Санкт-Петербургский государственный университет

Кафедра теории программирования

Сахаров Алексей Александрович

Выпускная квалификационная работа бакалавра

Разработка многопользовательской системы учета с возможностью автоматизированного построения расписаний

Направление 010400

Прикладная математика и информатика

Научный руководитель,

старший преподаватель

Стученков А.Б.

Санкт-Петербург

2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ..... 4
ПОСТАНОВКА ЗАДАЧИ..... 5
ОБЗОР ЛИТЕРАТУРЫ.....7
ГЛАВА 1. СРАВНЕНИЕ СУЩЕСТВУЮЩИХ СРЕДСТВ УЧЕТА ДЛЯ МЕДИЦИНСКИХ УЧРЕЖДЕНИЙ..... 8

1.1. Описание текущей организации рабочего процесса.....8
1.2. Обзор готовых решений..... 9
1.3. Описание выбранного способа решения поставленных задач.....11

ГЛАВА 2. МАТЕМАТИЧЕСКИЕ АЛГОРИТМЫ КАК СПОСОБ АВТОМАТИЗАЦИИ ПОСТРОЕНИЯ РАСПИСАНИЙ.....12

2.1. Предпосылки для применения автоматизации.....12
2.2. Основные понятия теории расписаний.....13
2.3. Применение ТР в рамках текущей задачи.....16
2.4. Описание алгоритма построения расписания сотрудников.....18

ГЛАВА 3. ЭТАПЫ РАЗРАБОТКИ ПРИЛОЖЕНИЯ.....20

3.1. Способы и средства разработки.....20
3.2. Проектирование структуры данных.....20
3.3. Создание базовых функциональных элементов.....23
3.4. Построение архитектуры приложения.....27
3.5. Логика работы и интерфейс приложения.....29
3.6. Создание дополнительной функциональности.....33

ГЛАВА 4. ТЕСТИРОВАНИЕ СИСТЕМЫ.....34

4.1. Тестирование заявленных функций.....34
4.2. Тестирование модульности..... 35
4.3. Тестирование вспомогательных систем.....36
4.4. Тестирования раздела построения расписаний..... 36

ВЫВОДЫ..... 37
ЗАКЛЮЧЕНИЕ..... 38
СПИСОК ЛИТЕРАТУРЫ.....39
ПРИЛОЖЕНИЕ 1.....40
ПРИЛОЖЕНИЕ 2.....41

ВВЕДЕНИЕ

В современном мире большинство компаний стремится по максимуму использовать достижения информационных технологий для организации ведения производства. Уже сложно представить организацию, которая пользуется схемой документооборота, основанной только на бумажных носителях. Каждый старается подобрать подходящий для себя инструмент. Но если в компании нет специалиста, который способен следить за современными тенденциями в развитии информационных технологий, то очень часто применяются далеко не оптимальные инструменты. Либо под задачу используется такое ПО, которое изначально создано для иных целей.

С развитием компании неверный выбор начального направления в применении информационных технологий может загнать в тупик. И для того, чтобы выбраться из этого тупика необходим глубокий анализ существующих на данный момент инструментов и технологий, к использованию которых стоит перейти для оптимизации работы компании.

В наше время существует множество инструментов для решения различных проблем, и очевидным кажется, что достаточно только выбрать подходящее решение. Но не во всех ситуациях стандартные решения могут решить существующий комплекс задач. А средства, обобщающие множество решений, не всегда удобны для конечного пользователя, так как они нагружены избыточной функциональностью.

В подобных ситуациях для конкретных организаций хорошо подходит разработка новых узконаправленных модульных инструментов. Узконаправленность позволяет исключить избыточную функциональность, а модульность дает возможность к гибкому расширению.

ПОСТАНОВКА ЗАДАЧИ

Цель работы – разработать приложение (систему учета), позволяющее систематизировать текущий рабочий процесс медицинского учреждения (Санкт-Петербургское государственное бюджетное учреждение здравоохранения «Городской психоневрологический диспансер № 7 (со стационаром)» подразделение Психотерапевтический центр) для упрощения оформления и обслуживания клиентов путем применения современных информационных технологий.

Итоговая система должна обладать следующими функциями:

- многопользовательский доступ;
- учет пациентов организации;
- формирование групп пациентов для проведения групповых приемов с возможностью фиксировать посещаемость данных групповых приемов;
- возможность ведения журнала задач для учета заметок о действиях сотрудника, которые ему требуется произвести по отношению к тому или иному пациенту;
- наличие административного раздела для учета пользователей и их прав доступа;
- управление расписаниями работы сотрудников (в процессе разработки данный пункт был дополнен возможностью автоматизированного построения расписаний);
- формирование отчетов о работе сотрудников и приеме пациентов на основании расписаний, хранимых в системе.

Создаваемая система учета изначально предназначена для организации: «Городской психоневрологический диспансер № 7 (со стационаром)» подразделение Психотерапевтический центр, однако может быть применена и для других медицинских учреждений.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить анализ существующей системы учета данных, применяемой в организации;

- провести исследование существующих методов разработки многопользовательских систем с учетом необходимой функциональности;
- выполнить проектирование структуры базы данных и функциональных элементов приложения с выделением отдельных разделов системы;
- выбрать средства разработки и разработать систему учета с реализацией всех необходимых функций;
- изучить методы автоматизации построения расписаний;
- реализовать автоматизированный механизм построения расписаний в качестве одной из функций системы учета;
- выполнить анализ влияния новой системы учета на работу медицинского учреждения (психотерапевтический центра).

