Под принятием решений понимают процесс человеческой деятельности, направленный на выбор наилучшего из возможных вариантов. Фраза «принятие решений» довольно часто применяется в наиболее широком смысле. К примеру говорят, о принятии решений в следствии математических расчетов, произведенных компьютерами или роботами.

Принятие решения сложный процесс, который может быть рассмотрен как со стороны отдельных этапов, так и с точки зрения системы, учитывая явные и косвенные связи между этими этапами. Обнаружение, определение, описание и преподнесение данных связей в наглядном, удобном для восприятия варианте, считается первостепенной задачей при формировании модели процесса принятия решения.



Рисунок 1.1 – Классификация методов, используемых при принятии решений

Люди имеют возможность различными способами принимать участие в процессе принятия решений. Например, если человек отвечает за выбор наибо-

лее подходящего действия, то его называют лицом, принимающим решения (ЛПР).

Принятие конкретного решения, в общем случае, подразумевает выбор единственного из представленных вариантов действий, альтернатив. Формирование перечня альтернатив из всего множества представленных вариантов – обязательная составляющая проблемы, касающейся процесса принятия решений. Задачи принятия решений имеет место, если есть, по крайней мере 2 альтернативы. В случае, если альтернатив слишком много, для ЛПР не предоставляется возможным провести анализ по каждой представленной альтернатив, и в следствии этого возникает потребность во вспомогательных средствах для содействия в подборе решений.

Современная теория принятия решений подразумевает то, что виды решений описываются по различным характеристикам, признакам привлекательности для ЛПР. Данные показатели служат в виде критериев при выборе решения. Большинство задач содержит множество критериев. В свою очередь каждый критерий может быть как зависимым, так и не зависимым.

Сложность конкретной задачи обычно напрямую зависит от числа представленных в ней критериев. Если задача имеет немного критериев, то сравнение пары альтернатив является простой и прозрачной операцией. В таком случае критерии возможно сопоставить и выбрать предпочтительную альтернативу. Если задача содержит много критериев, то она становится необозрима ЛПР [38].

Обычно при рассмотрении процесса принятия решения, в нем выделяют несколько этапов:

- 1) Сбор всех доступных в момент принятия решения данных: теоретических и практических сведений, математические модели, различного рода опросы, определяются критерии, по которым происходит выбор решения и т.п.
- 2) Определение возможностей реализации и исполнения с учетом существующих ограничений.

3) Сопоставление альтернатив и отбор оптимального варианта, вариантов решения.

Создание математической модели осуществляется с целью её использования, в связи с дороговизной или невозможностью проверки решения на реальном объекте. Одними из основных требований к математической модели, является её соответствие, идентичность, соразмерность в рамках решаемой задачи. Теоретически, моделирование понимают, как способ познания реальности.

Классический аспект изучения операций подразумевает существование только одного критерия при оценке качества решения [26]. Тем не менее, из-за увеличения сферы использования методов исследований операций, специалисты начали оказываться в ситуациях, когда в задаче, значимыми являются несколько критериев оценки качества решения. В данных задачах прослеживается отличительная черта: модель, в которой описано множество возможных решений, является объективной, а качество оценивается несколькими критериями.

В связи с этим, при выборе оптимальных вариантов решений, нужен компромисс. Так как задача не содержит в себе данных для нахождения такого компромисса, следовательно, определить его, взяв за основу объективные расчеты не предоставляем возможно. Исследование многочисленных проблем, связанных с подобного рода задачами, привел к возникновению класса много-критериальных задач [25].

Данного рода задачи появляются в таких случаях, если имеет место существование нескольких первостепенных целей, невозможных для определения одним критерием. Необходимо предоставить точку в диапазоне возможных решений, посредством которой станет возможно минимизировать или максимизировать критерии.

Оптимизация используется во всех сферах жизнедеятельности человека. Оптимизация, это активность, направленная на достижения лучших результатов, при определенных обстоятельствах [19].

1.2 Постановка задачи многокритериальной оптимизации

Многокритериальные задачи могут иметь различное содержание, отличаясь по объему, а также подразумевать наличие разного качества информации. На текущий момент отсутствуют общепризнанная классификация многокритериальных задач и сформулированные практики по их решению. Большая неопределенность, в связи с недостаточным количеством информации при рассмотрении многокритериальных задач, является ключевой проблемой в их решении.

Формулировка задачи по оптимизации, подразумевает наличие нескольких конкурирующих качеств, параметров процесса, к примеру, в данной работе можно выделить следующие:

- a) удаленность клиники с необходимой услугой;
- b) цена услуги;
- c) предпочтительная половая принадлежность врача;
- d) стаж специалиста.

Определение компромиссной альтернативы среди всех представленных, являет собою операцию по решению задачи оптимизации.

Постановка задачи требует наличие цели и объекта оптимизации.

Стоит отметить, что для постановки задачи, также необходимо следующее:

- 1) Ресурсы оптимизации, подразумевается допустимость выбора значений характеристик оптимизируемого процесса;
- 2) Доступность приведения оптимизируемой величины к количественной оценке, так как лишь в данном варианте получится сопоставить результаты при выборе одних или других управляющих действий;
- 3) Учесть ограничения. Процесс постановление задачи оптимизации непременно должен содержать, кроме управляющих характеристик, ещё и установку границ на эти характеристики. Они накладываются в связи с техническими или экономическими убеждениями.

Параметр для количественной оценки оптимизируемого варианта работы объекта называется критерием оптимальности.

После определения критерия, на его основе конструируется главная функция, являющая собою взаимосвязь между критерием оптимальности и параметрами, влияющими на её значение.

Тип критерия оптимальности зависит от определенной задачи оптимизации.

Общая постановка задачи оптимальности, несет в себе представление критерия оптимальности, как экономической оценки (эффективность, первоначальная стоимость продукта, рентабельность, доход). Но в частном случае оптимизационных задач, где объект – часть процесса, выделять непосредственный экономический показатель, не всегда является целесообразным [2].

Если математическая при исследовании отсутствует, можно использовать сам объект, но в этом случае, оптимизация, проводимая опытным путем, несет в себе ряд весомых недостатков:

- a) необходимость в наличии реального объекта;
- b) необходимость изменения технологического режима, а это не всегда допустимо;
- c) увеличение времени проведения испытаний и сложность в обработке данных.

Математическая модель, которая в достаточной мере определяет, описывает процесс, помогает намного легче решить оптимизационную задачу численными или аналитическими методами.

1.3 Математическая модель многокритериальной оптимизации

Теория многокритериальной оптимизации, далее МКО, подразумевает под собой решение задач принятия решений единовременно, по нескольким критериям. Постановка задач МКО осуществляется надлежащим способом: не-

обходимо определить такие x_1, x_2, \dots, x_n , которые удовлетворяли бы ограничивающей системе

$$g_i(x_1, x_2, \dots, x_n) \leq b_i, \quad i = 1, 2, \dots, m \quad (1.1)$$

и значения функции

$$z_k = f_k(x_1, x_2, \dots, x_n), \quad k = 1, 2, \dots, K, \quad (1.2)$$

достигали максимума.

Допустимая область $D \subset R^n$ формируется из точек в диапазоне $X = (x_1, x_2, \dots, x_n)$, которые удовлетворяют ограничивающей системе (1.1). Альтернативами или допустимыми решениями называют элементы диапазона D , а критериями или целевыми функциями называют числовые функции f_k , $k = 1, 2, \dots, K$, определенными на диапазоне D . Задачи (1.1) – (1.2) подразумевают наличие K целевых функций. Данные функции проецируют множество $D \subset R^n$ в множество $F \subset R^K$, называемое множеством достижимости.

Математическая модель МКО (1.1) – (1.2) записывается в векторном виде следующим способом:

$$\vec{f}(X) = (f_1(X), \dots, f_K(X)) \rightarrow \max, \text{ при } X \in D, \quad (1.3)$$

где $\vec{f}(X)$ – вектор-функция значения $X \in D$.

Итальянский экономист Вильфредо Парето в 1904 году первым определил проблему МКО во время математического исследования товарообмена. После этого интерес связанный с проблемой МКО начал возрастать и был обусловлен технологическим развитием в сфере вычислительной техники. В связи с этим стало понятно, что возникновение многокритериальных задач проявляется, к примеру при моделировании больших инженерных систем.

Отличительной чертой от оптимизационных задач с единственным критерием, МКО подразумевает наличие некой неопределенности. Рассматривая задачи МКО с точки зрения математики, принято считать, что они могут иметь только компромиссное решение, и следовательно, считаются неопределенными.

В силу этого, термин оптимальность может быть истолкован разными способами, сама же теория имеет 3 ключевых направленности:

- 1) на разработку понятия оптимальности;
- 2) на доказательство наличия решения, в надлежащем смысле – оптимального;
- 3) на разработку методов поиска наиболее оптимального решения.

1.4 Возможные проблемы, возникающие при использовании методов решения задач МКО

При анализе методов решения задач многокритериальной оптимизации приходится учитывать специфические особенности каждого метода, во избежание представленных ниже проблем.

Несравнимость решений. Главная трудность логического анализа многокритериальных задач заключается в том, что, в отличие от «простых» (однокритериальных) задач, в многокритериальных задачах возникает эффект несравнимости вариантов (решений). Несравнимость решений представляет собой форму неопределённости, которая, в отличие от неопределённости, которая вызвана влиянием среды, связана со стремлением личности, которая принимает решение «достигнуть противоречивых целей» и может быть названа ценностной неопределённостью. Выбор среди несравнимых решений считается непростой концептуальной задачей и составляет основное содержание многокритериальной оптимизации [37].

Нормализация критериев. Так как частные критерии обладают разным физическим смыслом, т. е. измеряются в различных единицах; масштабы их не соизмеримы, по этой причине нельзя сопоставлять качества полученных результатов по каждому критерию. Процедура приведения масштабов локальных критериев к общему, как правило, безразмерному, называется нормализацией критериев.

Выбор принципа оптимальности. Речь идет об определении правила, которое позволило бы отметить то или иное решение лучшим среди остальных. Подбор принципа оптимальности – основная проблема векторной оптимизации. Формально описать принцип оптимальности (критерии "правильности решения") – является трудным, по ряду причин:

- 1) объекты, рассматриваемые теорией принятия решений до такой степени разнообразны, что определить единые принципы оптимальности для всех классов задач не представляется возможным;
- 2) цели участников процессов принятия решений – различны и часто противоположны;
- 3) критерии правильности решения находятся в зависимости не только от характера задачи, её цели и т. п., но и от того, насколько объективно они выбраны, в ином случае будет подготовка под ответ;
- 4) проблемы выбора решения могут скрываться и в самой постановке задачи, в случае если требуется достижение неосуществимых результатов [36].

Учёт приоритета критериев. Как правило, из физического смысла проблемы следует, что локальные критерии обладают разной значимостью (весом) при решении задачи, т. е. один локальный критерий имеет приоритет над другим локальным критерием. Это необходимо принимать во внимание при выборе принципа оптимальности и нахождении области возможных решений, отдавая предпочтение наиболее важным критериям [20].

1.5 Оптимальность по Парето

Состояние определенной системы, в котором значения всех критериев, в частности, не могут быть улучшены без ухудшения остальных, называется оптимальностью по Парето.

«Множество Парето» – это все состояния системы, оптимальные по Парето, также называют – «множество альтернатив, оптимальных в смысле Парето» или «множество Парето-оптимальных альтернатив». Применимо к альтернати-

вам, также используют определения: «компромиссных», «не улучшаемых» альтернатив.

Пусть X – множество возможных решений к какой-либо задаче, тогда допустимое решение должно удовлетворять условию $x \in X$. Допустим, что любое решение из допустимого диапазона оценивается по n критериям, где $n > 2$.

Существует вспомогательный механизм по решению многокритериальных задач – определение эффективного решения, либо оптимального по Парето.

Пусть $H_i(x)$, $x \in X$ –функция, с вещественными значениями, являющими оценками решения $x \in X$ по критерию i , $i = \overline{1, n}$.

Из этого следует, что вектор $H(x) = (H_1(x), \dots, H_n(x))$, $x \in X$ является набором оценок решения $x \in X$ по всевозможным критериям. Допустим, что предпочтительность решения $x \in X$ увеличивается с увеличением составляющих вектора H , следовательно, с ростом значения $H_i(x)$, улучшается решение x по критерию i , $i = \overline{1, n}$.

Парето – оптимальным называют решением $x^* \in X$, где нет еще одного решения $x \in X$, при котором выполняются условия

$$\begin{aligned} H_i(x) &\geq H_i(x^*), i = \overline{1, n}, \\ \exists i_0 : H_{i_0}(x) &> H_{i_0}(x^*) \end{aligned} \quad (1.4)$$

Другими словами, при наличии одного Парето – оптимального решения $x^* \in X$, не может существовать второго решения $x \in X$, которое бы превосходило x^* , хоть по одному критерию, и по оставшимся было бы не хуже.

Оптимальность по Парето не гарантирует, что операция является «самой лучшей». Но истинным является замечание, что такие операции не окажутся худшими. Для определения конкретного решения, принадлежащего Парето – оптимальному множеству, необходима дополнительная информация, данные [11]. Определение значений для Парето множества, не даёт понять, какое из решение является оптимальным, но данная операция существенно облегчает

использование алгоритмов, которые работают с данными, так как урезает множества допустимых вариантов [13, 14].

1.6 Существующие методы решения задач многокритериальной оптимизации

Методы, используемые для решения задач МКО, отражены на рисунке 1.2.

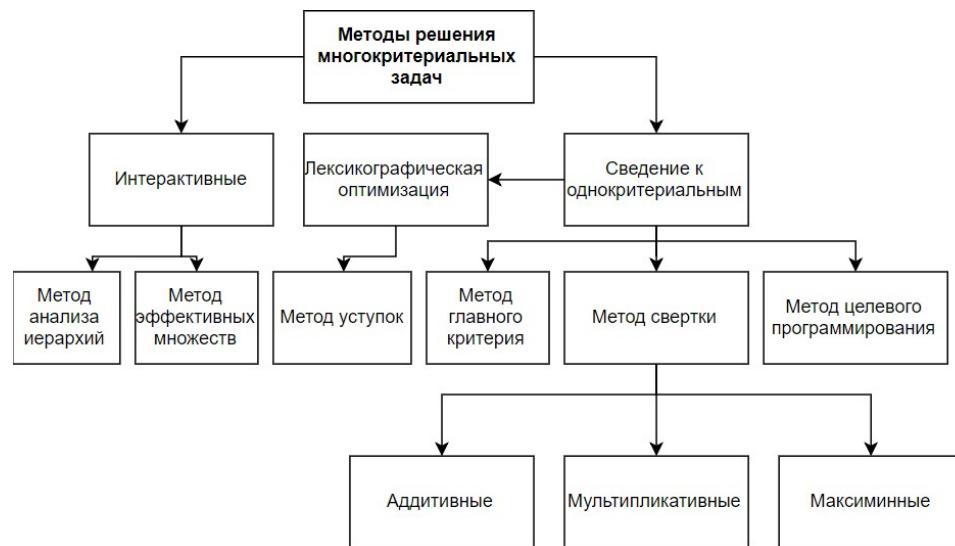


Рисунок 1.2 – Классификация методов решения задач МКО

Наиболее распространен, давно известен и всё еще используется чаще остальных метод, в основе которого лежит свертывание всех критериев в единственный, обобщенный (составной, комплексный, компромиссный и т.п.) критерий F. Данный критерий несет в себе сумму всех критериев, а именно весов значимости. Текущий метод – метод взвешенной суммы критериев (МВСК).

МВСК очень давно известен часто применяется в различных сферах исследовательской деятельности. Обширная популярность данного метода обусловлена несколькими факторами. Часть из них вызвана заманчивыми плюсами, такими как:

- 1) удобство математических расчетов;

2) адаптируемость под решение различных задач принятия решений, к примеру, найти единственное наилучшее либо множество лучших значений, отсортировать все значения по их предпочтительности, и т. д.

Также может быть обусловлена следующим:

- a) при использовании метода, не все имеют представление о нынешней теории по принятию решений для нескольких критериев, а также, в частности, о выводах научных анализов МВСК;
- b) по сравнению с МВСК, другие методы решения задач МКО требуют больше временных затрат и сил, для обработки данных о предпочтениях ЛПР;
- c) в связи с проведением анализа МКО только лишь на основании данных о предпочтениях ЛПР, объективные решения, которые можно взять за основу, как эталонные для сравнения результатов, полученных МВСК, отсутствуют;
- d) задачи МКО крайне многообразны, но не под все типы задач имеются эффективные методы решения.

Приступим к рассмотрению МВСК. Сперва обозначим исходные данные: формулы, определения. Допустим Z_i – множество значений, градаций критерия f_i . Затем для определенности станем полагать, что с увеличением значения любого критерия, предпочтения также увеличиваются. Любая отдельно взятая альтернатива, может быть охарактеризована значениями критериев $y_1 = f_1(x), \dots, y_m = f_m(x)$, из которых составляется векторная, или критериальная оценка варианта $y = (y_1, \dots, y_m)$. Определим множество всех вариантов, как X .

Для того чтобы сравнить предпочтительность вариантов необходимо взять за основу их векторные оценки. Очевидно, если при сравнении вариантов, одни из значений критериев больше у первого, а другие у второго, то появляются сложности. Чтобы их преодолеть, в соответствии с МВСК к векторному критерию применяется «свертывание» – определяется для рассмотрения взвешенная сумма критериев (при надобности проводится нормализация критериев):

$$F(f | w) = w_1 f_1 + \dots + w_m f_m, \quad (1.5)$$

где положительные числа w_i (как правило сумма равна единице) – коэффициенты важности, также называются весами, предназначение которых – учет различных относительных важностей, значимостей критериев или весомостей. Далее назначаются величины коэффициентов и все варианты x , при использовании формулы (1.5) характеризуются единственным значением взвешенной суммы критериев

$$F(f(x) | w) = w_1 f_1(x) + \dots + w_m f_m(x). \quad (1.6)$$

Наиболее предпочтительным вариантом является тот, которому соответствует наибольшее значение взвешенной суммы (1.6). Следовательно, оптимален тот вариант, сумма которого имеет наибольшее значение [7, 8].

Далее рассмотрим существующие варианты методов, основанных на свертке критериев.

Метод аддитивной свертки критериев. Аддитивную свертку критериев возможно трактовать, как реализацию принципа достоверной компенсации абсолютных значений нормированных частных критериев [12, 13, 27].

Допустим, что критерии соизмеримы, пронормированы и определен вектор весов для критериев $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_K)$, определяющих важность определенного критерия. Тогда, $\alpha_i \geq \alpha_j$, при критерии f_i имеющем приоритет над критерием f_j . И в свою очередь

$$\sum_{k=1}^K \alpha_k = 1, \quad \alpha_k \geq 0. \quad (1.7)$$

Далее, формируется новая целевая функция, для аддитивного метода

$$f(X) = \sum_{k=1}^K \alpha_k f_k(X), \quad (1.8)$$

а затем решается задача по оптимизации скалярного критерия

$$z = f(X) \rightarrow \max, \text{ при условии } X \in D. \quad (1.9)$$

Возможно доказать эффективность подобного рода решения (со скалярным критерием) задачи (1.3).

Данный вид свертки может быть использован в том случае, если ЛПР допускает компенсацию абсолютного уменьшения оценки по одному из критериев суммарным абсолютным ростом оценок по прочим критериям [3].

Данный метод можно адаптировать и использовать в сервисе онлайн – записи к врачу для поиска альтернативных вариантов с учетом приоритетов пользователя.

Метод мультипликативной свертки критериев. Для мультипликативного метода подход к решению аналогичен, только целевая функция имеет вид

$$f(X) = \prod_{k=1}^K f_k^{\alpha_k}(X), \quad (1.10)$$

$$\text{причем } \sum_{k=1}^K \alpha_k = 1, \quad \alpha_k \geq 0.$$

Основной и очень существенный недостаток методов свертывания критериев состоит в субъективности выбора коэффициентов α_k .

При использовании мультипликативных критериев не требуется нормировка частных критериев, и это является их преимуществом [4,5,6,7,8].

Данный метод так же подходит для использования в сервисе, так как является альтернативой метода аддитивной свертки критериев.

Агрегирование частных критериев используют также различные варианты агрегирования. В частности, если компенсация значений одних показателей эффективности другими недопустима, то используют функции агрегирования вида:

$$F(A_i) = \text{extr}_i(a_i f_i(A_i)) \quad (1.11)$$

Для каждого частного критерия, находится его нормированное значение и умножается на весовой коэффициент. А потом из всех полученных величин выбирается либо максимальное, либо минимальное значение.

Если первые m показателей надо увеличить, а остальные – уменьшить, то используют функцию агрегирования вида:

$$F(A_i) = \frac{\prod_{i=1}^m f_i^{a_i}(A_i)}{\prod_{i=m+1}^p f_i^{a_i}(A_i)} \quad (1.12)$$

В числители находятся произведение тех критериев, значение которых нам надо максимизировать, а в знаменателе находится произведение тех критериев, значение которых нам надо минимизировать. И поэтому мы получаем новый критерий, который нам надо будет максимизировать [1,8,13,19,28].

Методы свертывания критериев широко используются в решение задач многокритериальной оптимизации. Однако они имеют также проблемы и недостатки. В частности трудно обосновать выбор метода свертывания критериев, а от выбора метода часто зависит получаемый результат. Другим недостатком является трудность обоснования выбора весовых коэффициентов, часто для этого привлекаются эксперты, проводятся опросы, потом обрабатываются полученные результаты, однако это требует много времени и затраты других ресурсов. Еще одна проблема связана с тем, что эти методы, как правило, дают возможность компенсировать малые значения одних критериев большими значениями других, что часто бывает неприемлемо для конкретных решений [9,10,11,12].

Метод главного критерия. Выделяем главный критерий. Допустим, он равен $f_1(X)$. Оставшиеся целевые функции относим к разряду ограничений согласно представленному далее правилу.

Согласно требованиям ЛПР все критерии должны удовлетворять некоторым ограничениям. Определяем систему контрольных характеристик \tilde{f}_k , в соответствии с которой, значения по всем критериям, обязаны быть больше либо равны установленных значений \tilde{f}_k :

$$f_k(X) \geq \tilde{f}_k, \quad k = 1, 2, \dots, K. \quad (1.13)$$

Когда главный критерий выбран и заданы нижние границы оставшихся критериев, все готово к решению задачи однокритериальной оптимизации:

$$f_1(X) \rightarrow \max, \quad (1.14)$$

где

$$\begin{cases} f_k(X) \geq \tilde{f}_k, \quad k = 1, 2, \dots, K \\ X \in D \end{cases}. \quad (1.15)$$

Данный способ часто применяется в инженерной деятельности.

Метод главного критерия необходимо использовать для параметров, значения которых являются количественной мерой, например, себестоимость, объем производства, и др. В сервисе TalonDo в качестве параметров для метода главного критерия можно использовать:

- a) цену услуги (в рублях);
- b) средний рейтинг врача, оказывающего конкретную услугу (от 1 до 5);
- c) средний рейтинг клиники (от 1 до 5).

Плюсами метода главного критерия является легкость интерпретации полученных в ходе его применения результатов, отсутствие завышенных требований при подготовке экспертов в области математики, а также реализация данного метода не требует высоких вычислительных мощностей [13].

Для метода главного критерия, верно, следующее утверждение: решение задачи эффективно по Парето, если оно единственное. В случае, когда в задаче более одного допустимого решения, то их множество включает решения, которые являются эффективными по Парето [6,8,13,14]. Но это не гарантирует, что множество не содержит малоэффективные решения.

Исходя из этого процесс поиска решений состоит из двух этапов:

- 1) поиск множества возможных решений;
- 2) выбор конкретного наилучшего по Парето решения.

При использовании метода главного критерия на практике, зачастую критерии заданы в различных масштабах, в связи с этим приходится проводить нормировку данных критериев.

Нормировка критериев. Для приведения критериев к сопоставимому виду и обеспечения их эквивалентности используется нормировка:

$$f_n^{norm} = \frac{f_n(A_i) - \min(f_n(A_i))}{\max(f_n(A_i)) - \min(f_n(A_i))}, \quad (1.16)$$

$$\text{при } \min(f_n(A_i)) \neq \max(f_n(A_i)).$$

От значения критерия отнимаем минимальное значение этого критерия и делим полученную разность на разность между максимальным и минимальным значениями этого критерия [13]. Предполагается при этом, что минимальное и максимальное значение не совпадают. При такой нормировке, все нормированные значение $f_n^{norm}(A_i)$ будут лежать в интервале $[0; 1]$.

Метод последовательных уступок. Другой способ решения задачи МКО носит название метода последовательных уступок. В этом методе критерии нумеруются в порядке убывания важности [13]. Пусть критерии f_1, f_2, \dots, f_K записаны в порядке уменьшения их важности. Тогда должны быть выполнены следующие действия.

1–й шаг. Решается однокритериальная задача по 1–му критерию:

$$z_1^* = \max_{X \in D} f_1(X). \quad (1.17)$$

2–й шаг. Назначается разумная с инженерной точки зрения уступка Δz_1 , составляется и решается новая задача оптимизации по 2–му критерию:

$$z_2^* = \max_{\substack{X \in D \\ f_1(X) \geq z_1^* - \Delta z_1}} f_2(X). \quad (1.18)$$

3–й шаг. Назначается уступка для 2–го критерия Δz_2 , составляется и решается задача оптимизации по 3–му критерию:

$$z_3^* = \max_{\substack{X \in D \\ f_1(X) \geq z_1^* - \Delta z_1 \\ f_2(X) \geq z_2^* - \Delta z_2}} f_3(X). \quad (1.19)$$

Процесс назначения уступок по каждому критерию и решения однокритериальных задач продолжается, пока не дойдем до последнего K – го шага.

K – й шаг. Назначается уступка для $K - 1$ – го критерия Δz_{K-1} , составляется и решается задача оптимизации по последнему K – му критерию:

$$z_K^* = \max_{\substack{X \in D \\ f_1(X) \geq z_1^* - \Delta z_1 \\ f_2(X) \geq z_2^* - \Delta z_2 \\ \dots \\ f_{K-1}(X) \geq z_{K-1}^* - \Delta z_{K-1}}} f_K(X). \quad (1.20)$$

Основной недостаток методов, использующих ограничения на критерии, состоит в субъективности выбора контрольных показателей и в субъективности выбора уступок. При использовании метода последовательных уступок следует помнить, что уступки могут быть несравнимы между собой, поэтому надо предварительно организовать нормализацию критериев.

Существенным недостатком метода последовательных уступок является также и то, что решение, полученное этим методом, может оказаться не Парето – оптимальным [14].

Методы целевого программирования. Название этой группы методов связано с тем, что ЛПР задает определенные цели $\bar{f}_1, \bar{f}_2, \dots, \bar{f}_K$ для каждого критерия. Задача МКО в этом случае преобразуется в задачу минимизации суммы отклонений с некоторым показателем p :

$$z = \left(\sum_{k=1}^K w_k |f_k(X) - \bar{f}_k|^p \right)^{\frac{1}{p}} \rightarrow \min, \text{ при } X \in D, \quad (1.21)$$

где w_k – некоторые весовые коэффициенты, характеризующие важность того или иного критерия.

Задачу (1.21) можно конкретизировать в зависимости от значений параметра p и заданных целей. В частности, при $p = 2$ и $w_k = 1$ получим задачу минимизации суммы квадратов отклонений:

$$z = \sqrt{\sum_{k=1}^K |f_k(X) - f_k^*|^2} \rightarrow \min \text{ при } X \in D, \quad (1.22)$$

в которой минимизируется евклидово расстояние от множества достижимости F до «абсолютного максимума» $f^* = (f_1^*, f_2^*, \dots, f_l^*)$ в пространстве критериев. Здесь $f_k^* = \max_{X \in D} f_k(X)$.

Осложнения, обусловленные несоизмеримостью величин $|f_k(X) - f_k^*|$, можно преодолеть с помощью нормализации критериев, рассматривая следующую задачу оптимизации:

$$z = \sqrt{\sum_{k=1}^K \left(\frac{|f_k(X) - f_k^*|}{f_k^*} \right)^2} \rightarrow \min \text{ при } X \in D. \quad (1.23)$$

Методы гарантированного результата. Рассматриваются методы, которые дают наилучший результат даже для самого наименьшего из критериев, а именно, компромиссное решение находится путем решения следующей задачи оптимизации:

$$z = \min_{k=1,2,\dots,K} f_k(X) \rightarrow \max \text{ при } X \in D. \quad (1.24)$$

Это есть максиминная задача. С учетом нормализации критериев методы гарантированного результата образуют наиболее перспективное направление в решении задач МКО.

Для нормализованных критериев $\lambda_k(X) = \frac{f_k(X)}{f_k^*}$, где $f_k^* = \max_{X \in D} f_k(X)$,

максиминная задача формулируется в виде:

$$z = \min_{k=1,2,\dots,K} \lambda_k(X) \rightarrow \max, \text{ при } X \in D. \quad (1.25)$$

Остановимся на рассмотрении двух случаев, когда критерии равнозначны и неравнозначны (с заданным приоритетом).

Равнозначные критерии. Задача (1.25) эквивалентна задаче

$$z = \lambda \rightarrow \max \quad (1.26)$$

при условиях

$$\begin{cases} \lambda \leq \lambda_k(X), k = 1, 2, \dots, K \\ X \in D \end{cases}. \quad (1.27)$$

Задача (1.26)–(1.27) называется λ – задачей. Она имеет линейную целевую функцию и $m + K$ ограничений. Если все функции f_k и g_i линейны, то λ – задача относится к линейному программированию. В этом случае доказано, что оптимальное решение X^* λ – задачи оптимально по Парето.

В нашем случае, при поиске предложений по услугам в сервисе записи к врачу критерии поиска не могут быть равнозначными. Значимость каждого критерия определяет пользователь.

Критерии с заданным приоритетом. Рассмотрим только два критерия $f_1(X)$ и $f_2(X)$, и пусть $\lambda_1(X)$ и $\lambda_2(X)$ – соответствующие нормализованные критерии. Разобьем допустимую область на две части $D = D_1 \cup D_2$ так, что в области D_1 выполняется неравенство $\lambda_1(X) > \lambda_2(X)$, то есть первый критерий имеет приоритет над вторым, а в области D_2 выполняется неравенство $\lambda_1(X) \leq \lambda_2(X)$, то есть второй критерий имеет приоритет над первым.

Для числовой характеристики приоритета вводится коэффициент связи $p(X)$: $\lambda_1(X) = p(X)\lambda_2(X)$, который показывает, во сколько раз относительная оценка $\lambda_1(X)$ больше $\lambda_2(X)$. Если X^* – оптимальная точка для равнозначных критериев, то $p(X^*) = 1$.

Если X_1^* – точка оптимума по 1 – му критерию, то $\lambda_1(X_1^*) = 1$, $\lambda_2(X_1^*) < 1$, то есть $X_1^* \in D_1$, и значит $p(X_1^*) > 1$. Аналогично, если X_2^* – точка оптимума по 2–му критерию, то $\lambda_1(X_2^*) < 1$, $\lambda_2(X_2^*) = 1$, и значит $p(X_2^*) < 1$.

Предположим, что первый критерий имеет приоритет над вторым. Тогда коэффициент $p(X)$ необходимо задать в интервале $(1; p(X_1^*))$, а затем составить и решить λ – задачу, включив в систему ограничений равенство $\lambda_1(X) = p(X)\lambda_2(X)$. В результате получим точку X^* , которая будет принадлежать множеству D_1 , где 1 – ый критерий имеет приоритет над 2 – ым.

Доказано, что для выпуклых задач МКО точка X^* , являющаяся решением λ – задачи, единственная и оптимальна по Парето.

Недостаток рассмотренного метода состоит в субъективности задания коэффициента связи $p(X)$.

Решение задачи МКО методом гарантированного результата, как правило, проходит следующие этапы.

- 1) Разработка математической модели системы на основе заданных целей и ограничений; при этом часто используется мнение экспертов.
- 2) Предварительный анализ системы отдельно по каждому критерию; используются методы и программные средства оптимизации с одним критерием.
- 3) Нормализация критериев.
- 4) Решение задачи МКО при равнозначных критериях.
- 5) Задание приоритетов критериев и решение задачи МКО с назначенными приоритетами.

Таким образом, проанализировав вышеуказанные методы, было решено реализовывать метод аддитивной свертки и метод главного критерия.

2 Особенности реализации сервиса для выбора коммерческой клиники

Сервис TalonDo предоставляет возможность записаться на прием к врачу любой специализации. Регистрация записи занимает всего несколько минут. На сайте присутствует поиск медицинских учреждений по названию, врачу, адресу местонахождения филиала клиники или предоставляемой услуге. Для посетителей сайта доступна дополнительная информация, которая включает в себя, отзывы о медицинских учреждениях и специалистах, их рейтинг, доступные медицинские услуги. На рисунке 2.1 представлена главная страница сервиса.

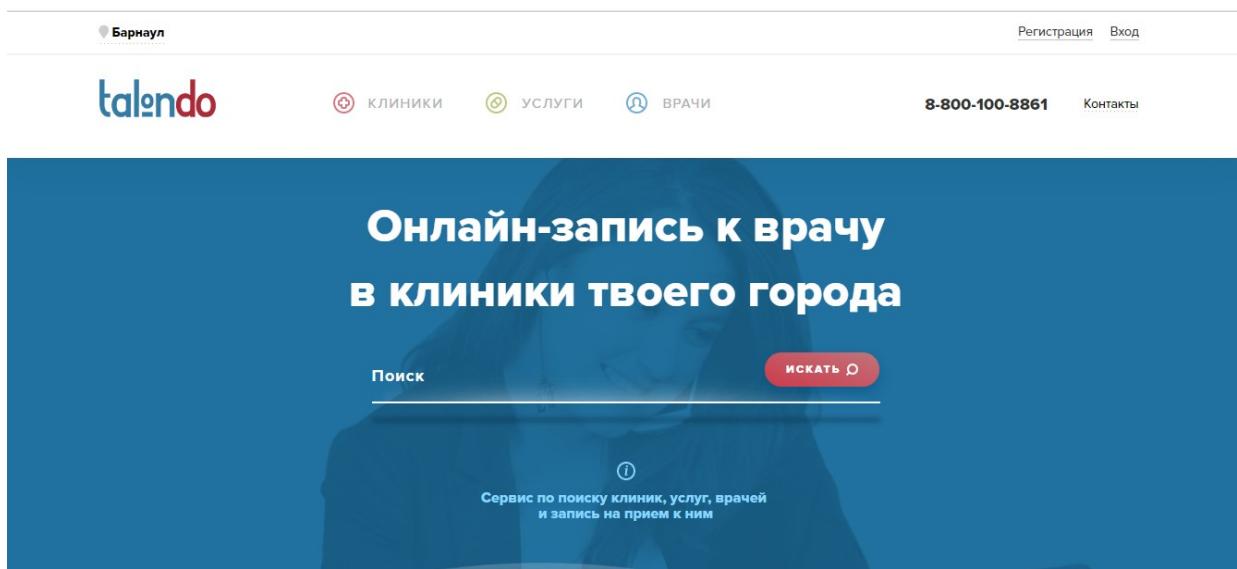


Рисунок 2.1 – Главная страница web–сервиса TalonDo

2.1 Сервисы для выбора коммерческой клиники

На данный момент существуют готовые решения. Рассмотрим аналогичные сервисы записи к врачу для выявления основные достоинств и недостатков:

1) Healthengine. Сайт – healthengine.com.au. Сервис предоставляет возможность записи на прием к различным специалистам, разбитым на группы в зависимости от профессиональной деятельности. После выбора необходимой группы специалистов осуществляется переход к выбору клиники. Клиники

можно отсортировать по местоположению, наличию специалиста определенного пола. Далее система предлагает выбрать желаемое время и дату приема. Затем предоставляется возможность ввести дополнительную информацию о клиенте и о бронировании, и перейти на страницу подтверждения бронирования. Результатом работы является запись на прием к специалисту. Основные достоинства системы – это дружественный, интуитивно понятный интерфейс и система защиты персональных данных.

2) Медоблако. Сайт – medoblako.ru. Медицинский портал Медоблако — это площадка, чьей основной задачей является предоставление пользователям информационных услуг. Данные, размещенные на сайте, проходят необходимую проверку. Воспользовавшись порталом Медоблако, Вы сможете посетить врача в любом медучреждении по вашему выбору. В базе данных портала находится 2500 медицинских учреждений и 7500 медицинских услуг. Клиники, которые представлены на сайте <http://medoblako.ru>, готовы предоставить все требующиеся медуслуги: от консультации специалистов до терапии. В качестве достоинств портала можно выделить наличие информации о клиниках во многих городах и областях РФ (Москва, Московская область, Санкт–Петербург, Ленинградская область, Белгород и др.), а также возможность записи к специалисту в указанных городах.

3) Яндекс.Здоровье. Сайт – health.yandex.ru. Яндекс.Здоровье — сервис, с помощью которого можно записаться к врачу без похода в регистратуру или посещения сайта госуслуг. Также можно пообщаться с врачом онлайн с помощью видео–вызова или онлайн–чата.

4) DocDoc. Сайт – docdoc.ru. DocDoc — бесплатный российский интернет–сервис по поиску врачей. Специализируется на предоставлении информации о медицинских специалистах, а также предлагает помочь пациентам в подборе врачей. Проект заявляет, что его задача — сделать работу врачей и клиник более прозрачной для потребителей, дать пациентам возможность найти врачей, а также упростить процесс записи на прием к выбранному доктору. Поиск врача возможен по следующим параметрам: опыт, география приема, цена

и отзывы пациентов, побывавших у доктора. База портала насчитывает более 21 000 врачей. Одной из обязательных и самых важных функций портала является сбор отзывов о врачах, на приеме которых побывали пациенты. Операторы call-центра звонят каждому записавшемуся через портал пациенту и задают вопросы о его впечатлениях от посещения доктора и клиники. Все собранные отзывы публикуются на сайте. Отзывы, которые посетители оставляют на сайте самостоятельно, проходят обязательную проверку и только потом публикуются на портале. Мнение пациентов, побывавших на приеме у врача, напрямую влияет на его рейтинг на портале DocDoc.

5) Meds. Сайт – meds.ru. «Meds» — это онлайн–сервис, с помощью которого можно не выходя из дома записаться на прием к врачу, или же на прохождение обследования. Этот сервис учитывает местоположение человека, что позволяет подобрать клинику, или другое медицинское учреждение максимально близко к дому. Кроме того, на сайте существует функция сортировки полученных результатов поиска по ценовой категории. Это очень удобно, так как все пациенты имеют индивидуальные финансовые возможности. Стоит отметить, что перечень обследований и специалистов, к которым можно записаться, достаточно широкий. Существует возможность записаться также на прием к более узким специальностям, таким как аллерголог, иммунолог, фтизиатр, рефлексотерапевт. Преимуществом этого сайта является то, что у пациента имеется возможность узнать подробную информацию о враче, который будет проводить прием, а также прочитать отзывы о его уровне квалификации и компетентности.

6) Medesk. Сайт – medesk.ru. Medesk – медицинская информационная система, в которой совмещены технологии и медицина, чтобы управление частной клиникой стало прозрачным и простым. Медицинская информационная система создана врачами для врачей совместно с программистами. Понятный интерфейс МИС Medesk позволяет быстро адаптироваться к медицинской системе и начать работать в ней уже с первого дня подключения. Простая структура системы позволяет снизить вероятность ошибок персонала клиники. Спе-

циалисты медицинского центра не тратят время и нервы на рутину — они более доброжелательны к пациентам. А пациенты, в свою очередь, становятся лояльнее к клинике. Данная система относится к классу МИС, а это значит, что запись на прием к врачу не является ее основной функцией. Задача МИС – управление и оптимизация деятельности ЛПУ.

7) Medihost. Сайт – medihost.ru. Портал предоставляет следующие возможности:

- a) запись на прием к специалисту (в том числе и узкой специализации), не выходя из дома (посредством телефона или интернета);
- b) получение онлайн консультаций врача;
- c) постоянно обновляющиеся медицинские статьи на все интересующие пациентов темы, написанные профессионалами;
- d) учет пациентов, интеграция с МИС;
- e) размещение рекламы клиник и медицинских центров и многое другое.

Важные разделы портала:

- a) Клиники. Можно искать нужное лечебное заведение по названию, метро, e-mail, карте, адресу.
- b) Врачи. На портале собрана база лучших специалистов в различных областях медицины. Для выбора нужного врача достаточно указать заболевание и система сама выдаст список необходимых специалистов, к которым можно тут же записаться на прием.
- c) Отзывы. Здесь можно поделиться своим мнением о качестве оказанных услуг и почитать отзывы об интересующем вас медицинском учреждении от других пациентов.
- d) Запись. Открыв данную вкладку можно записаться на прием к нужному специалисту.
- e) Консультации. Здесь врачи различных специальностей бесплатно консультируют посетителей портала. Для получения консультации необходимо просто зарегистрироваться на сайте и задать свой вопрос.

f) Цены. В этом разделе можно сравнить цены в клиниках на интересующие пациента услуги и выбрать наиболее приемлемые для себя.

g) Инфо. Здесь располагается информация о проводимых в клиниках акциях. Кроме того, нажав на интересующее предложение, можно распечатать бесплатный купон, дающий право на какие-либо привилегии.

8) НаПоправку. Сайт – napopravku.ru. Napopravku.ru – сервис для удобного выбора врача и эффективного взаимодействия с ним. Он помогает решить базовую проблему человека, когда он заболел: найти хорошего врача и вылечиться быстрее. Сервис состоит из 3 блоков:

a) «ВРАЧИ» – выбор врача по специальности, отзывам, району и другим параметрам;

b) «КЛИНИКИ» – выбор клиник по району, отзывам, специализации, уровню цен и т.д.;

c) «БИБЛИОТЕКА» – полезные материалы о болезнях и медицине доступным языком.

Главные особенности Napopravku:

a) На сайте представлены государственные врачи и клиники, а не только частные;

b) Реальные развернутые отзывы – не только хорошие, но и плохие, с деталями и нюансами;

c) Удобный поиск по всем нужным параметрам – а не каталог, которым сложно пользоваться;

d) Верифицированный контент доступным языком – разумный баланс деталей и понятности;

e) Увязка всех блоков друг с другом – посетитель решает вопросы «под ключ» на одном ресурсе.

9) Медрег22. Сайт – medreg22.ru. С середины мая у жителей Алтайского края появилась альтернативная возможность онлайн-записи на прием к врачу через медрег22.ru. Ранее, чтобы воспользоваться услугами электронной регистратуры, необходимо было осуществить вход в личный кабинет пациента

на краевом ресурсе медрег22.рф (рег22.рф), используя номер своего медицинского полиса. Сегодня для записи на прием к врачу в электронном виде и входа в личный кабинет пациенту достаточно лишь ввести пароль от портала «Госуслуги».

10) MEDBOX. Сайт – medbox.ru. С помощью приложения Medbox вы можете записаться на прием к нужному вам специалисту, вести свою медицинскую карту в удобном виде и получить консультацию в режиме online. Более 60 000 пользователей Medbox уже нашли своего специалиста. Через Medbox вы сможете:

- a) Записаться к врачу online. — найти нужного врача рядом с домом, выбрав его специальность; — получить информацию о расписании врачей, времени работы клиники; — записаться к нужному врачу или отменить прием;
- b) Хранить и добавлять данные в свою медкарту, которые будут доступны Вам и Вашему врачу в телефоне в удобном виде;
- c) Записаться на видео–консультацию с нужным специалистом и получить рекомендации не выходя из дома;
- d) Все данные хранятся на защищенных серверах.

После проведения анализа всех вышеуказанных сервисов были выявлены следующие недостатки:

- a) описанные сервисы используют для поиска предложений метод полного совпадения. Несмотря на наличие нескольких критериев поиска оптимального предложения, данные приложения не используют методы решения задачи МКО для нахождения наиболее подходящего варианта для ЛПР;
- b) эксплуатируется некоторых сервисов не производится на территории Российской Федерации и доступна только для англоязычных пользователей;
- c) избыточность функциональных возможностей пользователя, что пагубно сказывается на эффективности использования сервиса записи;
- d) услуги сервиса предоставляются платно;

- e) географический охват достаточно ограничен, например, отсутствует информация о клиниках Алтайского края и города Барнаул;
- f) информация о ЛПУ, врачах, услугах не актуальна, или не доступна;
- g) возможность записаться на прием отсутствует, на сервисе имеется лишь информация о клинике;
- h) некоторые описанные решения предоставляют возможность записи только в государственные клиники, в то время как существует множество частных МО.

2.2 Основные функции сервиса TalonDo

Ниже представлены основные функции сервиса TalonDo:

- 1) возможность записи пациента на приём к выбранному врачу, в соответствии с существующим расписанием этого врача;
- 2) возможность просмотра данных о клинике (медицинской организации), услугах и ценах, врачах, расписании врача или группы специалистов;
- 3) возможность просмотра расписания приёмов специалиста или филиала организации (актуального на текущий момент времени);
- 4) возможность генерации и печати талона амбулаторного пациента (ТАП);
- 5) возможность создания отзыва и просмотр рейтинга организации;
- 6) возможность создания отзыва и просмотр рейтинга врачей клиники;
- 7) возможность ведения реестров и генерации отчетов по количеству оказанных услуг организацией или конкретными специалистами.

Дополнительные функции сервиса:

- a) возможность создания личного кабинета для клиента, организации;
- b) возможность просмотра данных о прошлых бронированиях;
- c) возможность добавление статей для посетителей сервиса на различные медицинские темы;

- d) возможность интеграции со сторонними медицинскими системами для обмена актуальными данными о расписании, услугах и др.

2.3 Структура web–сервиса

В связи с тем, что сервис TalonDo имеет сложную структуру, было принято решение разделить приложение на пять составляющих – модулей (рисунок 2.2).

```
<modules>
    <module>domain</module>
    <module>web</module>
    <module>service</module>
    <module>repository</module>
    <module>mis-api</module>
</modules>
```

Рисунок 2.2 – Модули приложения

Модуль `domain` используется для хранения основных сущностей системы. Для каждой сущности или таблицы в базе данных соответствует отдельный класс.

Модуль `repository` – модуль преобразования объектов класса, запросов в SQL–запросы. Также данный модуль отвечает за взаимодействие с базой данных.

Модуль `service` включает в себя методы для работы с сущностями, определенными в модуле `domain`, с бизнес логикой логистического процесса, также осуществляет связь с репозиториями системы.

Модуль `service` включает в себя методы для работы с сущностями, определенными в модуле `domain`, с бизнес – логикой процесса, также осуществляет связь с репозиториями системы.

Модуль `mis–api` служит для интеграции сервиса TalonDo с внешними медицинскими системами. Данный модуль содержит методы для актуализации расписания, услуг, врачей, филиалов внешней клиники и т.п.

Модуль web включает в себя контроллеры по работе с web-страницами, статические ресурсы системы, такие как: файлы каскадных таблиц стилей, файлы скриптов, картинки и т.п.

2.4 Структура базы данных сервиса TalonDo

Таблицы в базе данных логически разбиты на схемы. На рисунке 2.3 представлена схема Basic базы данных сервиса TalonDo.

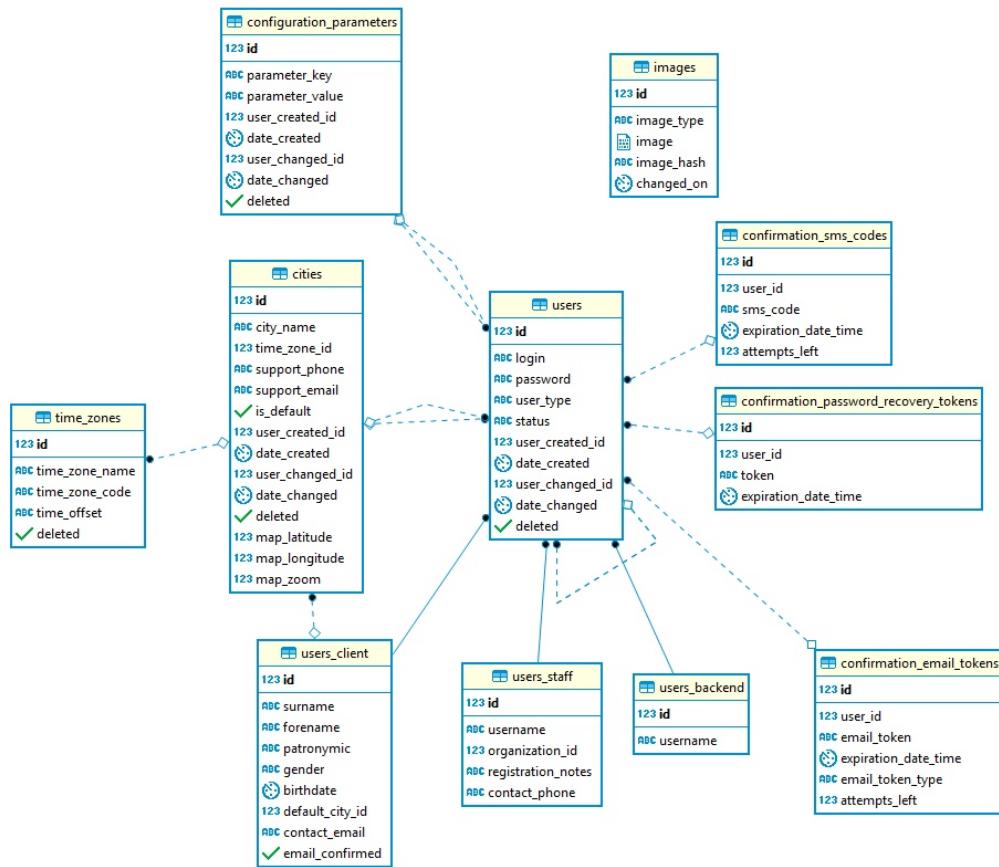


Рисунок 2.3 – Таблицы схемы Basic

Таблица **Users** – список зарегистрированных пользователей, различных типов. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) login – логин пользователя для авторизации (email или номер телефона);
- 3) password – пароль пользователя;

- 4) user_type – тип пользователя;
- 5) status – статус пользователя (активный/неактивный);
- 6) user_created_id – идентификатор пользователя, который создал конкретного пользователя;
- 7) date_created – дата создания пользователя;
- 8) user_changed_id – идентификатор пользователя, который изменил конкретного пользователя;
- 9) date_changed – дата изменения записи о пользователе;
- 10) deleted – флаг, который показывает был ли пользователь удален.

Таблица users_staff – пользователи типа «Сотрудник организации». Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) username – имя пользователя;
- 3) organization_id – идентификатор организации;
- 4) contact_phone – контактный телефон;
- 5) registration_notes – описание созданного пользователя при регистрации.

Таблица users_backend – пользователи типа «Администратор» или «Техническая поддержка». Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) username – имя пользователя.

Таблица users_client – пользователи типа «Клиент». Список пользователей с типом клиент – конечные пользователи сервиса. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) surname – фамилия клиента;
- 3) forename – имя клиента;
- 4) patronymic – отчество клиента;
- 5) gender – пол клиента;
- 6) birthdate – дата рождения;

- 7) default_city_id – идентификатор города по умолчанию;
- 8) contact_email – контактный адрес почты, для отправки оповещений и для авторизации, при наличии подтверждения;
- 9) email_confirmed – флаг, который показывает подтвержден ли почтовый адрес клиента.

Таблица images – изображения системы. Список всех изображений, которые имеются в системе. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) image_type – тип изображения (аватар доктора, логотип организации и др.);
- 3) image – изображение в виде двоичного массива;
- 4) image_hash – хэш изображения;
- 5) changed_on – дата изменения изображения.

Таблица confirmation_password_recovery_tokens – токены восстановления пароля. Токен – сгенерированный уникальный код, добавляемый в URL для подтверждения пользовательской информации. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) user_id – идентификатор пользователя, который использовал данный токен;
- 3) token – значение токена;
- 4) expiration_date_time – дата и время истечения срока действия токена.

Таблица confirmation_email_tokens – токены подтверждения электронной почты. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) user_id – идентификатор пользователя, который использовал данный токен;
- 3) email_token – строковое значение токена;

- 4) expiration_date_time – дата и время истечения срока действия токена;
- 5) attempts_left – количество пройденных попыток отправки кода подтверждения. Данное поле используется для временной блокировки отправки кода email подтверждения;
- 6) email_token_type – тип токена. Данное поле используется для хранения типа подтверждения (подтверждение бронирования/ подтверждение регистрации).

Таблица confirmation_sms_codes – коды подтверждения sms. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) user_id – идентификатор пользователя, который получил sms;
- 3) sms_code – значение кода;
- 4) expiration_date_time – дата и время истечения срока действия кода;
- 5) attempts_left – количество пройденных попыток отправки кода подтверждения. Данное поле используется для временной блокировки отправки кода sms подтверждения.

Таблица configuration_parameters – параметры системы. Список параметров для настройки системы, которые может изменять пользователь с типом «Администратор». Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) parameter_key – системное название параметра;
- 3) parameter_value – значение параметра;
- 4) user_created_id – идентификатор пользователя, который создал параметр;
- 5) date_created – дата создания параметра;
- 6) user_changed_id – идентификатор пользователя, который изменил параметр;
- 7) date_changed – дата изменения записи о параметре;

- 8) deleted – флаг, который показывает был ли параметр удален.

Таблица cities – города. Список городов, которые могут быть выбраны клиентом как город по умолчанию, и в которых находятся медицинские организации. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) city_name – название города;
- 3) time_zone_id – идентификатор часового пояса;
- 4) support_phone – телефон службы поддержки;
- 5) support_email – почтовый адрес службы поддержки;
- 6) is_default – флаг, который определяет, является ли город городом по умолчанию. В системе предусмотрен только один город по умолчанию;
- 7) user_created_id – идентификатор пользователя, который создал город;
- 8) date_created – дата создания города;
- 9) user_changed_id – идентификатор пользователя, который изменил город;
- 10) date_changed – дата изменения записи о городе;
- 11) deleted – флаг, который показывает был ли город удален;
- 12) map_latitude – строковое значение широты для города. Поле используется для отображения города на карте;
- 13) map_longitude – строковое значение долготы для города. Поле используется для отображения города на карте;
- 14) map_zoom – значение увеличения для города на карте.

Таблица time_zones – список часовых поясов. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) time_zone_name – название часового пояса;
- 3) time_zone_code – код часового пояса;
- 4) time_offset – смещение по времени;

5) deleted – флаг, который показывает удален ли часовой пояс (влияет на отображение в системе).

На рисунке 2.4 представлена схема Booking базы данных сервиса TalonDo.

bookings	
123	id
123	client_id
123	talondo_service_id
123	foreign_service_id
123	price
123	schedule_item_id
ABC	status
ABC	comment
123	user_created_id
⌚	date_created
123	user_changed_id
⌚	date_changed
✓	deleted
✓	private_office_visibility

Рисунок 2.4 – Таблицы схемы Booking

Таблица booking – список бронирований, которые совершил пользователь с типом клиент. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) client_id – идентификатор клиента;
- 3) talondo_service_id – идентификатор услуги TalonDo;
- 4) foreign_service_id – идентификатор внешней услуги организации;
- 5) price – цена;
- 6) schedule_item_id – идентификатор времени приема;
- 7) status – статус бронирования;
- 8) comment – комментарий к бронированию;

- 9) user_created_id – идентификатор пользователя, который создал бронирование;
- 10) date_created – дата создания бронирования;
- 11) user_changed_id – идентификатор пользователя, который изменил бронирование;
- 12) date_changed – дата изменения записи о бронировании;
- 13) deleted – флаг, который показывает было ли бронирование удалено;
- 14) private_office_visibility – видимость бронирования в личном кабинете клиента.

На рисунке 2.5 представлена схема Cache базы данных сервиса TalonDo.

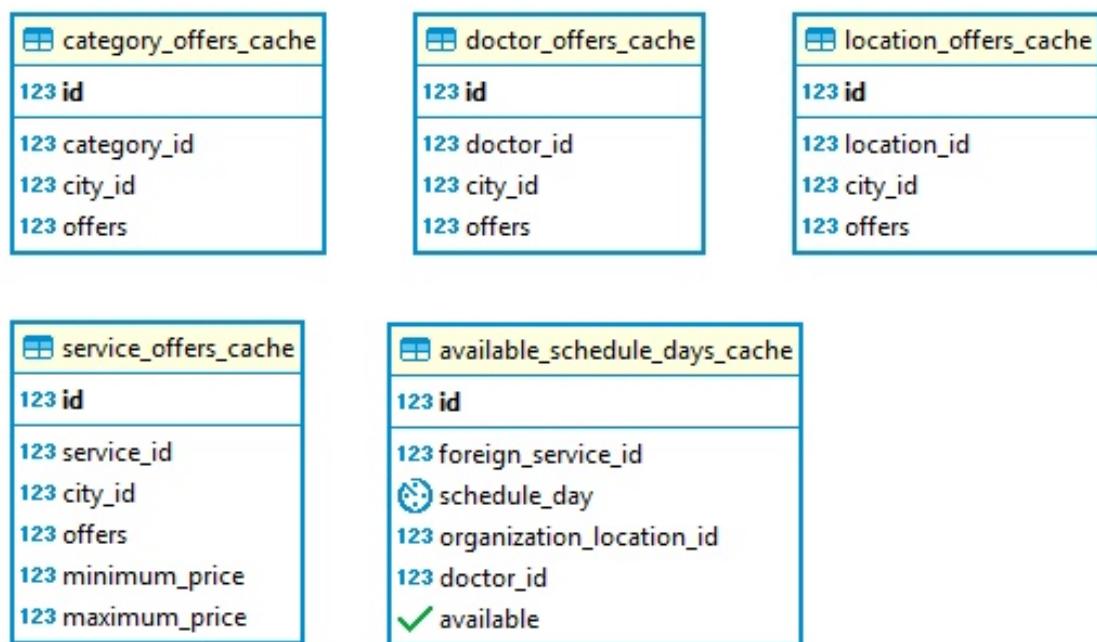


Рисунок 2.5 – Таблицы схемы Cache

Таблица available_schedule_days_cache – кэш доступности расписания по дням. Для ускорения генерации расписания на определенный промежуток времени в системе используется технология кэширования. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) foreign_service_id – идентификатор услуги организации;
- 3) schedule_day – дата расписания;
- 4) organization_location_id – идентификатор филиала организации;

- 5) doctor_id – идентификатор врача;
- 6) available – флаг, который показывает доступно ли расписание для конкретного промежутка времени.

Таблица service_offers_cache – кэш предложений по услугам сервиса TalonDo с учетом доступности расписания. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) service_id – идентификатор услуги;
- 3) city_id – идентификатор города;
- 4) minimum_price – минимальная цена услуги;
- 5) maximum_price – максимальная цена услуги;
- 6) offers – количество предложений на услугу.

Таблица category_offers_cache – кэш предложений по категориям сервиса TalonDo. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) category_id – идентификатор категории;
- 3) city_id – идентификатор города;
- 4) offers – количество предложений на категорию.

Таблица doctor_offers_cache – кэш предложений по врачам. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) doctor_id – идентификатор врача;
- 3) city_id – идентификатор города;
- 4) offers – количество предложений.

Таблица location_offers_cache – кэш предложений по филиалам клиник. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) location_id – идентификатор филиала клиники;
- 3) city_id – идентификатор города;

- 4) offers – количество предложений.

На рисунке 2.6 представлена схема Doctor базы данных сервиса TalonDo.

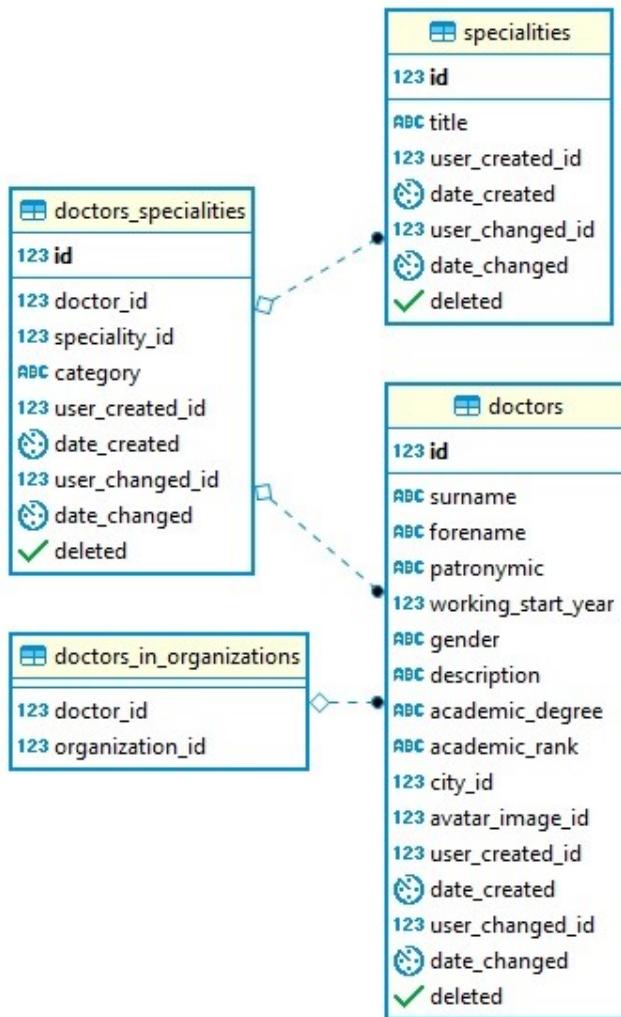


Рисунок 2.6 – Таблицы схемы Doctor

Таблица **doctors** – список врачей системы. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) surname – фамилия врача;
- 3) forename – имя врача;
- 4) patronymic – отчество врача;
- 5) gender – пол врача;
- 6) academic_degree – ученная степень;
- 7) academic_rank – ученное звание;
- 8) working_start_year – год начала работы по специальности;

- 9) description – описание;
- 10) city_id – идентификатор города;
- 11) avatar_image_id – идентификатор изображения для аватара;
- 12) user_created_id – идентификатор пользователя, который создал врача;
- 13) date_created – дата создания врача;
- 14) user_changed_id – идентификатор пользователя, который изменил врача;
- 15) date_changed – дата изменения записи о враче;
- 16) deleted – флаг, который показывает была ли запись о враче удалена.

Таблица doctors_specialities – список специальностей для врачей. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) specialty_id – идентификатор специальности;
- 3) doctor_id – идентификатор врача;
- 4) category – категория, которая может быть присвоена врачу по данной специальности;
- 5) user_created_id – идентификатор пользователя, который создал специальность для врача;
- 6) date_created – дата создания специальности для врача;
- 7) user_changed_id – идентификатор пользователя, который изменил специальность для врача;
- 8) date_changed – дата изменения записи о специальности для врача;
- 9) deleted – флаг, который показывает была ли специальность для врача удалена.

Таблица specialities – список специальностей. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) title – название специальности;

- 3) user_created_id – идентификатор пользователя, который создал специальность;
- 4) date_created – дата создания специальности;
- 5) user_changed_id – идентификатор пользователя, который изменил специальность;
- 6) date_changed – дата изменения записи о специальности;
- 7) deleted – флаг, который показывает была ли специальность удалена.

Таблица doctors_in_organizations – список врачей в организациях. Поля таблицы:

- 1) organization_id – идентификатор организации;
- 2) doctor_id – идентификатор врача.

На рисунке 2.7 представлена схема Organization базы данных сервиса TalandDo.

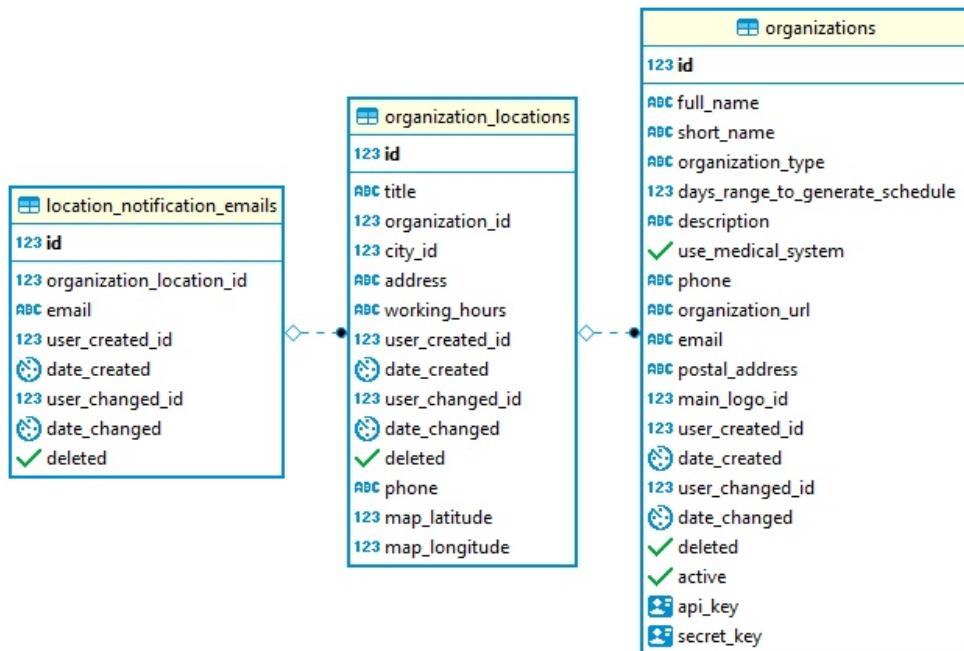


Рисунок 2.7 – Таблицы схемы Organization

Таблица organizations – список организаций системы. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) full_name – полное наименование;

- 3) short_name – сокращенное наименование;
- 4) organization_type – тип организации;
- 5) days_range_to_generate_schedule – диапазон дней для генерации расписания;
- 6) description – описание организации;
- 7) use_medical_system – флаг, который показывает использует ли организация стороннюю МИС;
- 8) phone – номер телефона;
- 9) organization_url – адрес официального сайта организации;
- 10) email – адрес электронной почты организации;
- 11) postal_address – почтовый адрес;
- 12) main_logo_id – идентификатор изображения – логотипа организации;
- 13) user_created_id – идентификатор пользователя, который создал организацию;
- 14) date_created – дата создания организации;
- 15) user_changed_id – идентификатор пользователя, который изменил организацию;
- 16) date_changed – дата изменения организации;
- 17) deleted – флаг, который показывает была ли организация удалена;
- 18) active – флаг, который показывает, активна ли организация.

Таблица organization_locations – список филиалов организации. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) title – наименование филиала;
- 3) organization_id – идентификатор организации;
- 4) city_id – идентификатор города, где находится филиал;
- 5) address – адрес;
- 6) phone – номер телефона филиала;

- 7) map_longitude – координаты широты для отображения филиала на карте;
- 8) map_latitude – координаты долготы для отображения филиала на карте;
- 9) working_hours – часы работы;
- 10) user_created_id – идентификатор пользователя, который создал филиал;
- 11) date_created – дата создания филиала;
- 12) user_changed_id – идентификатор пользователя, который изменил филиал;
- 13) date_changed – дата изменения филиала;
- 14) deleted – флаг, который показывает был ли филиал удален.

Таблица location_notification_email – список подтвержденных электронных адресов филиалов. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) organization_location_id – идентификатор филиала;
- 3) email – адрес электронной почты;
- 4) user_created_id – идентификатор пользователя, который создал электронный адрес;
- 5) date_created – дата создания электронного адреса;
- 6) user_changed_id – идентификатор пользователя, который изменил электронный адрес;
- 7) date_changed – дата изменения электронного адреса;
- 8) deleted – флаг, который показывает был ли электронный адрес удален.

На рисунке 2.8 представлена схема Schedule базы данных сервиса Talon-Do.

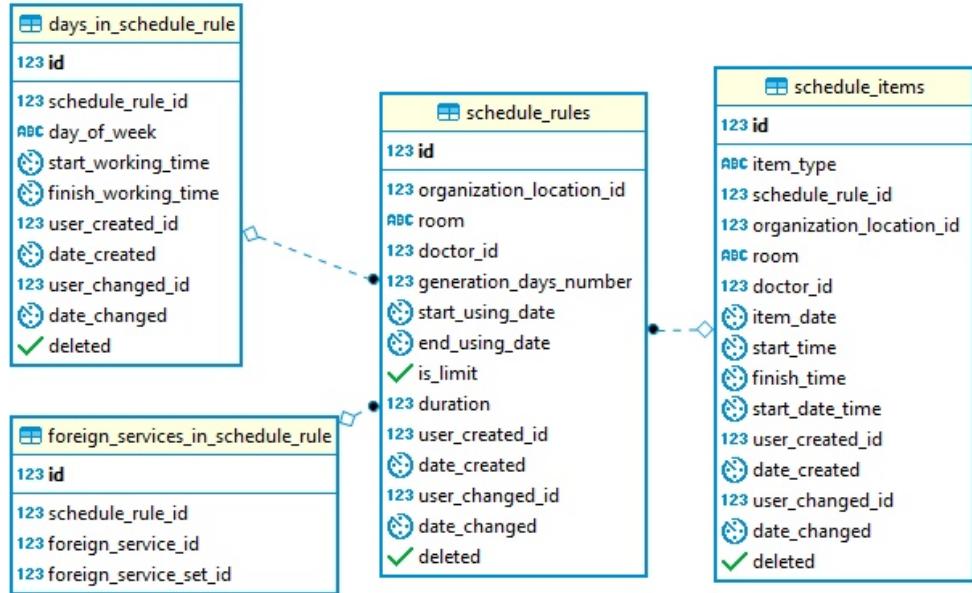


Рисунок 2.8– Таблицы схемы Schedule

Таблица `foreign_services_in_schedule_rule` – список внешних услуг в правилах генерации расписания. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) schedule_rule_id – идентификатор правила расписания;
- 3) foreign_service_id – идентификатор внешней услуги;
- 4) foreign_service_set_id – идентификатор набора услуг организации.

Таблица `days_in_schedule_rule` – список дней недели, по которым будет генерироваться времена расписания. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) schedule_rule_id – идентификатор правила расписания;
- 3) day_of_week – указание дня в неделе (понедельник, вторник и т.д.);
- 4) start_working_time – время начала работы;
- 5) finish_working_time – время окончания работы;
- 6) user_created_id – идентификатор пользователя, который создал день в правиле расписания;
- 7) date_created – дата создания дня;
- 8) user_changed_id – идентификатор пользователя, который изменил день в правиле расписания;

- 9) date_changed – дата изменения дня;
- 10) deleted – флаг, который показывает, был ли день в правиле расписания удален.

Таблица schedule_rules – список правил, по которым генерируется расписание. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) organization_location_id – идентификатор филиала организации;
- 3) doctor_id – идентификатор врача;
- 4) room – номер кабинета, в котором принимает врач;
- 5) generation_days_number – количество дней, на которое необходимо сгенерировать расписание по данному правилу;
- 6) start_using_date – дата начала действия правила;
- 7) finish_using_date – дата окончания действия правила;
- 8) is_limit – флаг, который показывает лимитность услуги;
- 9) duration – продолжительность приема;
- 10) user_created_id – идентификатор пользователя, который создал правило расписания;
- 11) date_created – дата создания правила;
- 12) user_changed_id – идентификатор пользователя, который изменил правило расписания;
- 13) date_changed – дата изменения правила;
- 14) deleted – флаг, который показывает, было ли правило удалено.

Таблица schedule_items – список занятых (с записью или исключенных) времен в расписании. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) item_type – тип занятого времени в расписании (исключено, заблокировано, забронировано);
- 3) schedule_rule_id – идентификатор правила расписания;
- 4) organization_location_id – идентификатор филиала организации;

- 5) doctor_id – идентификатор врача, ведущего прием или предоставляющего услугу;
- 6) start_time – время начала приема (или оказания услуги);
- 7) finish_time – время окончания приема (или оказания услуги);
- 8) room – кабинет;
- 9) item_date – день, в котором занято время расписания;
- 10) start_date_time – момент начала занятого времени (дата + время + часовая зона);
- 11) user_created_id – идентификатор пользователя, который создал занятое время;
- 12) date_created – дата создания занятого времени;
- 13) user_changed_id – идентификатор пользователя, который изменил занятое время;
- 14) date_changed – дата изменения занятого времени;
- 15) deleted – флаг, который показывает, было ли занятое время удалено.

На рисунке 2.9 представлена схема Service базы данных сервиса TalonDo.

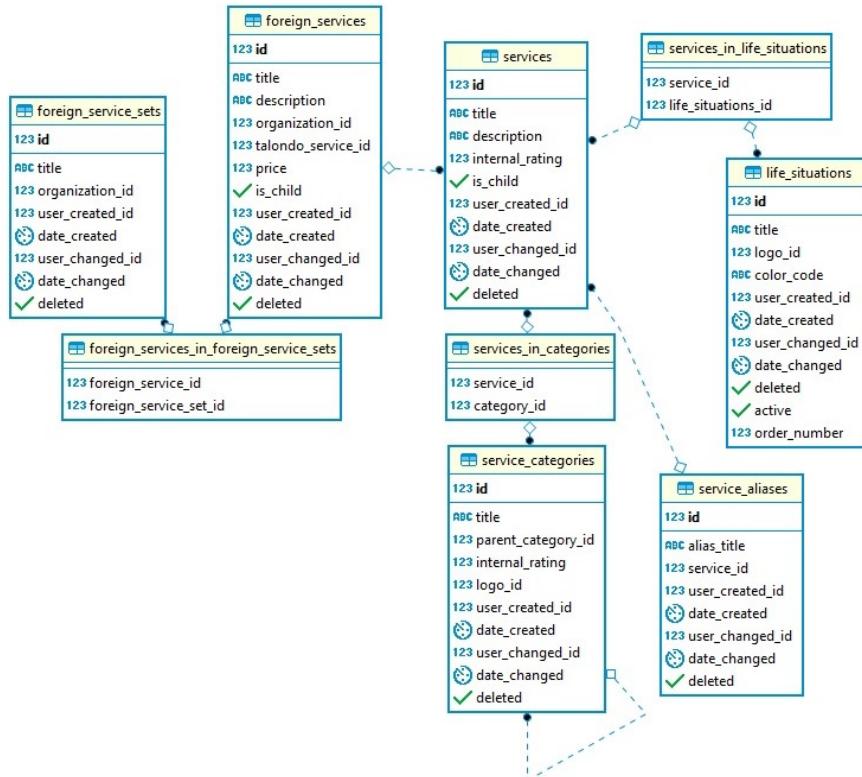


Рисунок 2.9 – Таблицы схемы Service

Таблица services – услуги системы. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) title – название услуги;
- 3) description – описание услуги;
- 4) internal_rating – внутренний рейтинг. Используется для сортировки услуг по рейтингу;
- 5) is_child – флаг, который показывает, является ли услуга детской;
- 6) user_created_id – идентификатор пользователя, который создал услугу;
- 7) date_created – дата создания услуги;
- 8) user_changed_id – идентификатор пользователя, который изменил услугу;
- 9) date_changed – дата изменения услуги;
- 10) deleted – флаг, который показывает, была ли услуга удалена.

Таблица life_situations – жизненные ситуации. Используются для условной группировки услуг. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) title – название жизненной ситуации;
- 3) logo_id – идентификатор логотипа (изображения) жизненной ситуации;
- 4) color_code – код цвета, которым будут выделены жизненные ситуации;
- 5) user_created_id – идентификатор пользователя, который создал жизненную ситуацию;
- 6) active – флаг, который показывает активна ли данная жизненная ситуация;
- 7) order_number – порядковый номер. Поле используется для отображения жизненных ситуаций на главной странице;
- 8) date_created – дата создания жизненной ситуации;

9) user_changed_id – идентификатор пользователя, который изменил жизненную ситуацию;

10) date_changed – дата изменения жизненной ситуации;

11) deleted – флаг, который показывает, была ли жизненная ситуация удалена.

Таблица service_in_life_situations – список услуг для жизненных ситуаций. Таблица служит для «раскрытия» связи между таблицами типа «много – ко – многим». Поля таблицы:

1) service_id – идентификатор услуги;

2) life_situation_id – идентификатор жизненной ситуации.

Таблица service_categories – категории для услуг. Поля таблицы:

1) id – уникальный идентификатор;

2) title – название категории;

3) parent_category_id – идентификатор родительской категории;

4) internal_rating – внутренний рейтинг;

5) logo_id – идентификатор логотипа категории услуги;

6) user_created_id – идентификатор пользователя, который создал категорию;

7) date_created – дата создания категории;

8) user_changed_id – идентификатор пользователя, который изменил категорию;

9) date_changed – дата изменения категории;

10) deleted – флаг, который показывает, была ли категория удалена.

Таблица service_in_categorie – список услуг в категориях. Таблица служит для «раскрытия» связи между таблицами типа «много – ко – многим». Поля таблицы:

1) service_id – идентификатор услуги;

2) category_id – идентификатор категории.

Таблица service_aliases – список алиасов (альтернативных названий) услуг.

Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) alias_title – наименование алиаса;
- 3) service_id – идентификатор связной услуги;
- 4) user_created_id – идентификатор пользователя, который создал алиас;
- 5) date_created – дата создания алиаса;
- 6) user_changed_id – идентификатор пользователя, который изменил алиас;
- 7) date_changed – дата изменения алиаса;
- 8) deleted – флаг, который показывает, был ли алиас удален.

Таблица foreign_services – список услуг организаций. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) title – название услуги организации;
- 3) organization_id – идентификатор организации;
- 4) description – описание услуги;
- 5) price – цена услуги;
- 6) is_child – флаг, который показывает является ли услуга детской;
- 7) talondo_service_id – идентификатор связной услуги сервиса TalonDo;
- 8) user_created_id – идентификатор пользователя, который создал внешнюю услугу;
- 9) date_created – дата создания внешней услуги;
- 10) user_changed_id – идентификатор пользователя, который изменил внешнюю услугу;
- 11) date_changed – дата изменения внешней услуги;
- 12) deleted – флаг, который показывает, была ли внешняя услуга удалена.

Таблица `foreign_service_sets` – наборы услуг организаций. Поля таблицы:

- 1) `id` – уникальный идентификатор;
- 2) `title` – название набора услуг организации;
- 3) `organization_id` – идентификатор организации;
- 4) `user_created_id` – идентификатор пользователя, который создал набор услуг;
- 5) `date_created` – дата создания набора услуг;
- 6) `user_changed_id` – идентификатор пользователя, который изменил набор услуг;
- 7) `date_changed` – дата изменения набора услуг;
- 8) `deleted` – флаг, который показывает, был ли набор услуг удален.

Таблица `foreign_services_in_foreign_service_set` – список услуг организации в наборе услуг. Поля таблицы:

- 1) `foreign_service_id` – идентификатор услуги;
- 2) `foreign_service_set_id` – идентификатор набора услуг.

На рисунке 2.10 представлена схема Comments базы данных сервиса TlonDo.

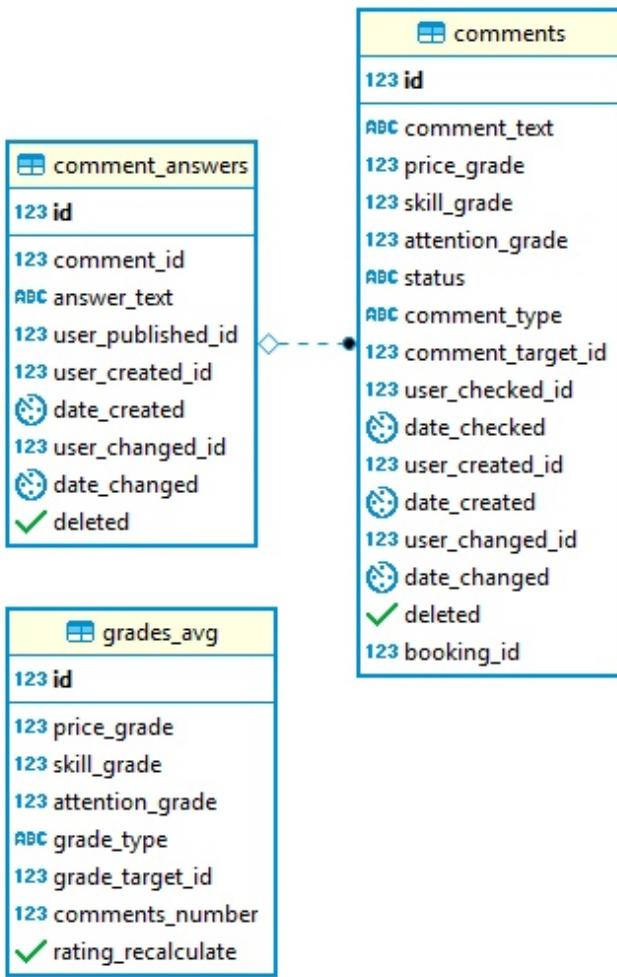


Рисунок 2.10 – Таблицы схемы Comments

Таблица comments – отзывы для врачей и клиник. Поля таблицы:

- 1) **id** – уникальный идентификатор;
- 2) **comment_text** – текст отзыва;
- 3) **price_grade** – оценка цены;
- 4) **skill_grade** – оценка навыков врача;
- 5) **attention_grade** – оценка уровня обращения с пациентом;
- 6) **status** – статус отзыва. Поле используется для хранения информации о проверенности отзыва;
- 7) **comment_type** – тип отзыва (отзыв о враче/о клинике);
- 8) **comment_target_id** – идентификатор врача или клиники, к которым относится отзыв;
- 9) **booking_id** – идентификатор бронирования;

- 10) user_checked_id – идентификатор пользователя, который верифицировал отзыв;
- 11) date_checked – дата верификации отзыва;
- 12) user_created_id – идентификатор пользователя, который создал отзыв;
- 13) date_created – дата создания отзыва;
- 14) user_changed_id – идентификатор пользователя, который изменил отзыв;
- 15) date_changed – дата изменения отзыва;
- 16) deleted – флаг, который показывает, был ли отзыва удален.

Таблица comments_answers – ответы на оставленные отзывы. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) comment_id – идентификатор отзыва;
- 3) answer_text – текст ответа;
- 4) user_published_id – идентификатор пользователя, который опубликовал ответ на отзыв;
- 5) user_created_id – идентификатор пользователя, который создал ответ;
- 6) date_created – дата создания ответа;
- 7) user_changed_id – идентификатор пользователя, который изменил ответ;
- 8) date_changed – дата изменения ответа;
- 9) deleted – флаг, который показывает был ли ответ на отзыв удален.

Таблица grades_avg – средние значения оценок. При просмотре информации о клинике или враче отображается информация из данной таблицы. Поля таблицы:

- 1) id – уникальный идентификатор;
- 2) price_grade – среднее значение оценки цены;

- 3) skill_grade – среднее значение оценки навыков;
- 4) attention_grade – среднее значение оценки отношения к клиенту;
- 5) grade_type – тип оценки;
- 17) grade_target_id – идентификатор врача или клиники, к которым относится оценка;
- 6) comments_number – количество отзывов;
- 7) rating_recalculated – флаг, который показывает нужен ли пересчет среднего значения оценок. Когда у клиники или врача добавляется отзыв, который содержит оценки, данный флаг помечается как true. В системе с определенной частотой запускается процедура, которая выполняет пересчет оценок при необходимости.

В описанной БД присутствуют 8 схем и 41 таблица. Все необходимые для работы системы данные хранятся в БД. В базе данных содержится как актуальная информация о клиниках, пользователях сервиса и бронированиях, так и служебная информация для корректной работы системы. Некоторые таблицы являются расшировочными, не содержат конкретной информации и служат для создания корректных ссылок на реальные объекты в системе [39].

2.5 Среда разработки и используемые технологии

Для разработки web-приложения TalonDo была использована IDE NetBeans.

NetBeans IDE — свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, Python, PHP, JavaScript, C, C++, Ада и ряда других.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведётся независимым сообществом разработчиков-энтузиастов и компанией NetBeans Org.

Последние версии NetBeans IDE поддерживают рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение наби-

раемых конструкций в режиме реального времени и множество предопределённых шаблонов кода.

В сентябре 2016 года Oracle передала интегрированную среду разработки NetBeans в руки фонда Apache.

NetBeans – это мощный инструмент с открытым исходным кодом, позволяющий работать не только на ОС Windows, но и на Linux, MacOS и даже Oracle Solaris. NetBeans считается Java–ориентированной средой разработки, хотя позволяет использовать и сочетать несколько языков программирования. Данная интегрированная среда разработки успешно взаимодействует со Spring Web MVC framework, с Hibernate библиотекой, а также с JPA, JSP, Struts и т.п. [29].

Для хранения информации, которая необходима для работы сервиса используется СУБД – PostgreSQL 10. PostgreSQL — свободная объектно–реляционная система управления базами данных (СУБД). Существует в реализациях для множества UNIX–подобных платформ, включая AIX, различные BSD–системы, HP–UX, IRIX, Linux, MacOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

PostgreSQL создана на основе некоммерческой СУБД Postgres, разработанной как open–source проект в Калифорнийском университете в Беркли. К разработке Postgres, начавшейся в 1986 году, имел непосредственное отношение Майкл Стоунбрейкер, руководитель более раннего проекта Ingres. Название расшифровывалось как «Post Ingres», и при создании Postgres были применены многие уже ранее сделанные наработки.

Стоунбрейкер и его студенты разрабатывали новую СУБД в течение восьми лет с 1986 по 1994 год. За этот период в синтаксис были введены процедуры, правила, пользовательские типы и другие компоненты. В 1995 году разработка снова разделилась: Стоунбрейкер использовал полученный опыт в создании коммерческой СУБД Illustra, продвигаемой его собственной одноимённой компанией, а его студенты разработали новую версию Postgres —

Postgres95, в которой язык запросов POSTQUEL — наследие Ingres — был заменен на SQL.

Разработка Postgres95 была выведена за пределы университета и передана команде энтузиастов. Новая СУБД получила имя, под которым она известна и развивается в текущий момент — PostgreSQL [30].

Сервер приложений — Apache Tomcat 9.0. Apache Tomcat — это контейнер, который позволяет вам использовать интернет приложения такие, как Java сервлеты и JSP (серверные страницы Java). Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java. Разработка и поддержка Tomcat осуществляется фондом Apache Software Foundation и добровольцами. Пользователи имеют свободный доступ к исходным кодам и бинарным файлам Tomcat согласно лицензии Apache License 2.0.

HTML/CSS framework — BootStrap. Bootstrap (также известен как Twitter Bootstrap) — свободный набор инструментов для создания сайтов и web-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов web-интерфейса, включая JavaScript-расширения. Bootstrap использует современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров [31].

Основные инструменты Bootstrap:

- 1) Сетки — заранее заданные размеры колонок, которые можно сразу же использовать, например ширина колонки 140 px относится к классу .span2 (.col-md-2 в третьей версии фреймворка), который можно использовать в CSS-описании документа.
- 2) Шаблоны — фиксированный или резиновый шаблон документа.
- 3) Типографика — описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. п.
- 4) Медиа — представляет некоторое управление изображениями и видео.

5) Таблицы – средства оформления таблиц, вплоть до добавления функциональности сортировки.

6) Формы – классы для оформления форм и некоторых событий, происходящих с ними.

7) Навигация – классы оформления для табов, вкладок, страничности, меню и панели инструментов.

8) Алерты – оформление диалоговых окон, подсказок и всплывающих окон [32].

Application framework – Spring. Spring Framework (или коротко Spring) – универсальный фреймворк с открытым исходным кодом для Java–платформы. Также существует форк для платформы .NET Framework, названный Spring.NET.

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development».

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первый стабильный релиз 1.0 был выпущен в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 5.0.1.

Несмотря на то, что Spring не обеспечивал какую–либо конкретную модель программирования, он стал широко распространённым в Java–сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет большую свободу Java–разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java–приложении, и существует множество расширений и усовершенствований для построения web–приложений на Java Enterprise платформе. По этим причинам

Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк [33].

Web-framework – Thymeleaf. Thymeleaf – это механизм шаблонов Java XML / XHTML / HTML5, который может работать в web – средах (на основе Servlet). Он лучше подходит для обслуживания XHTML / HTML5 на уровне представления web–приложений на базе MVC, но он может обрабатывать любой XML–файл даже в автономных средах. Он обеспечивает полную интеграцию Spring Framework.

В web–приложениях Thymeleaf стремится стать полной заменой JSP и реализует концепцию Natural Templates: файлы шаблонов, которые могут быть непосредственно открыты в браузерах и которые по–прежнему отображаются правильно как web –страницы. Thymeleaf – это программное обеспечение с открытым исходным кодом, лицензированное в соответствии с лицензией Apache 2.0.

3 Реализация модифицированных методов расчета параметров оптимизации

При поиске объектов в TalonDo запросы пользователя использовались для поиска полных совпадений в базе данных. Согласно статистике, в подавляющем большинстве случаев метод поиска по строковому совпадению не предоставляет пользователю полного перечня объектов, которые могут использоваться в качестве решения. Следует отметить, что использование методов решения задач МКО в TalonDo поможет увеличить вероятность нахождения наиболее оптимального варианта записи к врачу. Более того, наличие данного функционала в системе является отличительной характеристикой сервиса относительно его аналогов.

Исходя из вышесказанного, было принято решение реализовать 2 метода решения задачи МКО в сервисе TalonDo. Для реализации выбраны следующие методы:

- 1) метод аддитивной свертки критериев (для отбора наибольшего количества возможных вариантов предложений);
- 2) метод главного критерия (для отбора наиболее оптимального решения).

С главной страницы сервиса TalonDo пользователь может перейти к поиску предложений через список категорий услуг, список врачей или список клиник. После выбора врача, клиники или категории услуг пользователь переходит к списку услуг. Для поиска возможных предложений необходимо выбрать услугу из списка. Для каждого предложения указываются цена услуги, клиника и врач (рисунок 3.1).

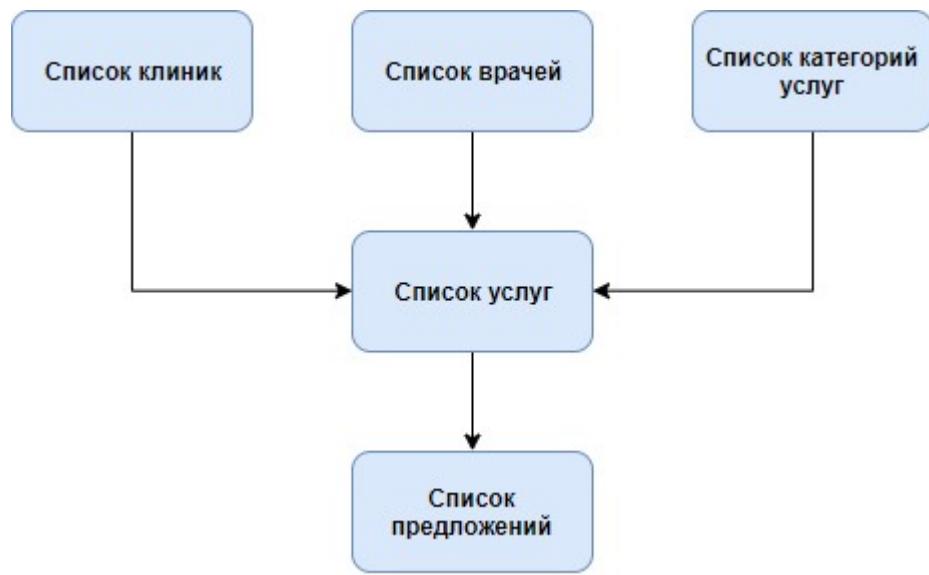


Рисунок 3.1 – Упрощенная схема пользовательского сценария по выбору предложения

В TalonDo используются следующие параметры для формирования поискового запроса пользователем: услуги, врачи, клиники и предложения. Набор критериев поиска определяется набором данных, которые хранятся в базе.

Критерии поиска клиники:

- a) название;
- b) врач (ФИО);
- c) наличие сайта.

Критерии поиска врача:

- a) ФИО;
- b) клиника;
- c) пол врача;
- d) ученое звание;
- e) ученая степень;
- f) специальность;
- g) стаж (в годах).

В сервисе TalonDo услуги разделены на категории. После выбора категории отображается список услуг. Критерии поиска услуги:

- a) название;
- b) цена;

- c) клиника (название);
- d) жизненная ситуация;
- e) категория услуг;
- f) пол врача;
- g) врач (ФИО);
- h) применима ли данная услуга к детям.

Поиск предложений осуществляется по следующим критериям:

- a) клиника;
- b) пол врача;
- c) врач (ФИО);
- d) цена;
- e) применима ли данная услуга к детям.

Бронирование – является результатом использования сервиса. Бронирование характеризуется информацией об услуге, клинике, враче, а так же датой и временем оказания услуги. Создание бронирования происходит на странице списка предложений. Исходя из этого, для реализации выбранных методов решено использовать страницу поиска предложений.

3.1 Метод аддитивной свертки

3.1.1 Описание алгоритма работы метода

Данный тип свертки может использоваться, если лицо, принимающее решение, считает допустимым, что абсолютное уменьшение оценки по одному критерию может быть компенсировано суммарным абсолютным увеличением оценки по другим критериям.

Возьмем $W(x)$ – функция значимости в зависимости от количества критериев поиска и их коэффициентов (весов). Вектор $\Lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ – вектор степеней относительной важности (приоритетов, весов) критериев.

Тогда значение функции $W(x)$ вычисляем как:

$$W_{add}(x) = \sum_{j=1}^n \lambda_j f_j(x) \rightarrow \max, x \in X \quad (3.1)$$

Переменная f_j равна 1, если введенный пользователем запрос для j -ого критерия совпал со значением из базы данных, и 0, если совпадение не найдено. Таким образом, если $f_j = 0$, то все слагаемое не учитывается в подсчете суммы. Если $f_j = 1$, то вес для данного критерия, учитывается и сумма всего «варианта» увеличивается.

Самый оптимальный для пользователя вариант – это вариант, когда $W_{add}(x) \rightarrow \max, x \in X$.

3.1.2 Реализация метода

На рисунке 3.2 представлен метод контроллера `findOffers()`, которому на вход подаются значения фильтра и в теле данного метода вызывается подпрограмма `getOffersForTalondoService()`, находящаяся внутри сервиса `OffersService`. Метод `getOffersForTalondoService()` (рисунок 3.3) возвращает данные, которые необходимы для отрисовки страницы предложений (номер страницы, количество предложений на странице, список предложений и т.п.). Метод `findOffers()` возвращает строку–путь до HTML–шаблона, внутри приложения. По данному шаблону происходит отрисовка страницы.

```

@RequestMapping(value = "/offers/find", method = RequestMethod.GET)
public String findOffers(Model model, HttpServletRequest request, HttpServletResponse response,
    @RequestParam(name = "serviceId") Integer talondoServiceId,
    @RequestParam(name = "minPrice", required = false) Integer minPrice,
    @RequestParam(name = "maxPrice", required = false) Integer maxPrice,
    @RequestParam(name = "sorter", required = false) ServicesOffersSorter sorter,
    @RequestParam(name = "gender", required = false) Gender gender,
    @RequestParam(name = "doctorId", required = false) Integer doctorId,
    @RequestParam(name = "locationId", required = false) Integer locationId,
    @RequestParam(name = "child", required = false) boolean isChild,
    @RequestParam(name = "page", defaultValue = "0") Integer page,
    //calc fields
    @RequestParam(name = "filters", required = false) String filters,
    @RequestParam(name = "withCalc", required = false) String withCalc,
    @RequestParam(name = "clinicCalc", required = false, defaultValue = "0") Integer clinicCalc,
    @RequestParam(name = "doctorGenderCalc", required = false, defaultValue = "0") Integer doctorGenderCalc,
    @RequestParam(name = "doctorCalc", required = false, defaultValue = "0") Integer doctorCalc,
    @RequestParam(name = "priceCalc", required = false, defaultValue = "0") Integer priceCalc,
    @RequestParam(name = "childCalc", required = false, defaultValue = "0") Integer childCalc) {
    sorter = ServicesOffersSorter.getSorterOrDefault(sorter);
    List<Integer> filtersList = new Gson().fromJson(filters, new TypeToken<List<Integer>>() {
    }.getType());
    gender = Gender.getThisOrDefaultIfNull(gender);
    City city = basicBeanHelper.getDefaultCityFromCookie(request, response);
    OfferFilter filter = new OfferFilter(
        serviceService.getSimpleServiceBeanById(talondoServiceId),
        city.getId(),
        minPrice, maxPrice,
        doctorService.getSimpleDoctorBean(doctorId),
        gender,
        locationService.getSimpleClinicWithAddressBeanById(locationId),
        isChild);

    filter.setCalcValues(filtersList, clinicCalc, doctorGenderCalc, doctorCalc, priceCalc, childCalc);
    Page<OfferBean> offerBeans = offerService.getOffersForTalondoService(filter, new PageRequest(page, OFFERS_TO_SHOW), sorter);
    model.addAttribute("offerBeans", offerBeans);

    model.addAttribute("page", page);
    model.addAttribute("pages", offerBeans.getTotalPages());
    model.addAttribute("currentSorter", sorter);
    model.addAttribute("filter", filter);
    return "offers/listResultsPanel :: resultsPanel";
}

```

Рисунок 3.2 – Исходный код метода findOffers()

Если пользователь, при заполнении фильтра в интерфейсе, указал очередьность использования методов решения ЗМКО, то сначала осуществляется поиск списка предложений методом getServiceOffersCachesWithoutDoctor() (рисунок 3.5). Затем над полученными данными производится фильтрация методами решения ЗМКО, в порядке, указанном пользователем.

```

public PageImpl<OfferBean> getOffersForTalondoService(OfferFilter offersFilter, Pageable pageable, ServicesOffersSorter sorter) {
    if (offersFilter.isWithCalc()) {
        List<GroupedCache2> cachesWithoutDoctor = cacheService.getServiceOffersCachesWithoutDoctor(offersFilter, sorter);
        List<OfferBean> offers = cachesWithoutDoctor
            .stream()
            .map(group -> {
                ForeignService foreignService = foreignServiceService.getForeignServiceById(group.getForeignServiceId());
                OrganizationLocation organizationLocation = locationService.getOrganizationLocationById(group.getLocationId());
                return new OfferBean(
                    foreignService,
                    organizationLocation,
                    cacheService.getServiceOfferBeanDoctors(group.getForeignServiceId(), group.getLocationId(), offersFilter),
                    averageGradeService.findAverageGradeByTargetIdAndType(CommentType.CLINIC, group.getLocationId())
                );
            })
            .collect(Collectors.toList());
        for(Integer filter : offersFilter.getFilters()){
            switch(filter){
                case 1:
                    offers = calcMethod1(offers, offersFilter);
                    break;
                case 2:
                    offers = calcMethod2(offers, offersFilter);
                    break;
            }
        }
        return new PageImpl(offers, pageable, offers.size());
    } else {
        List<OfferBean> offers = calcSimple(offersFilter, pageable, sorter);
        return new PageImpl(offers);
    }
}

```

Рисунок 3.3 – Исходный код метода getOffersForTalondoService()

Если пользователь не указал очередность, то вызывается метод `calcSimple()`, в котором производится поиск предложений по полному совпадению значений параметров поискового запроса со значениями из базы данных (рисунок 3.4).

```
private List<OfferBean> calcSimple(OfferFilter offersFilter, Pageable pageable, ServicesOffersSorter sorter) {
    Page<GroupedCache2> cachesWithoutDoctor = cacheService.getServiceOffersCachesWithoutDoctor(offersFilter, pageable, sorter);
    List<OfferBean> offers = cachesWithoutDoctor
        .getContent()
        .stream()
        .map(group -> {
            ForeignService foreignService = foreignServiceService.getForeignServiceById(group.getForeignServiceId());
            OrganizationLocation organizationLocation = locationService.getOrganizationLocationById(group.getLocationId());
            return new OfferBean(
                foreignService,
                organizationLocation,
                cacheService.getServiceOfferBeanDoctors(group.getForeignServiceId(), group.getLocationId(), offersFilter),
                averageGradeService.findAverageGradeByTargetIdAndType(CommentType.CLINIC, group.getLocationId())
            );
        })
        .collect(Collectors.toList());
    return offers;
}
```

Рисунок 3.4 – Исходный код метода `calcSimple()`

В методе `getServiceOffersCachesWithoutDoctor()` формируется SQL–запрос для обращения к БД.

```
public List<GroupedCache2> getServiceOffersCachesWithoutDoctor(OfferFilter filter, ServicesOffersSorter sorter) {
    QForeignService foreignService = QForeignService.foreignService;
    QService service = QService.service;
    QOrganizationLocation organizationLocation = QOrganizationLocation.organizationLocation;
    List<GroupedCache2> caches;
    caches = new JPAQuery<>(entityManager)
        .select(Q_DAYS_CACHE.foreignService().id,
            Q_DAYS_CACHE.location().id,
            Q_DAYS_CACHE.foreignService().price,
            Q_DAYS_CACHE.foreignService().title)
        .distinct()
        .from(Q_DAYS_CACHE)
        .join(foreignService).on(foreignService.id.eq(Q_DAYS_CACHE.foreignService().id))
        .join(service).on(service.id.eq(foreignService.talondoService().id),
            service.id.eq(filter.getTalondoServiceBean().getId()))
        .join(organizationLocation).on(organizationLocation.id.eq(Q_DAYS_CACHE.location().id),
            organizationLocation.city().id.eq(filter.getCityId()))
        .where(
            Q_DAYS_CACHE.available.isTrue()
        )
        .orderBy(sorter.getOrderSpecifier())
        .fetch()
        .stream()
        .map(tuple -> new GroupedCache2(
            tuple.get(Q_DAYS_CACHE.foreignService().id),
            tuple.get(Q_DAYS_CACHE.location().id)
        ))
        .collect(Collectors.toList());
    return caches;
}
```

Рисунок 3.5 – Исходный код метода `getServiceOffersCachesWithoutDoctor()`

На рисунках 3.6–3.7 представлен исходный код метода аддитивной свертки. После считывания введенных пользователем весов для параметров фильтра, происходит вычисление процентной значимости критерия. Это сделано для того, чтобы пользователь мог вводить числа в любых диапазонах, например, от 0

до 1 или от 1 до 1000. На рисунке 3.6 представлена обработка коэффициентов для следующих параметров фильтра:

- 1) название клиники;
- 2) применима ли данная услуга к детям;
- 3) диапазон цены.

```
private List<OfferBean> calcMethod1(List<OfferBean> offers, OfferFilter offersFilter) {  
    if(offers.isEmpty()) { return offers; }  
    Double lowerVal = offersFilter.getGradeCalc();  
    Integer clinicCalc = offersFilter.getClinicCalc();  
    Integer doctorGenderCalc = offersFilter.getDoctorGenderCalc();  
    Integer doctorCalc = offersFilter.getDoctorCalc();  
    Integer priceCalc = offersFilter.getPriceCalc();  
    Integer childCalc = offersFilter.getChildCalc();  
    Integer sum = clinicCalc + doctorGenderCalc + doctorCalc + priceCalc + childCalc;  
    for (OfferBean bean : offers) {  
        Double calc = 0d;  
        if (clinicCalc != 0) {  
            if (Objects.equals(bean.getOrganizationLocationId(), offersFilter.getClinicId())) {  
                calc += 1d / sum * clinicCalc;  
            }  
        }  
        if (childCalc != 0) {  
            if (Objects.equals(bean.isChild(), offersFilter.isChild())) {  
                calc += 1d / sum * childCalc;  
            }  
        }  
        if (priceCalc != 0) {  
            if (offersFilter.getPrice().isFull()) {  
                if (offersFilter.getPrice().getLowerInRubles()  
                    <= bean.getPrice() && bean.getPrice()  
                    <= offersFilter.getPrice().getUpperInRubles()) {  
                    calc += 1d / sum * priceCalc;  
                }  
            } else if (offersFilter.getPrice().hasMax()) {  
                if (bean.getPrice() <= offersFilter.getPrice().getUpperInRubles()) {  
                    calc += 1d / sum * priceCalc;  
                }  
            } else if (offersFilter.getPrice().hasMin()) {  
                if (offersFilter.getPrice().getLowerInRubles() <= bean.getPrice()) {  
                    calc += 1d / sum * priceCalc;  
                }  
            }  
        }  
    }  
}
```

Рисунок 3.6 – Исходный код метода аддитивной свертки (часть 1)

На рисунке 3.7 представлена обработка коэффициентов для параметров, связанных с врачом:

- 1) ФИО врача;
- 2) пол врача.

При совпадении значений фильтра со значениями из БД, происходит увеличение суммарного веса предложения.

```

        Double localDoctorGenderCalc = 0d;
        Double localDoctorCalc = 0d;
        List<OfferBeanDoctor> doctors = bean.getDoctors();
        if (doctors != null) {
            if (doctorGenderCalc != 0) {
                if (doctors.stream().filter(d -> d.getDoctorGender() == offersFilter.getDoctorsGender()).count() > 0) {
                    localDoctorGenderCalc = ld / sum * doctorGenderCalc;
                    calc += localDoctorGenderCalc;
                }
            }
            if (doctorCalc != 0) {
                if (doctors.stream().filter(d -> Objects.equals(d.getDoctorId(), offersFilter.getDoctorId())).count() > 0) {
                    localDoctorCalc = ld / sum * doctorCalc;
                    calc += localDoctorCalc;
                }
            }
            if (calc > lowerVal) {
                final Double sumCalc = calc;
                final Double ldgcalc = localDoctorGenderCalc;
                final Double ldCalc = localDoctorCalc;
                List<OfferBeanDoctor> newDoctors = doctors.stream().filter(d -> {
                    return (sumCalc
                            - (d.getDoctorGender() == offersFilter.getDoctorsGender() ? 0 : ldgcalc)
                            - (Objects.equals(d.getDoctorId(), offersFilter.getDoctorId()) ? 0 : ldCalc)) > lowerVal;
                }).collect(Collectors.toList());
                bean.setDoctors(newDoctors);
            }
        }
        bean.setCalc(calc);
    }
    offers = offers.stream().filter(b -> b.getCalc() > lowerVal).collect(Collectors.toList());
    offers.sort((o1, o2) -> o2.getCalc().compareTo(o1.getCalc()));
    return offers;
}

```

Рисунок 3.7 – Исходный код метода аддитивной свертки (часть 2)

Завершающей стадией метода аддитивной свертки является сравнение суммарной величины веса предложения с пороговым значением, равным 0,7. Метод возвращает список предложений, суммарный вес которых строго больше порогового значения.

3.1.3 Пользовательский сценарий использования метода

Для того чтобы продемонстрировать преимущества работы метода наглядно протестируем его работу на конкретном примере.

Чтобы зайти на страницу поиска предложений по услуге нужно на главной странице сервиса TalonDo (talondo.ru) перейти в раздел «Услуги». Затем выбрать категорию услуг. Например, «Стоматология» (рисунок 3.8–3.9).

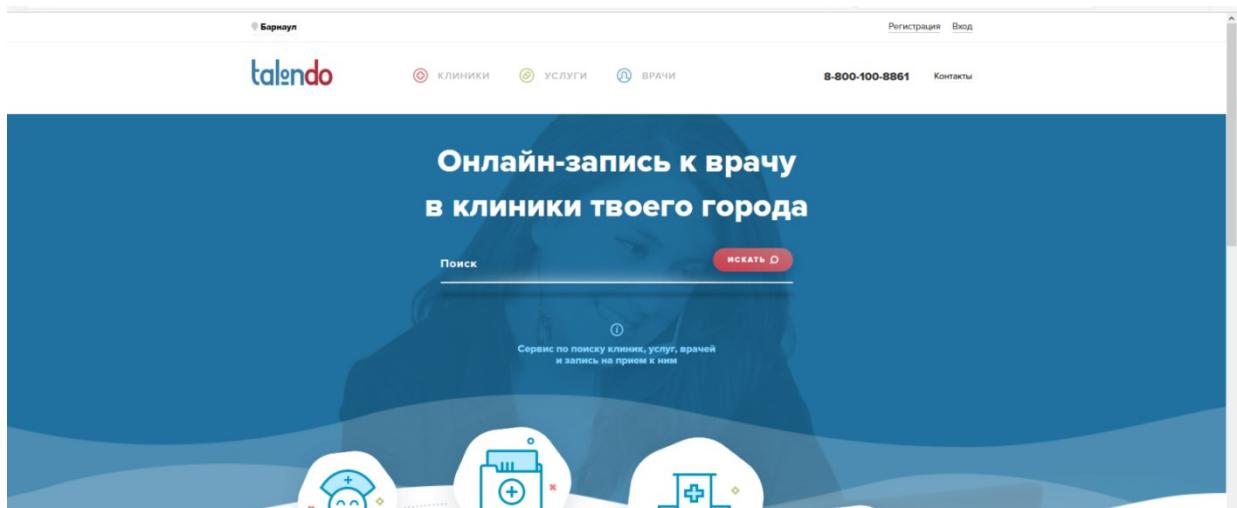


Рисунок 3.8 – Главная страница сервиса TalonDo

Рисунок 3.9 – Страница со списком услуг в категории «Стоматология»

После этого необходимо выбрать нужную подкатегорию, например «Детская стоматология». Откроется список услуг данной подкатегории. Пользователь сервиса выбираем услугу из списка. Выбираем услугу «Рассечение уздечки языка у детей» и нажимаем на кнопку «Записаться» (рисунок 3.10–3.11).

Барнаул Регистрация Вход

talendo КЛИНИКИ УСЛУГИ ВРАЧИ 8-800-100-8861 Контакты

Категории услуг → Стоматология → Детская стоматология

Рассечение уздечки языка у детей
Всего 24 предложения

254 - 1 557 ₽ [ЗАПИСАТЬСЯ](#)

Фторирование
Всего 20 предложений

451 - 1 536 ₽ [ЗАПИСАТЬСЯ](#)

Кариес эмали
Всего 16 предложений

916 - 1 963 ₽ [ЗАПИСАТЬСЯ](#)

Рисунок 3.10 – Список услуг подкатегории «Детская стоматология»

Услуги → Рассечение уздечки языка у детей

Рассечение уздечки языка у детей
Гигиенический кабинет
площадь Бакунина, 101
Виктор Александров Павлович | Екатерина Мария Игоревна |
Юрий Иван Петрович | Иванов Петр Сергеевич |
Иванов Сергей Петрович | Николаева София Олеговна |
Петров Виталий Юрьевич

1557 ₽ [ЗАПИСАТЬСЯ](#)

ПОСМОТРЕТЬ НА КАРТЕ

Сортировать По убыванию цены

Применить коэффициенты

Клиника Клиника не выбрана

Пол врача Не выбран

Врач Бранч не выбран

Цена, ₽ 253 - 1557

Детская Да Не важно

ПРИМЕНИТЬ

Рассечение уздечки языка у детей
[TEST] Клиника
ул.Попова, 65
Виктор Александров Павлович | Екатерина Мария Игоревна |
Юрий Иван Петрович | Иванов Петр Сергеевич |
Иванов Сергей Петрович | Николаева София Олеговна |
Петров Виталий Юрьевич

1287 ₽ [ЗАПИСАТЬСЯ](#)

Рассечение уздечки языка у детей
[TEST] Клиника
ул.283-летия Барнаула, 123
Виктор Александров Павлович | Екатерина Мария Игоревна |
Юрий Иван Петрович | Иванов Петр Сергеевич |
Иванов Сергей Петрович | Николаева София Олеговна |
Петров Виталий Юрьевич

1287 ₽ [ЗАПИСАТЬСЯ](#)

Рассечение уздечки языка у детей
им по-старинке
ул.Петрова, 123
Виктор Александров Павлович | Екатерина Мария Игоревна |
Юрий Иван Петрович | Иванов Петр Сергеевич |
Иванов Сергей Петрович | Николаева София Олеговна |
Петров Виталий Юрьевич

1184 ₽ [ЗАПИСАТЬСЯ](#)

Рисунок 3.11 – Страница списка предложений по выбранной услуге

В правой части страницы отображается фильтр, в котором реализована возможность ввода коэффициентов.

Попробуем найти услугу без использования реализованного метода оптимизации критериев поиска. При введенных нами значениях фильтра было найдено только одно предложение (рисунок 3.12).

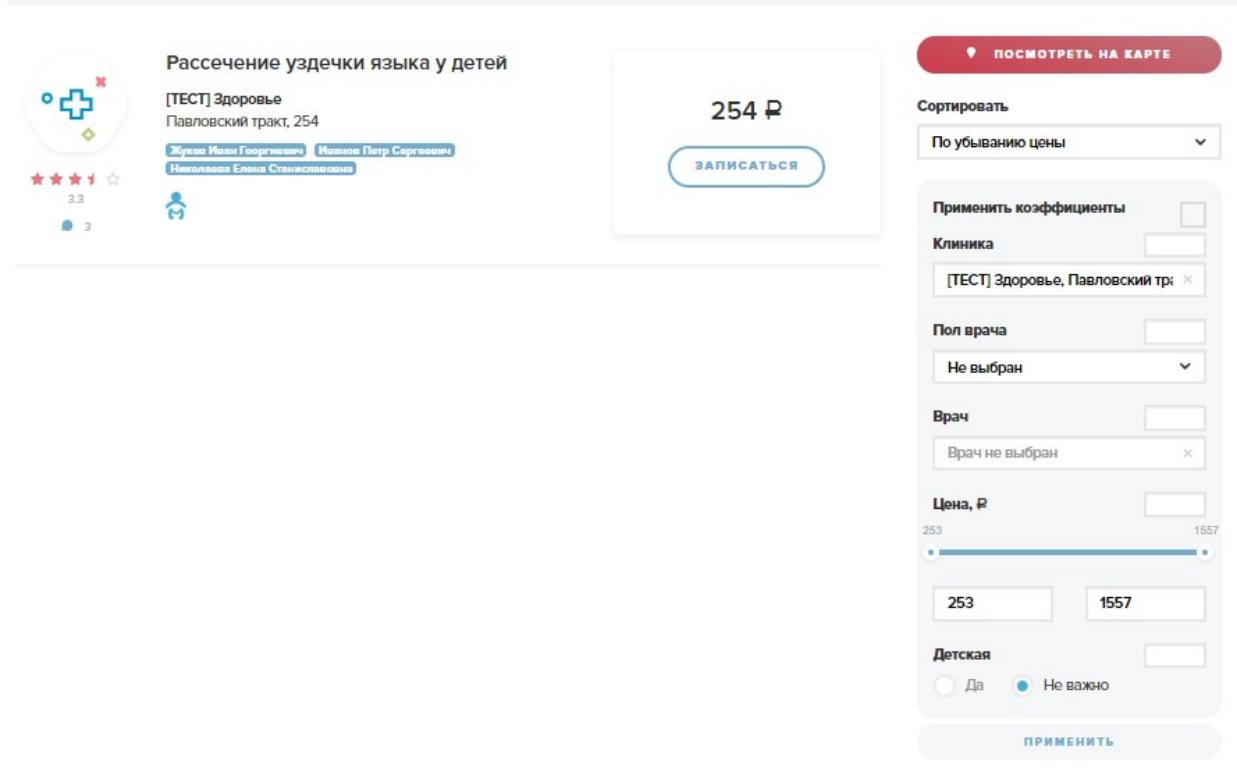
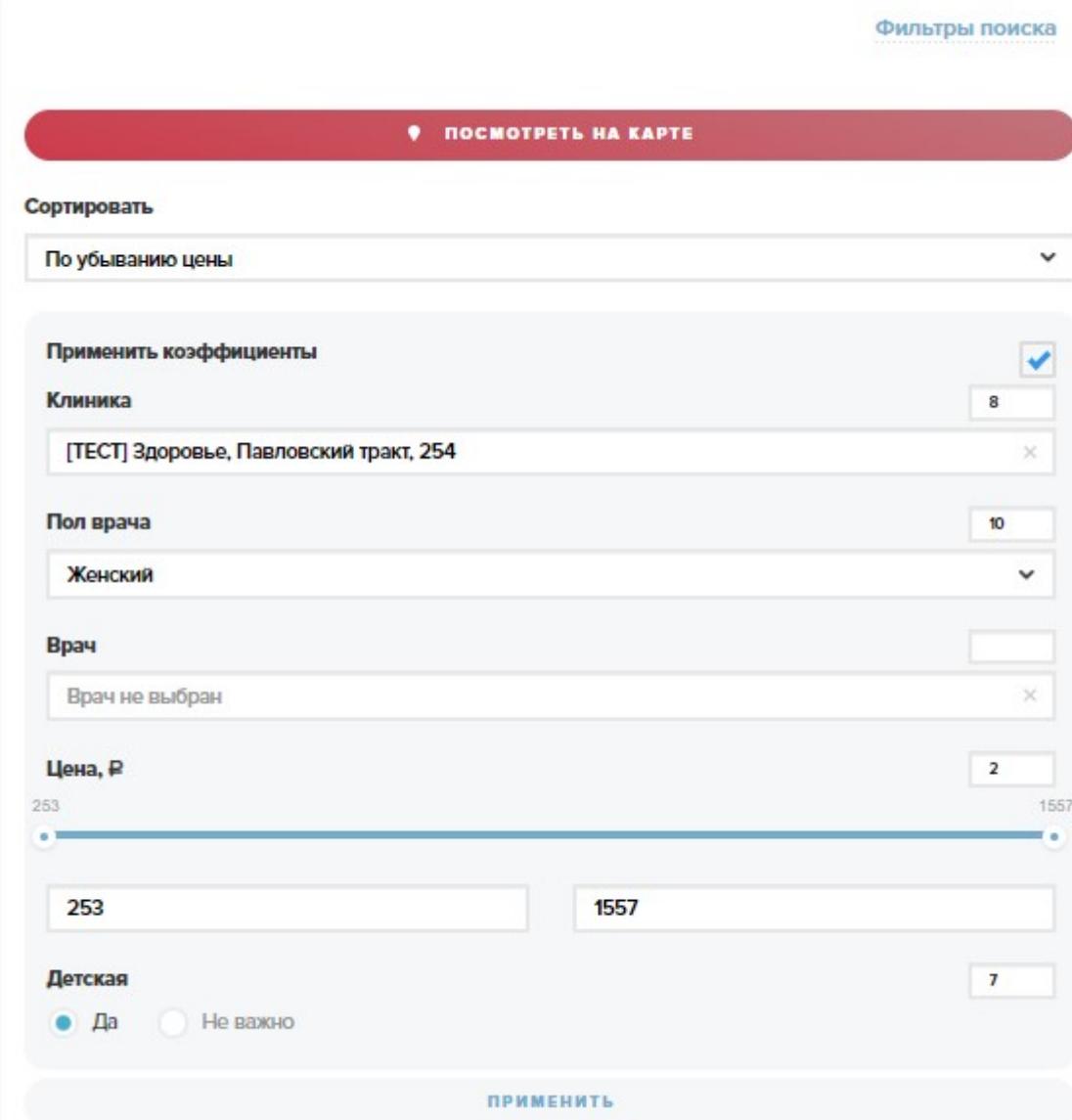


Рисунок 3.12 – Поиск предложений без использования метода аддитивной свертки

Теперь попробуем указать коэффициенты для поиска. Чем больше значение коэффициента для определенного критерия, тем большую значимость для ЛПР имеет этот критерий. Иными словами, мы можем внести число «10» для того, чтобы показать системе, что нам определенный критерий важен, и число «1», чтобы указать наименее важный критерий (рисунок 3.13).

ПОСМОТРЕТЬ НА КАРТЕ

Сортировать

По убыванию цены

Применить коэффициенты

Клиника

[ТЕСТ] Здоровье, Павловский тракт, 254

Пол врача

Женский

Врач

Врач не выбран

Цена, ₽

253



2

1557

253

1557

Детская

 Да Не важно

7

ПРИМЕНЕНИЕ

Рисунок 3.13 – Заполненный фильтр со значениями коэффициентов

После нажатия на кнопки «Применить» система отобразила список вариантов, которые удовлетворяют нашему поиску (рисунок 3.14).

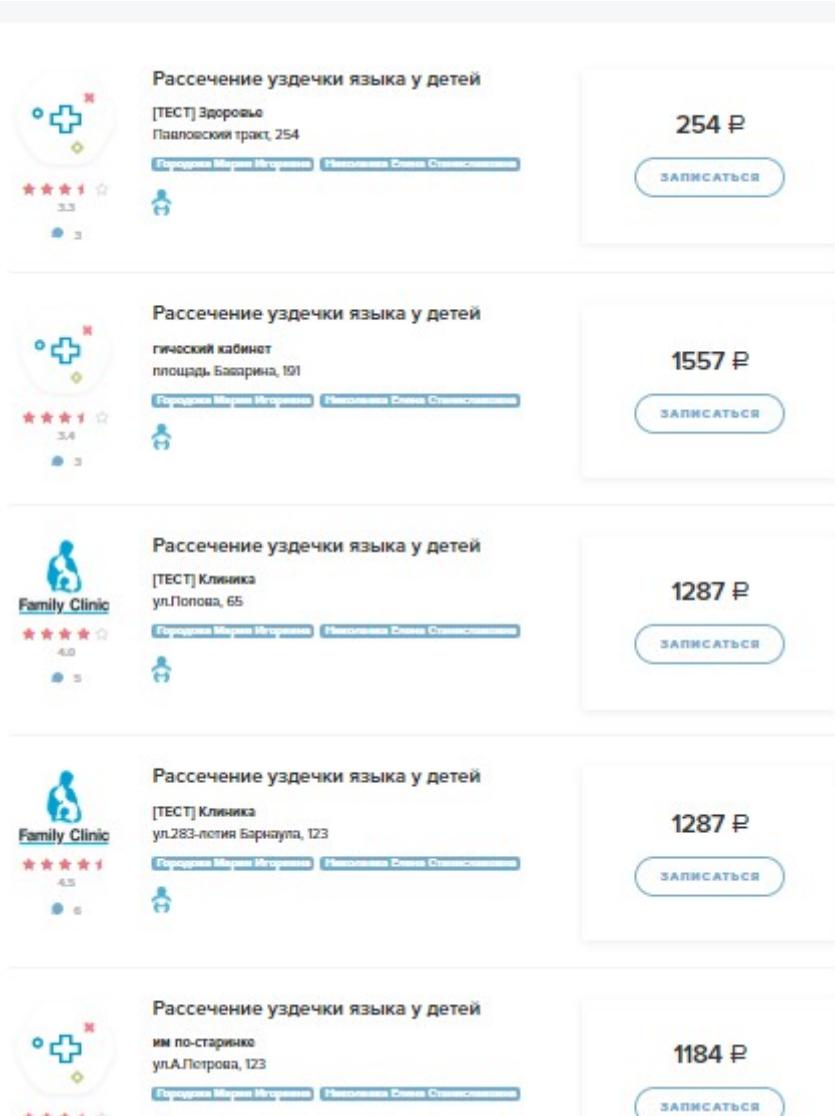


Рисунок 3.14 – Список предложений, полученный с использованием метода оптимизации критериев поиска

Следовательно, пользователю предоставляется более широкий список для выбора предложений, которые подходят по его запросу.

3.2 Метод главного критерия

3.2.1 Описание алгоритма работы метода

Алгоритм работы метода:

- 1) Пользователь выбирает услугу. Получаем список предложений по заданной услуге;

- 2) Главным критерием указываем цену услуги;
- 3) Далее значения остальных критериев нормируем по вышеуказанной формуле (1.14);
- 4) Значения критериев, для которых выполняется неравенство $f_n > k * \max(f_n)$, будут учитываться для определения оптимального решения. Здесь k – пороговое значение для частных критериев;
- 5) Выполняем анализ частных критериев для поиска предложений A_i , которые удовлетворяют условию $f_n > k * \max(f_n)$;
- 6) Если вариантов предложений по услуге получилось более 1, то проводим анализ значений главного критерия. Оптимальным решением считаем вариант, для которого $f_1(x) \rightarrow \min$. Было принято решение не максимизировать значения главного критерия, а минимизировать, так как главным критерием является цена услуги.

3.2.2 Реализация метода

Пошаговый вызов методов от контроллера страницы предложений до функции, в которой реализован метод главного критерия, представлен на рисунках 3.2–3.4. В том случае, если пользователь указал необходимость использования метода главного критерия, то вызывается метод, представленный на рисунке 3.15.

Находим максимальные и минимальные значения для критериев, которые не являются главными: средний рейтинг клиники и средний рейтинг врача. Затем производим нормализацию данных критериев и сравнение с пороговым значением, равным 0,7. Варианты предложений, нормализованные значения средних рейтингов которых, больше значения порогового коэффициента, будут учитываться в последующей обработке. Если в полученном списке имеются предложения с различной ценой, то согласно алгоритму метода главного крите-

рия, с результирующий список попадут приложения только с минимальной ценой.

```
private List<OfferBean> calcMethod2(List<OfferBean> offers, OfferFilter offersFilter) {
    if(offers.isEmpty()){
        return offers;
    }
    Double lowerVal = offersFilter.getGradeCalc();
    Double maxClinic = offers
        .stream()
        .map(o -> o.getAverageGrade())
        .max((o1, o2) -> o1.compareTo(o2))
        .get();
    Double minClinic = offers
        .stream()
        .map(o -> o.getAverageGrade())
        .min((o1, o2) -> o1.compareTo(o2))
        .get();
    Double maxDoctor = offers
        .stream()
        .map(o -> o.getDoctors())
        .stream()
        .map(d -> d.getAverageGrade())
        .max((d1, d2) -> d1.compareTo(d2))
        .orElse(0D)
        .max((o1, o2) -> o1.compareTo(o2))
        .get();
    Double minDoctor = offers
        .stream()
        .map(o -> o.getDoctors())
        .stream()
        .map(d -> d.getAverageGrade())
        .min((d1, d2) -> d1.compareTo(d2))
        .orElse(0D)
        .min((o1, o2) -> o1.compareTo(o2))
        .get();
    offers = offers
        .stream()
        .filter(o -> Objects.equals(maxClinic, minClinic) || ((o.getAverageGrade() - minClinic) / (maxClinic - minClinic)) > lowerVal)
        .collect(Collectors.toList());
    for(OfferBean bean : offers){
        List<OfferBeanDoctor> doctors = bean.getDoctors();
        doctors = doctors
            .stream()
            .filter(d -> Objects.equals(maxDoctor, minDoctor) || ((d.getAverageGrade() - minDoctor) / (maxDoctor - minDoctor)) > lowerVal)
            .collect(Collectors.toList());
        bean.setDoctors(doctors);
    }
    Integer minPrice = offers.stream().map(o -> o.getPrice()).min((o1, o2) -> o1.compareTo(o2)).get();
    offers = offers
        .stream()
        .filter(o -> o.getPrice().equals(minPrice))
        .collect(Collectors.toList());
    return offers;
}
```

Рисунок 3.15 – Исходный код метода главного критерия

3.2.3 Пользовательский сценарий использования метода

Для того чтобы продемонстрировать работу метода, протестируем его на конкретном примере.

Заходим на страницу списка услуг выбранной категории (рисунок 3.16). Выбираем услугу «КТ энтерография». На рисунке 3.17 представлен список предложений по данной услуге.

Категории услуг → Компьютерная томография (КТ, МСКТ)

Услуга	Цена, ₽	Действие
КТ энтерография	312 - 1 657 ₽	Записаться
Виртуальная колоноскопия	202 - 1 900 ₽	Записаться
КТ головного мозга	359 - 1 662 ₽	Записаться
КТ кости	1 008 - 1 974 ₽	Записаться

Рисунок 3.16 – Список услуг, относящихся к категории «Компьютерная томография (КТ, МСКТ)»

Услуги → КТ энтерография

Клиника	Адрес	Цена, ₽	Действие
Family Clinic	ул. Попова, 65	1657 ₽	Записаться
Family Clinic	ул. 283-летия Барнаула, 123	1657 ₽	Записаться
Здоровье	Павловский тракт, 254	963 ₽	Записаться

ПОСМОТРЕТЬ НА КАРТЕ

Сортировать

По убыванию цены

Метод главного критерия

Метод свертки критериев

Клиника

Врач

Цена, ₽

Детская

Рисунок 3.17 – Страница списка предложений для услуги «КТ энтерография»

Рассмотрим измененный внешний вид фильтра (рисунок 3.18). Для удобства выставления порядка использования методов были добавлены специальные поля. Чтобы указать порядок использования метода (1 или 2), необходимо кликнуть мышкой в нужное поле. Метод главного критерия не использует выбранную пользователем информацию из фильтра.

The screenshot shows a user interface for filtering search results. At the top is a red bar with a location pin icon and the text 'ПОСМОТРЕТЬ НА КАРТЕ' (View on Map). Below it is a section titled 'Сортировать' (Sort) with a dropdown menu set to 'По убыванию цены' (Sort by price in descending order). The main area contains several filter sections with dropdown menus and checkboxes:

- Метод главного критерия**: A checkbox is present.
- Метод свертки критериев**: A checkbox is present.
- Клиника**: A dropdown menu shows 'Клиника не выбрана' (Clinic not selected).
- Пол врача**: A dropdown menu shows 'Не выбран' (Not selected).
- Врач**: A dropdown menu shows 'Врач не выбран' (Doctor not selected).
- Цена, ₽**: A slider with values 311 and 1657, and input fields showing the same numbers.
- Детская**: Radio buttons for 'Да' (Yes) and 'Не важно' (It's not important), with 'Не важно' selected.

At the bottom is a blue button labeled 'ПРИМЕНИТЬ' (Apply).

Рисунок 3.18 – Измененный фильтр поиска предложений

Чтобы выполнить поиск предложения методом главного критерия необходимо установить «1» напротив названия метода (рисунок 3.19) и нажать на кнопку «Применить».

Метод главного критерия 1

Метод свертки критериев

Клиника ×

Клиника не выбрана

Пол врача

Не выбран ▼

Врач ×

Врач не выбран

Цена, ₽

311 1657

311 1657

Детская

Да Не важно

ПРИМЕНить

Рисунок 3.19 – Фильтра для использования метода главного критерия

На рисунке 3.20 изображен список предложений до использования фильтра.

Family Clinic

КТ энтерография
[ТЕСТ] Клиника
ул.Лопатина, 65
Городова Мария Игоревна
Журова Ирина Геннадьевна
Макаров Петр Сергеевич

1657 ₽ **ЗАПИСАТЬСЯ**

N!Q cosmetics

КТ энтерография
[ТЕСТ] Медицинский центр
ул.Красный площадь, 2
Волков Александр Павлович
Николаева Елена Станиславовна
Петров Виталий Юрьевич

694 ₽ **ЗАПИСАТЬСЯ**

N!Q cosmetics

КТ энтерография
[ТЕСТ] Медицинский центр
ул.Красных партизан, 236
Волков Александр Павлович
Николаева Елена Станиславовна
Петров Виталий Юрьевич

694 ₽ **ЗАПИСАТЬСЯ**

будущего

КТ энтерография
ул.Ноградская, 13
Журова Ирина Геннадьевна
Николаева Елена Станиславовна
Петров Виталий Юрьевич

466 ₽ **ЗАПИСАТЬСЯ**

им по-старинке

КТ энтерография
ул.А.Петрова, 123
Волков Александр Павлович
Городова Мария Игоревна
Макаров Петр Сергеевич

312 ₽ **ЗАПИСАТЬСЯ**

Рисунок 3.20 – Список предложений по услуге «КТ энтерография»

На рисунке 3.21 изображен список предложений, после использования метода главного критерия. Исходя из входных данных – средних рейтингов врачей и клиник, цен услуги приложение вернуло единственный оптимальный вариант.

N!Q cosmetics

КТ энтерография
[ТЕСТ] Медицинский центр
ул.Красная площадь, 2
Волков Александр Павлович Николаева Елена Станиславовна
Петров Виталий Юрьевич

694 ₽ **ЗАПИСАТЬСЯ**

Рисунок 3.21 – Список предложений после запуска работы метода

4 Исследования разработанного модуля для расчета параметров оптимизации при выборе коммерческой клиники

Анализ применения методов, приведенный в данной работе, производится на тестовом стенде для услуги «Суточное мониторирование артериального давления».

Список предложений, по вышеуказанной услуге без применения фильтров, включает в себя восемь клиник, в каждой из которых возможен прием трех врачей. Следовательно, список состоит из 24 предложений (рисунок 4.1-4.2).

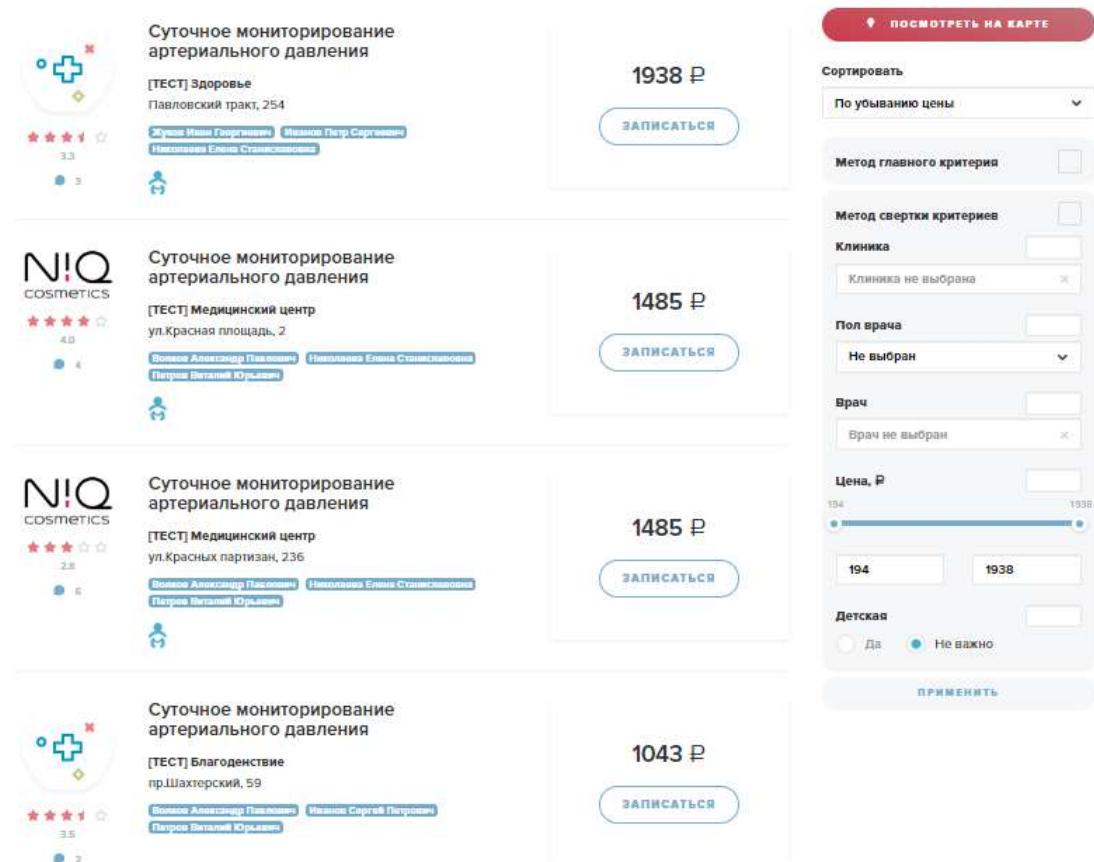


Рисунок 4.1 – Список предложений по услуге «Суточное мониторирование артериального давления». Часть 1

Суточное мониторирование артериального давления

[TEST] Стоматологический кабинет
площадь Баварина, 191

Городова Мария Игоревна, Манюк Петр Сергеевич,
Иванов Сергей Петрович

898 ₽

ЗАПИСАТЬСЯ

Суточное мониторирование артериального давления

[TEST] Клиника будущего
ул.Ноградская, 18

Жуков Иван Георгиевич, Николаева Елена Станиславовна,
Петров Виталий Юрьевич

226 ₽

ЗАПИСАТЬСЯ

Суточное мониторирование артериального давления

[TEST] Клиника
ул.Попова, 65

Городова Мария Игоревна, Жуков Иван Георгиевич,
Манюк Петр Сергеевич

195 ₽

ЗАПИСАТЬСЯ

Суточное мониторирование артериального давления

[TEST] Клиника
ул.283-летия Барнаула, 123

Городова Мария Игоревна, Жуков Иван Георгиевич,
Манюк Петр Сергеевич

195 ₽

ЗАПИСАТЬСЯ

Рисунок 4.2 – Список предложений по услуге «Суточное мониторирование артериального давления». Часть 2

Применив фильтр без учета методов решения ЗМКО, получим пустой список (рисунок 4.3). Это доказывает как факт, что применение метода поиска по полному совпадению, не всегда эффективен.

К сожалению, предложения по заданным критериям не найдены!

Пожалуйста, попробуйте уточнить условия поиска.



Сортировать

По убыванию цены ▾

Метод главного критерия

Метод свертки критериев

Клиника 5

[ТЕСТ] Клиника, ул.Попова, 65

Пол врача 5

Женский

Врач 5

Николаева Елена Станиславовна

Цена, ₽ 5



194 1000

Детская

Да Не важно

ПРИМЕНİТЬ

Рисунок 4.3 – Результат применения фильтра без использования методов решения ЗМКО

Применим только метод аддитивной свертки, задав в качестве входных данных веса для каждого критерия (рисунок 4.4). Значения параметров фильтра не были изменены, но, учитывая уровень значимости (вес) критериев для пользователя, метод выдал в качестве результата 2 предложения.

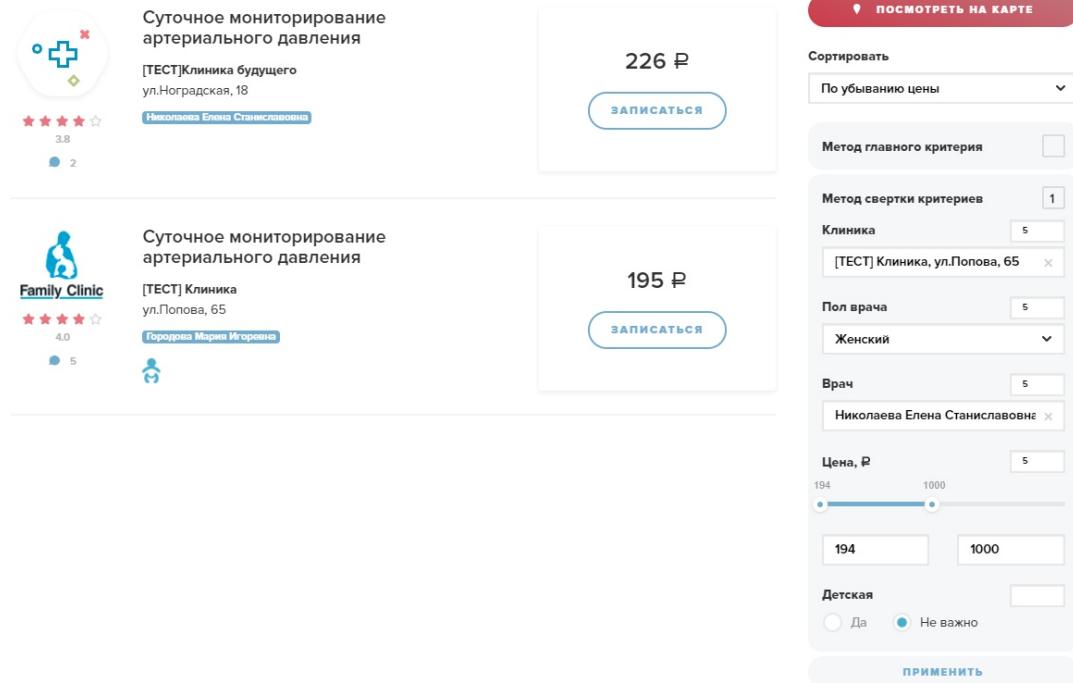


Рисунок 4.4 – Результат применения метода аддитивной свертки

Метод главного критерия не использует в своей работе значения фильтра. Он учитывает только средние рейтинги врачей и услуг, а также цены услуг. Результат работы метода главного критерия представлен на рисунке 4.5.

The screenshot shows a search results page for 'Суточное мониторирование артериального давления' (24-hour blood pressure monitoring). There are two entries listed:

- [TEST] Клиника**, ул.Попова, 65:
 - Городова Мария Игоревна, Иванов Петр Сергеевич
 - 195 ₽
 - ЗАПИСАТЬСЯ**
- [TEST] Клиника**, ул.283-летия Барнаула, 123:
 - Городова Мария Игоревна, Иванов Петр Сергеевич
 - 195 ₽
 - ЗАПИСАТЬСЯ**

To the right, a sidebar allows sorting by price (descending) and filtering by clinic name ([TEST] Клиника), gender (Female), doctor (Николаева Елена Станиславовна), and price range (194 - 1000 ₽). A 'Детская' (Child) filter is also present.

Рисунок 4.5 – Результат применения метода главного критерия

Далее применим оба метода в следующем порядке: 1 - метод аддитивной свертки, 2 – метод главного критерия (рисунок 4.6).

Услуги → Суточное мониторирование артериального давления

ПОСМОТРЕТЬ НА КАРТЕ

Сортировать

По убыванию цены

Метод главного критерия 1

Метод свертки критериев 2

Клиника 5

[TEST] Клиника, ул.Попова, 65

Пол врача 5

Женский

Врач 5

Николаева Елена Станиславовна

Цена, ₽ 5

194 1000

194 1000

Детская

Да Не важно

ПРИМЕНИТЬ

Рисунок 4.6 – Результат последовательного применения метода аддитивной свертки и метода главного критерия

Рассмотрим применение обоих рассмотренных методов, но в другом порядке (рисунок 4.7). Результат полностью совпал с результатом на рисунке 4.6.

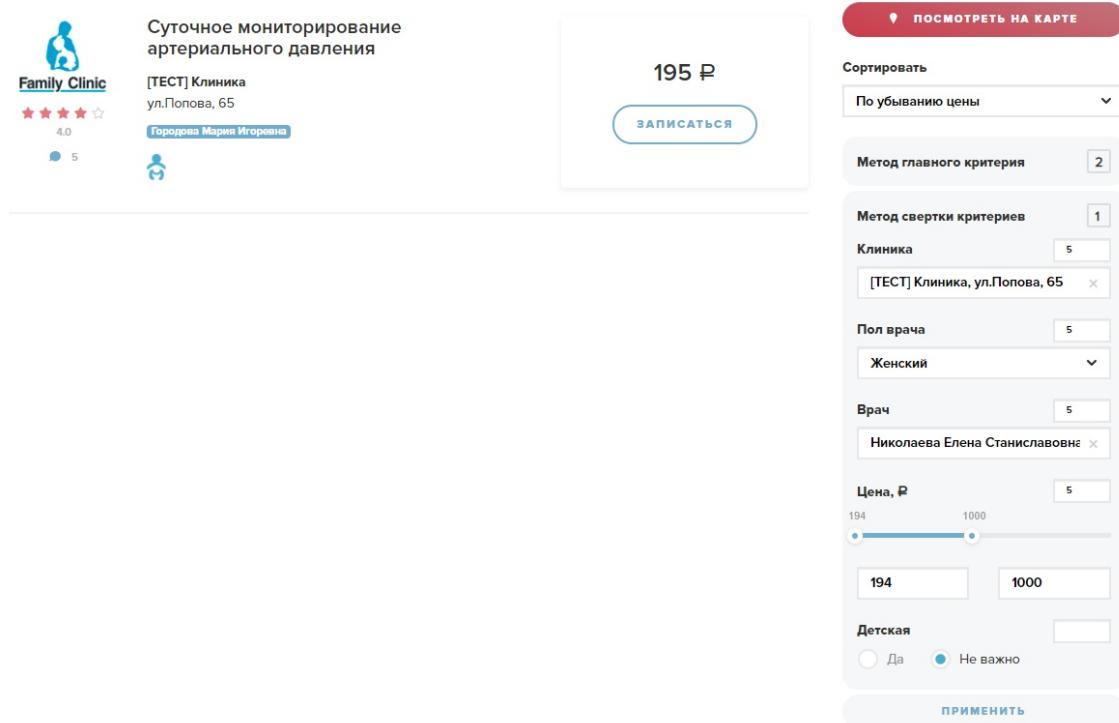


Рисунок 4.7 – Результат последовательного применения метода главного критерия и метода аддитивной свертки

Далее попробуем изменить диапазон значений параметра «Цена» в фильтре. Выставим минимальную цену равную 200 рублям.

Запустим работу модуля, указав следующий порядок выполнения методом: 1 – метод аддитивной свертки, 2 – метод главного критерия. Результат работы представлен на рисунке 4.8. В качестве оптимального варианта модуль выдал только 1 предложение. Так как сначала мы выполнили поиск предложений методом аддитивной свертки, итоговое предложение попало в список входных параметров для метода главного критерия.

The screenshot shows a search interface for medical services. On the left, there is a circular icon with a cross and a star, followed by the service name "Суточное мониторирование артериального давления" and its details: "[ТЕСТ] Клиника будущего, ул.Ноградская, 18" and "Николаева Елена Станиславовна". Below this are ratings: 3.8 stars and 2 reviews. In the center, the price is listed as "226 ₽" with a "ЗАПИСАТЬСЯ" button. To the right, there are several filter options:

- ПОСМОТРЕТЬ НА КАРТЕ** (View on map)
- Сортировать** (Sort): "По убыванию цены" (Sort by price: highest first)
- Метод главного критерия** (Main criterion method): Set to 2.
- Метод свертки критериев** (Criterion summary method): Set to 1.
- Клиника** (Clinic): Set to 5, with "[ТЕСТ] Клиника, ул.Попова, 65" selected.
- Пол врача** (Doctor's gender): Set to 5, with "Женский" (Female) selected.
- Врач** (Doctor): Set to 5, with "Николаева Елена Станиславовна" selected.
- Цена, ₽** (Price, ₽): A slider from 200 to 1000, with "200" and "1000" input fields.
- Детская** (Child): Two radio buttons: "Да" (Yes) and "Не важно" (It doesn't matter), with "Не важно" selected.

A "ПРИМЕНить" (Apply) button is located at the bottom right of the filter panel.

Рисунок 4.8 – Результат последовательного применения метода аддитивной свертки и метода главного критерия с измененным диапазоном цен

Попробуем запустить модуль с теми же значениями фильтра, но указав другой порядок выполнения методов. На рисунке 4.9 представлен результат работы модуля. В данном случае не было найдено ни одного оптимального варианта.

К сожалению, предложения по заданным критериям не найдены!

Пожалуйста, попробуйте уточнить условия поиска.



Сортировать

По убыванию цены ▾

Метод главного критерия 1

Метод свертки критериев 2

Клиника 5

[TEST] Клиника, ул.Попова, 65 ×

Пол врача 5

Женский ▾

Врач 5

Николаева Елена Станиславовна ×

Цена, ₽ 5

200 1000

200 1000

Детская

Да Не важно

ПРИМЕНИТЬ

Рисунок 4. 9 – Результат последовательного применения метода главного критерия и метода аддитивной свертки с измененным диапазоном цен

Таким образом, можно сделать вывод о том, что не целесообразно применять метод главного критерия перед выполнением метода аддитивной свертки. Так как возможна частичная потеря списка предложений, который подается на вход подпрограмме, которая реализует алгоритм работы метода аддитивной свертки.

Заключение

В ходе данной работы был разработан модуль для повышения эффективности работы сервиса бронирования, записи в коммерческие клиники. Повышение эффективности достигнуто за счет применения, при отборе предложений по услугам, методов решения задачи многокритериальной оптимизации. В качестве базового сервиса онлайн – записи на прием к врачу был взят сервис TalonDo (talondo.ru).

В ходе исследования методов решения задачи МКО был проведен анализ:

- 1) возможных проблем, возникающих при использовании методов решения задач МКО;
- 2) существующих методов решения задачи МКО;
- 3) возможность применения методов для решения задачи оптимизации выбора коммерческой клиники.

Далее были рассмотрены аналоги сервиса TalonDo и выявлены основные недостатки данных сервисов. Также были описаны основные функциональные возможности сервисов онлайн – записи. Полученные результаты анализа были использованы при разработке сервиса TalonDo.

Разработанный модуль поддерживает оптимизацию поиска предложений для записи к врачу по следующим методам решения задач МКО:

- 1) метод аддитивной свертки;
- 2) метод главного критерия.

Данные методы могут использоваться как отдельно, так и последовательно, в любом порядке. Но исследование, проведенное после разработки модуля, показало, что целесообразным, при последовательном использовании, является только применение сперва метода аддитивной свертки, а затем метода главного критерия.

Программное обеспечение было протестировано по различным пользовательским сценариям. Были исправлены выявленные ошибки.

В дальнейшем исследование может быть продолжено и добавлены другие методы решения задачи МКО.

Список использованных источников

- 1) Аттетков А.В. Методы оптимизации [Текст] : Учеб. для вузов / А.В. Аттетков, С.В. Галкин, В.С Зарубин; под ред. В.С. Зарубина, А.П. Крищенко [2-е изд.]. – МГТУ им. Н.Э. Баумана, 2003. – 440с.
- 2) Шешунова Е.В. Многокритериальные системы оптимизации в агропромышленном комплексе [Текст] // Вестник МГУ. 2017. №1. URL: <https://cyberleninka.ru/article/n/mnogokriterialnye-sistemy-optimizatsii-v-agropromyshlennom-komplekse> (дата обращения: 16.06.2019).
- 3) Розен В.В. Математические модели принятия решений в экономике. [Текст] : учеб. пособие / В.В. Розен. – М.: Книжный дом «Университет», «Высшая школа», 2002. – С. 54–67.
- 4) Минюк С.А. Математические методы и модели в экономике [Текст] : учеб. пособие / С.А. Минюк, Е.А. Ровба, К.К. Кузьмич. – Мн.: ТетраСистемс, 2002. – 432 с.
- 5) Пантелеев А.В. Методы оптимизации. Практический курс [Текст] / А.В. Пантелеев, Т.А. Летова. – М.: Логос, 2011. – 110с.
- 6) Соловьев В.И. Методы оптимальных решений [Текст] / Соловьев В.И. – М. : Финансовый университет, 2012. – 364с.
- 7) Струченков В.И. Методы оптимизации [Текст] / В.И. Струченков – М.: Экзамен, 2005. – 256с.
- 8) Черноруцкий И.Г. Методы оптимизации и принятия решений [Текст] / И.Г. Черноруцкий – СПб.: Лань, 2001. – 384с.
- 9) Гамидов, Рафаэль Гусейн Оглы Методы решения некоторых многокритериальных задач оптимизации [Текст] дис. ...канд. физ.–мат. наук: 01.01.09 / Гамидов, Рафаэль ГусейнОглы. – К.,2007. – 119с.
- 10) Стопченко Г.І. Задачі та концепції методів багатокритеріальних рішень в інтелектуальних системах [Текст] / Г.І. Стопченко, І.А. Макрушан , С. В. Білан // Біоніка інтелекту: наук.–техн. журнал. – 2010. – № 1 (72). – С. 122–125.

- 11) Корнеенко В.П. Методы оптимизации [Текст] / В.П. Корнеенко— М.: Высшая школа, 2007. – 664с.
- 12) Ларичев О.И. Теория и методы принятия решений, а также хроника событий в Волшебных Странах [Текст] / О.И. Ларичев— М.: Логос, 2000. – 296с.
- 13) Штойер Р. Многокритериальная оптимизация [Текст] / Р. Штойер: пер. с англ. – М. : Радио и связь, 1992. – 504с. – (Теория, вычисления и приложения).
- 14) Подиновский В.В. Парето–оптимальные решения многокритериальных задач [Текст] / В.В. Подиновский, В.Д. Ногин. – М.: Наука, 1982. –64с.
- 15) Монахов, В.В. Язык программирования Java и среда NetBeans [Электронный ресурс] : учеб. пособие — Электрон. дан. — Москва : , 2016. — 450 с. — Режим доступа: <https://e.lanbook.com/book/100544>. — Загл. с экрана.
- 16) Моргунов, Е. П. M79 Язык SQL. Базовый курс [Текст] : учеб.–практ. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова; Postgres Professional. — М., 2017. — 257 с.
- 17) Сильвио, М. Bootstrap в примерах [Электронный ресурс] — Электрон. дан. — Москва : ДМК Пресс, 2017. — 314 с. — Режим доступа: <https://e.lanbook.com/book/93573>. — Загл. с экрана.
- 18) Евдокимов, А.П. Создание сайтов своими руками на Bootstrap [Электронный ресурс] : учеб. пособие / А.П. Евдокимов, М.В. Финков. — Электрон. дан. — Санкт–Петербург : Наука и Техника, 2017. — 240 с. — Режим доступа: <https://e.lanbook.com/book/101545>. — Загл. с экрана.
- 19) Васильев Ф.П. Методы оптимизации [Текст] / Ф.П. Васильев— М.: Факториал Пресс, 2002. – 824с.
- 20) Подиновский В.В. Введение в теорию важности критериев в многокритериальных задачах принятия решений [Текст] / В.В. Подиновский – М.: Физматлит, 2007. – 64с.
- 21) Сидоренко Е.А. Особенности многокритериальной оптимизации при выборе пациентом коммерческой клиники с применением сервиса TalonDo» [Текст] / Сидоренко Е.А., Сучкова Л.И. // Материалы региональной

молодежной научно-практической конференции «Программно-техническое обеспечение автоматизированных систем» /под ред. Л.И. Сучковой. – Барнаул: Изд-во АлтГТУ, 2018. – С. 130-134.

22) Науменко М.А. Особенности моделирования процесса принимаемых управлеченческих решений [Текст] / М.А. Науменко // Транспорт. Транспортные сооружения. Экология. – 2013. – №1. – С. 98-104.

23) Сидоренко Е.А. Применение методов многокритериальной оптимизации при решении задачи выбора коммерческой клиники для сервиса TalonDo / Сидоренко Е.А., Сучкова Л.И. // Материалы XX Международной научно-технической конференции «Измерение, контроль, информатизация» – 2019. – Барнаул, АлтГТУ, 2019 г. – [Электронный ресурс].

24) Сидоренко Е.А. Применение метода главного критерия для решения многокритериальной задачи выбора пациентом коммерческой клиники с применением сервиса TalonDo / Сидоренко Е.А., Сучкова Л.И. // Материалы 16 Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых «Наука и молодежь – 2019» – Барнаул, АлтГТУ, 2019 г. – [Электронный ресурс].

25) Лотов А.В. Многокритериальные задачи принятия решений. Метод достижимых целей [Текст] / А.В. Лотов, И.И. Поспелова – М.: МАКС Пресс, 2008. – 197 с.

26) Гермейер Ю.Б. Введение в теорию исследования операций. М.: Наука, 1971.

27) Математические методы и модели в экономике: Учеб. пособие / [С.А. Минюк, Е.А. Ровба, К.К. Кузьмич]. – М.: ТетраСистемс, 2002. – 432с.

28) Гарипов В.Р. Многокритериальная оптимизация систем управления сложными объектами методами эволюционного поиска: дис. ... канд. техн. наук: 05.13.01 / Гарипов Валерий Рашилович. – Красноярск, 1999. – 166с.

29) Монахов, В.В. Язык программирования Java и среда NetBeans [Электронный ресурс] : учеб. пособие — Электрон. дан. — Москва: , 2016.

- 30) PostgreSQL 9.6 High Performance by Ibrar Ahmed, Gregory Smith. Copyright 2017 Packt Publishing. Published by Packt Publishing Ltd. Livery Place, 35 Livery Street, Birmingham, B3 2PB, UK, ISBN 978-1-78439-297-0
- 31) Евдокимов, А.П. Создание сайтов своими руками на Bootstrap [Электронный ресурс] : учеб. пособие / А.П. Евдокимов, М.В. Финков. — Электрон. дан. — Санкт-Петербург : Наука и Техника, 2017. — 240 с. — Режим доступа: <https://e.lanbook.com/book/101545>. — Загл. с экрана.
- 32) Сильвио, М. Bootstrap в примерах [Электронный ресурс] — Электрон. дан. — Москва : ДМК Пресс, 2017. — 314 с. — Режим доступа: <https://e.lanbook.com/book/93573>. — Загл. с экрана.
- 33) Beginning Spring by Mert Caliskan, Kenan Sevindik. Copyright 2015 by John Wiley, Inc. 10475 Crosspoint Boulevard, Indianapolis, IN 46256
- 34) Бородакий Ю.В. Основы теории систем управления. Исследование и проектирование / Ю.В. Бородакий, Ю.Г. Лободинский. — М.: Радио и связь, 2004. — 256с.
- 35) Гуменникова А.В. Адаптивные поисковые алгоритмы для решения сложных задач многокритериальной оптимизации :дис. ... кандидата технических наук : 05.13.01 / Гуменникова Александра Викторовна. — К., 2006. — 129с.
- 36) Горбунов, В.М. Теория принятия решений: учебное пособие / В.М. Горбунов. — Томск: Изд-во ТПУ, 2010. — 67 с.
- 37) Федунец, Н.И. Теория принятия решений / Н.И. Федунец, В.В. Куприянов. — М.: Горная книга, 2005. — 218 с.
- 38) Кини Р.Л. Принятие решений при многих критериях: предпочтения и замещения / Р.Л. Кини, Х. Райфа. — М.: Радио и связь, 1981. — 560с.
- 39) Гудов, А.М. Базы данных и системы управления базами данных. Программирование на языке PL/SQL: учеб. пособие [Электронный ресурс] : учеб. пособие / А.М. Гудов, С.Ю. Завозкин, Т.С. Рейн. — Электрон. дан. — Кемерово : КемГУ, 2010. — 133 с. — Режим доступа: <https://e.lanbook.com/book/30135>. — Загл. с экрана.

40) Дэвид, Х. Разработка приложений Java EE 7 в NetBeans 8 [Электронный ресурс] : рук. — Электрон. дан. — Москва : ДМК Пресс, 2016. — 348 с. — Режим доступа: <https://e.lanbook.com/book/97342>. — Загл. с экрана.

Приложение А

Задание на выполнение магистерской диссертации

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования «Алтайский государственный технический университет
им. И. И. Ползунова»

УТВЕРЖДАЮ
Заведующий кафедрой ИВТИИБ
А.Г. Якунин
подпись Ф.И.О.

ЗАДАНИЕ № 12

на выполнение магистерской диссертации
бакалаврской работы, дипломной работы (дипломного проекта), магистерской диссертации

по направлению подготовки (специальности) 09.04.01 Информатика и вычислительная техника

студенту группы 8ИВТ-71 Сидоренко Евгений Александровне
фамилия, имя, отчество

Тема Разработка программного обеспечения для расчета параметров оптимизации при выборе коммерческой клиники

Утверждено приказом ректора от 11.10.2018 г. № Л-2918

Срок выполнения работы

Задание принял к исполнению: _____ Е.А. Сидоренко
подпись инициалы, фамилия

Барнаул 2019

1 Исходные данные

Исходными данными являются язык программирования Java, персональный компьютер с ОС Linux, техническое задание на разработку.

2 Содержание разделов магистерской диссертации и календарный график ее выполнения:

Наименование разделов работы и их содержание	Трудо- ём- кость, %	Срок выполнения	Консультант (фамилия, инициалы, подпись)
Расчётно-пояснительная записка			
Введение. Отразить актуальность, цель и задачи исследования	5	27.03.2019	Сучкова Л.И.
Глава 1. Обзор и анализ методов оптимизации критериев для принятия решения	15	10.04.2019	Сучкова Л.И.
Глава 2. Особенности реализации сервиса для выбора коммерческой клиники	15	24.04.2019	Сучкова Л.И.
Глава 3. Реализация модифицированных методов расчета параметров оптимизации	35	22.05.2019	Сучкова Л.И.
Глава 4. Исследования разработанного модуля для расчета параметров оптимизации при выборе коммерческой клиники	20	01.06.2019	Сучкова Л.И.
Заключение	5	06.06.2019	Сучкова Л.И.
Графический материал Слайды	5	16.06.2019	Сучкова Л.И.

3 Научно-библиографический поиск

3.1 По научно-технической литературе просмотреть: РЖ «Горизонты образования» за последние 2 года.

3.2 По нормативной литературе просмотреть указатели государственных и отраслевых стандартов за последний год.

3.3 Патентный поиск провести за 2 года по странам _____ Россия _____

Руководитель _____
подпись

Л.И. Сучкова
инициалы, фамилия

Приложение Б

Акт о внедрении ПО



ООО "ЛинксСофт"
г. Барнаул, пр. Социалистический, 69
+7 905 989 5161
info@ccnpp.ru

Справка о внедрении результатов магистерской диссертации

Настоящим удостоверяю, что разработанный в рамках магистерской диссертации Сидоренко Е.А. модуль для расчета параметров оптимизации при выборе коммерческой клиники успешно прошел стадию тестирования и в настоящее время внедрен в сервис онлайн – записи к врачу TalonDo.

Генеральный директор
ООО «ЛинксСофт»

Лысенко А.Ф.



Рисунок Б.1 – Акт о внедрении

Приложение В

Исходный код

```
@RequestMapping(value = "/offers/find", method = RequestMethod.GET)
public String findOffers(Model model, HttpServletRequest request, HttpServletResponse response,
    @RequestParam(name = "serviceld") Integer talondoServiceId,
    @RequestParam(name = "minPrice", required = false) Integer minPrice,
    @RequestParam(name = "maxPrice", required = false) Integer maxPrice,
    @RequestParam(name = "sorter", required = false) ServicesOffersSorter sorter,
    @RequestParam(name = "gender", required = false) Gender gender,
    @RequestParam(name = "doctorId", required = false) Integer doctorId,
    @RequestParam(name = "locationId", required = false) Integer locationId,
    @RequestParam(name = "child", required = false) boolean isChild,
    @RequestParam(name = "page", defaultValue = "0") Integer page,
    //calc fields
    @RequestParam(name = "filters", required = false) String filters,
    @RequestParam(name = "withCalc", required = false) String withCalc,
    @RequestParam(name = "clinicCalc", required = false, defaultValue = "0") Integer clinicCalc,
    @RequestParam(name = "doctorGenderCalc", required = false, defaultValue = "0") Integer doctorGenderCalc,
    @RequestParam(name = "doctorCalc", required = false, defaultValue = "0") Integer doctorCalc,
    @RequestParam(name = "priceCalc", required = false, defaultValue = "0") Integer priceCalc,
    @RequestParam(name = "childCalc", required = false, defaultValue = "0") Integer childCalc) {
    sorter = ServicesOffersSorter.getSorterOrDefault(sorter);
    List<Integer> filtersList = new Gson().fromJson(filters, new TypeToken<List<Integer>>() {
        }.getType());
    gender = Gender.getThisOrDefaultIfNull(gender);
    City city = basicBeanHelper.getDefaultCityFromCookie(request, response);
    OfferFilter filter = new OfferFilter(
        serviceService.getSimpleServiceBeanById(talondoServiceId),
        city.getId(),
        minPrice, maxPrice,
        doctorService.getSimpleDoctorBean(doctorId),
        gender,
        locationService.getSimpleClinicWithAddressBeanById(locationId),
        isChild);
    filter.setCalcValues(filtersList, clinicCalc, doctorGenderCalc, doctorCalc, priceCalc, childCalc);
    Page<OfferBean> offerBeans = offerService.getOffersForTalondoService(filter, new PageRequest(page, OFFERS_TO_SHOW),
        sorter);
    model.addAttribute("offerBeans", offerBeans);

    model.addAttribute("page", page);
    model.addAttribute("pages", offerBeans.getTotalPages());
    model.addAttribute("currentSorter", sorter);
    model.addAttribute("filter", filter);
    return "offers/listResultsPanel :: resultsPanel";
}
```

```

@Override
@Transactional
public PageImpl<OfferBean>getOffersForTalondoService(OfferFilteroffersFilter, Pageable pageable, ServicesOffersSorter sorter) {
    if (offersFilter.isWithCalc()) {
        List<GroupedCache2>cachesWithoutDoctor = cacheService.getServiceOffersCachesWithoutDoctor(offersFilter, sorter);
        List<OfferBean> offers = cachesWithoutDoctor
            .stream()
            .map(group -> {
ForeignServiceforeignService = foreignServiceService.getForeignServiceById(group.getForeignServiceId());
OrganizationLocationorganizationLocation = locationService.getOrganizationLocationById(group.getLocationId());
                return new OfferBean(
foreignService,
organizationLocation,
                cacheService.getServiceOfferBeanDoctors(group.getForeignServiceId(), group.getLocationId(), offersFilter),
                averageGradeService.findAverageGradeByTargetIdAndType(CommentType.CLINIC, group.getLocationId())
            );
        })
        .collect(Collectors.toList());
    for(Integer filter :offersFilter.getFilters()){
        switch(filter){
            case 1:
                offers = calcMethod1(offers, offersFilter);
                break;
            case 2:
                offers = calcMethod2(offers, offersFilter);
                break;
        }
    }
    return new PageImpl(offers, pageable, offers.size());
} else {
    List<OfferBean> offers = calcSimple(offersFilter, pageable, sorter);
    return new PageImpl(offers);
}
private List<OfferBean>calcSimple(OfferFilteroffersFilter, Pageable pageable, ServicesOffersSorter sorter) {
    Page<GroupedCache2>cachesWithoutDoctor = cacheService.getServiceOffersCachesWithoutDoctor(offersFilter, pageable,
sorter);
    List<OfferBean> offers = cachesWithoutDoctor
        .getContent()
        .stream()
        .map(group -> {
ForeignServiceforeignService = foreignServiceService.getForeignServiceById(group.getForeignServiceId());
OrganizationLocationorganizationLocation = locationService.getOrganizationLocationById(group.getLocationId());
                return new OfferBean(
foreignService,
organizationLocation,
                cacheService.getServiceOfferBeanDoctors(group.getForeignServiceId(), group.getLocationId(), offersFilter),
                averageGradeService.findAverageGradeByTargetIdAndType(CommentType.CLINIC, group.getLocationId())
            );
        })

```

```

.collect(Collectors.toList());
return offers; }

private List<OfferBean> calcMethod1(List<OfferBean> offers, OfferFilteroffersFilter) {
    if(offers.isEmpty()) { return offers; }

    Double lowerVal = offersFilter.getGradeCalc();
    Integer clinicCalc = offersFilter.getClinicCalc();
    Integer doctorGenderCalc = offersFilter.getDoctorGenderCalc();
    Integer doctorCalc = offersFilter.getDoctorCalc();
    Integer priceCalc = offersFilter.getPriceCalc();
    Integer childCalc = offersFilter.getChildCalc();

    Integer sum = clinicCalc + doctorGenderCalc + doctorCalc + priceCalc + childCalc;
    for (OfferBean bean : offers) {

        Double calc = 0d;
        if (clinicCalc != 0) {
            if (Objects.equals(bean.getOrganizationLocationId(), offersFilter.getClinicId())) {
                calc += 1d / sum * clinicCalc;
            }
        }
        if (childCalc != 0) {
            if (Objects.equals(bean.isChild(), offersFilter.isChild())) {
                calc += 1d / sum * childCalc;
            }
        }
        if (priceCalc != 0) {
            if (offersFilter.getPrice().isFull()) {
                if (offersFilter.getPrice().getLowerInRubles()
                    <= bean.getPrice() && bean.getPrice()
                    <= offersFilter.getPrice().getUpperInRubles()) {
                    calc += 1d / sum * priceCalc;
                }
            } else if (offersFilter.getPrice().hasMax()) {
                if (bean.getPrice() <= offersFilter.getPrice().getUpperInRubles()) {
                    calc += 1d / sum * priceCalc;
                }
            } else if (offersFilter.getPrice().hasMin()) {
                if (offersFilter.getPrice().getLowerInRubles() <= bean.getPrice()) {
                    calc += 1d / sum * priceCalc;
                }
            }
        }
        Double localDoctorGenderCalc = 0d;
        Double localDoctorCalc = 0d;
        List<OfferBeanDoctor> doctors = bean.getDoctors();
        if (doctors != null) {
            if (doctorGenderCalc != 0) {
                if (doctors.stream().filter(d ->d.getDoctorGender() == offersFilter.getDoctorsGender()).count() > 0) {
                    localDoctorGenderCalc = 1d / sum * doctorGenderCalc;
                }
            }
        }
    }
}

```

```

        calc += localDoctorGenderCalc;
    }
}

if (doctorCalc != 0) {
    if (doctors.stream().filter(d ->Objects.equals(d.getDoctorId(), offersFilter.getDoctorId())).count() > 0) {
localDoctorCalc = 1d / sum * doctorCalc;
        calc += localDoctorCalc;
    }
}

if (calc >lowerVal) {
    final Double sumCalc = calc;
    final Double ldgcalc = localDoctorGenderCalc;
    final Double ldCalc = localDoctorCalc;
    List<OfferBeanDoctor>newDoctors = doctors.stream().filter(d -> {
        return (sumCalc
            - (d.getDoctorGender() == offersFilter.getDoctorsGender() ? 0 : ldgcalc)
            - (Objects.equals(d.getDoctorId(), offersFilter.getDoctorId()) ? 0 : ldCalc)) >lowerVal;
    }).collect(Collectors.toList());
    bean.setDoctors(newDoctors);
}

bean.setCalc(calc);
}

offers = offers.stream().filter(b ->b.getCalc() >lowerVal).collect(Collectors.toList());
offers.sort((o1, o2) -> o2.getCalc().compareTo(o1.getCalc()));
return offers;
}

private List<OfferBean> calcMethod2(List<OfferBean> offers, OfferFilteroffersFilter) {
    if(offers.isEmpty()){
        return offers;
    }

    Double lowerVal = offersFilter.getGradeCalc();
    Double maxClinic = offers
        .stream()
        .map(o ->o.getAverageGrade())
        .max((o1, o2) -> o1.compareTo(o2))
        .get();

    Double minClinic = offers
        .stream()
        .map(o ->o.getAverageGrade())
        .min((o1, o2) -> o1.compareTo(o2))
        .get();

    Double maxDoctor = offers
        .stream()

```

```

.map(o ->o.getDoctors()
    .stream()
    .map(d ->d.getAverageGrade())
    .max((d1, d2) -> d1.compareTo(d2))
    .orElse(0D))
    .max((o1, o2) -> o1.compareTo(o2))
    .get();
}

Double minDoctor = offers
    .stream()
    .map(o ->o.getDoctors()
        .stream()
        .map(d ->d.getAverageGrade())
        .min((d1, d2) -> d1.compareTo(d2))
        .orElse(0D))
        .min((o1, o2) -> o1.compareTo(o2))
        .get();
}

offers = offers
    .stream()
    .filter(o ->Objects.equals(maxClinic, minClinic) || ((o.getAverageGrade() - minClinic) / (maxClinic - minClinic)) >lowerVal)
    .collect(Collectors.toList());
}

for(OfferBean bean : offers){
    List<OfferBeanDoctor> doctors = bean.getDoctors();
    doctors = doctors
        .stream()
        .filter(d ->Objects.equals(maxDoctor, minDoctor) || ((d.getAverageGrade() - minDoctor) / (maxDoctor - minDoctor)) >lowerVal)
        .collect(Collectors.toList());
    bean.setDoctors(doctors);
}
}

Integer minPrice = offers.stream().map(o ->o.getPrice()).min((o1, o2) -> o1.compareTo(o2)).get();
offers = offers
    .stream()
    .filter(o ->o.getPrice().equals(minPrice))
    .collect(Collectors.toList());
}

return offers;
}

@Getter
public class OfferBean {
    private final Integer foreignServiceId;
    /** Название услуги организации.*/
    private final String title;
    /** Описание услуги в организации.*/
    private final String description;
    /** Цена услуг в копейках.*/
    private final int intPrice;
    /** Форматированная цена услуги.*/
}

```

```

private final Integer price;
/** Наименование организации.*/
private final String organizationTitle;
private final Integer organizationLocationId;
/** Адрес филиала организации.*/
private final String organizationLocationAddress;
/** Логотип организации.*/
private final String logo;
@Setter
private List<OfferBeanDoctor> doctors;
private final boolean hasNull;
private Double averageGrade = 0D;
private Integer commentCount = 0;
private final Double mapLatitude;
private final Double mapLongitude;
private final boolean isChild;
@Setter
private Double calc;
public OfferBean(ForeignService foreignService, OrganizationLocation location, List<OfferBeanDoctor> doctors, AverageGrade averageGrade) {
    this.foreignServiceId = foreignService.getId();
    this.title = foreignService.getTitle();
    this.description = foreignService.getDescription();
    this.intPrice = foreignService.getPrice();
    this.price = (int) Math.ceil(foreignService.getPrice() / 100.0);
    this.organizationTitle = location.getOrganization().getShortName();
    this.organizationLocationId = location.getId();
    this.organizationLocationAddress = location.getAddress();
    this.logo = location.getOrganization().getEncodedMainLogo();
    this.doctors = doctors;
    this.hasNull = doctors.get(doctors.size() - 1).getDoctorId() == null;
    if (averageGrade != null) {
        this.averageGrade = (Math.round((averageGrade.getAttentionGrade() + averageGrade.getPriceGrade() + averageGrade.getSkillGrade()) / 3.0) / 10.0);
        this.commentCount = averageGrade.getCommentsNumber();
    }
    this.mapLatitude = location.getLatitude();
    this.mapLongitude = location.getLongitude();
    this.isChild = foreignService.isChild();
}
@Getter
public class OfferBeanDoctor {
    private final Integer doctorId;
    private final String doctorFio;
    private final Gender doctorGender;
    private Double averageGrade = 0D;
}

```

```

    public OfferBeanDoctor(Doctor doctor, AverageGradeaverageGrade) {
        this.doctorId = doctor == null ? null : doctor.getId();
        this.doctorFio = doctor == null ? null : doctor.getFullFio();
        this.doctorGender = doctor == null ? null : doctor.getGender();
        if (averageGrade != null) {
            this.averageGrade = (Math.round((averageGrade.getAttentionGrade() + averageGrade.getPriceGrade() + averageGrade.getSkillGrade()) / 3.0) / 10.0);
        }
    }

    @Override
    public boolean equals(Object obj) {
        if (!(obj instanceof OfferBeanDoctor)) {
            return false;
        }
        if (!getClass().equals(obj.getClass())) {
            return false;
        }
        OfferBeanDoctor other = (OfferBeanDoctor) obj;
        return Objects.equals(this.getDoctorId(), other.getDoctorId());
    }

    }}  @Override
    @Transactional
    public Page<GroupedCache2> getServiceOffersCachesWithoutDoctor(OfferFilter filter, Pageable pageable, ServicesOffersSorter
sorter) {
    QForeignServiceforeignService = QForeignService.foreignService;
    QService service = QService.service;
    QOrganizationLocationorganizationLocation = QOrganizationLocation.organizationLocation;
    int queryLimitWithOneEntityFromNextPage = pageable.getPageSize() + 1;
    List<GroupedCache2> caches = new JPAQuery<>(entityManager)
        .select(Q_DAYS_CACHE.foreignService().id, Q_DAYS_CACHE.location().id, Q_DAYS_CACHE.foreignService().price,
Q_DAYS_CACHE.foreignService().title)
        .distinct()
        .from(Q_DAYS_CACHE)
        .join(foreignService).on(foreignService.id.eq(Q_DAYS_CACHE.foreignService().id))
        .join(service).on(service.id.eq(foreignService.talondoService().id), service.id.eq(filter.getTalondoServiceBean().getId()))
        .join(organizationLocation).on(organizationLocation.id.eq(Q_DAYS_CACHE.location().id), organizationLoca-
tion.city().id.eq(filter.getCityId()))
        .where(
            Q_DAYS_CACHE.available.isTrue(),
            createIntegerInRangeExpression(filter.getPrice(), foreignService.price),
            createDoctorGenderExpression(filter.getDoctorId(), filter.getDoctorsGender(), Q_DAYS_CACHE.doctor()),
            createExpression(filter.getClinicId(), Q_DAYS_CACHE.location().id),
            filter.isChild() ? foreignService.isChild.isTrue() : null
        )
        .offset(pageable.getOffset())
        .limit(queryLimitWithOneEntityFromNextPage)
        .orderBy(sorter.getOrderSpecifier())
        .fetch()
        .stream()
}

```

```

.map(tuple -> new GroupedCache2(
tuple.get(Q_DAYS_CACHE.foreignService().id),
tuple.get(Q_DAYS_CACHE.location().id)
))
.collect(Collectors.toList());
booleannextPageExist = caches.size() > pageable.getPageSize();
if (nextPageExist) {
caches.remove(pageable.getPageSize());
}
return new PageImpl(caches, pageable, caches.size());
@Override
@Transactional
public List<OfferBeanDoctor>getServiceOfferBeanDoctors(Integer foreignServiceId, Integer locationId, OfferFilter filter) {
QForeignServiceforeignService = QForeignService.foreignService;
QDoctor doctor = QDoctor.doctor;
QService service = QService.service;
QOrganizationLocationorganizationLocation = QOrganizationLocation.organizationLocation;
List<OfferBeanDoctor> list;
if (filter.isWithCalc()) {
list = new JPAQuery<>(entityManager)
.select(doctor)
.distinct()
.from(Q_DAYS_CACHE)
.leftJoin(doctor).on(Q_DAYS_CACHE.doctor().id.eq(doctor.id))
.join(foreignService).on(foreignService.id.eq(Q_DAYS_CACHE.foreignService().id), foreignService.id.eq(foreignServiceId))
.join(service).on(service.id.eq(foreignService.talondoService().id), service.id.eq(filter.getTalondoServiceBean().getId()))
.join(organizationLocation).on(organizationLocation.id.eq(Q_DAYS_CACHE.location().id), organizationLocation.city().id.eq(filter.getCityId()), organizationLocation.id.eq(locationId))
.where(
Q_DAYS_CACHE.available.isTrue())
.orderBy(new OrderSpecifier(Order.ASC, doctor.surname),
new OrderSpecifier(Order.ASC, doctor.forename),
new OrderSpecifier(Order.ASC, doctor.patronymic))
.fetch()
.stream()
.map(doc -> new OfferBeanDoctor(doc, doc == null ? null : averageGradeService.findAverageGradeByTargetIdAndType(CommentType.DOCTOR, doc.getId())))
.collect(Collectors.toList());
} else {
list = new JPAQuery<>(entityManager)
.select(doctor)
.distinct()
.from(Q_DAYS_CACHE)
.leftJoin(doctor).on(Q_DAYS_CACHE.doctor().id.eq(doctor.id))
.join(foreignService).on(foreignService.id.eq(Q_DAYS_CACHE.foreignService().id), foreignService.id.eq(foreignServiceId))
}
}

```

```

        .join(service).on(service.id.eq(foreignService.talondoService().id), service.id.eq(filter.getTalondoServiceBean().getId()))
        .join(organizationLocation).on(organizationLocation.id.eq(Q_DAYS_CACHE.location().id), organizationLocation.city().id.eq(filter.getCityId()), organizationLocation.id.eq(locationId))
        .where(
            Q_DAYS_CACHE.available.isTrue(),
            createIntegerInRangeExpression(filter.getPrice(), foreignService.price),
            createDoctorGenderExpression(filter.getDoctorId(), filter.getDoctorsGender(), Q_DAYS_CACHE.doctor()),
            createExpression(filter.getClinicId(), Q_DAYS_CACHE.location().id),
            filter.isChild() ? foreignService.isChild.isTrue() : null
        )
        .orderBy(new OrderSpecifier(Order.ASC, doctor.surname),
            new OrderSpecifier(Order.ASC, doctor.forename),
            new OrderSpecifier(Order.ASC, doctor.patronymic))
        .fetch()
        .stream()
        .map(doc -> new OfferBeanDoctor(doc, doc == null ? null : averageGradeService.findAverageGradeByTargetIdAndType(CommentType.DOCTOR, doc.getId())))
        .collect(Collectors.toList());
    }    return list;  }
}

```

ФГБОУ ВО АлтГТУ

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Группа 8ИВТ-71

Направление 09.04.01 – Информатика и вычислитель-
ная техника

Профиль Программно-техническое обеспечение авто-
матизированных систем

СИДОРЕНКО ЕВГЕНИЯ АЛЕКСАНДРОВНА

2019 г.