Обзор литературы

В работе была использована следующая литература:

1. Конвей Р.В., Максвелл В.Л., Миллер Л.В. Теория расписаний.

Эта книга описывает базовые определения теории расписаний. В ней рассмотрены принципы построения расписаний основных типов. Предлагаются решения для различных задач.

2. Лазарев А.А., Гафаров Е.Р. Теория расписаний.

Учебное пособие, разработанное в МГУ. В нем рассматривается классификация задач теории расписаний и предлагаются алгоритмы для их решения. Построенная классификация помогает определить тип рассматриваемой задачи и выбирать подходящие методы её решения.

3. Brucker P. Scheduling Algorithms.

Книга, написанная Питером Брукером, освещает современные методы решения задач теории расписаний. Она делает уклон в реализацию рассматриваемых алгоритмов. Глава 11 посвящена работе с многопроцессорными задачами. Эта теория нашла свое применений в данной работе.

4. Smith J. Patterns - WPF Apps With The Model-View-ViewModel Design Pattern.

Статья в журнале MSDN описывает шаблоны разработки программного обеспечения на базе WPF (Windows Presentation Foundation).

ГЛАВА 1. СРАВНЕНИЕ СУЩЕСТВУЮЩИХ СРЕДСТВ УЧЕТА ДЛЯ МЕДИЦИНСКИХ УЧРЕЖДЕНИЙ

1.1. Описание текущей организации рабочего процесса

Для работы регистратуры и учета пациентов персонал медицинского центра решил использовать таблицы Microsoft Office Excel[CITATION excel \l 1033]. Каждая строка таблицы – пациент. Таблица содержит основные данные о пациенте, такие как «Номер карты», «ФИО», «Дата рождения», «Адрес», информация о лечащих врачах, о передаче карты в архив и возвращении её в работу. (см. «Приложение 1»)

С данным подходом со временем возникло несколько проблем:

- Строка информации получается достаточно длинной и немногие мониторы способны её отобразить полностью.
- При росте количества записей (до нескольких тысяч) возникла проблема с поиском информации. Стандартная строка поиска Excel сильно нагружала систему, из-за чего при попытке поиска программа могла просто зависнуть. Проблема решалась разбиением данных на несколько файлов с меньшим объемом, но это вызвало неудобства в работе.
- Из-за низкой производительности оборудования возникали ошибки при сохранении изменений данного файла. Поэтому сотрудники постоянно делали резервную копию данных на съемных носителях.
- При большой наполненности полей данной таблицы информацию становилось сложнее воспринимать, что приводило к быстрой утомляемости.

Для формирования групп и учета посещаемости использовались бумажные бланки. Они накапливались в большом объеме, что затрудняло их подсчет при формировании отчетов по посещаемости групповых занятий и отчетов о работе сотрудников.

- Расписания были организованы при помощи Excel-таблиц. Каждый лист книги – расписание одного сотрудника на текущий месяц. Ежемесячно специалист по расписанию формирует данный файл. Ему необходимо точно проследить, чтобы не возникало никаких ошибок в

сопоставлении расписаний различных сотрудников. К примеру, групповой прием могут проводить два сотрудника, а значит необходимо, чтобы этот групповой прием стоял у сотрудников в одно время. Кроме того, необходимо следить, чтобы не возникало ситуации, в которой на два разных события назначен один кабинет.

Для построения отчетов специалист собирает бланки посещений за интересующий период и вручную подсчитывает все индивидуальные и групповые приемы.

Общие проблемы данной схемы организации работы:

- низкая читабельность данных;
- отсутствие единой системы учета;
- большие затраты времени на различные действия;
- отсутствие контроля изменений.

1.2. Обзор готовых решений

Различные компании уже предлагали свои продукты для внедрения в данной организации. Причины, по которым данные продукты не были приняты к использованию описаны далее.

Сначала рассмотрим МИС «Ариадна»[CITATION Ком \l 1049]. Данная система отлично проработана, включает множество различных модулей, способных решать разнообразные задачи для организации рабочего процесса медицинского центра. Большой уклон этой системы сделан на формировании, обработке и печати документации. Судя по информации на сайте проекта[CITATION Ком \l 1049], данная система успешно работает в различных клиниках страны. Для рассматриваемой организации было предложено использовать несколько модулей этой системы. Но определяющим фактором осталась невозможность вести групповые приемы и большая стоимость системы. Хоть эта система и построена на модульной архитектуре, но если необходима какая-то часть функций модуля, то

приходится покупать полный модуль, а это значит, что в системе будут неиспользуемые функции.

Далее рассмотрим МИС «КПС «САМСОН»[CITATION ООО \l 1049]. Эта система, как и МИС «Ариадна» готова для решения задач организации работы регистратуры, хранении данных пациентов, ведение расписаний сотрудников. Главное преимущество перед предыдущим решением заключается в том, что «САМСОН» является Свободным Программным Обеспечением (СПО), а значит установка не потребует от организации финансовых затрат. Но, как и в предыдущей системе данная МИС не может вести учет групповых приемов. И в то же время имеет набор функций, которые не нужны для психотерапевтического центра.

Наиболее подходящей оказалась МИС «Авиценна»[CITATION ЗАО \l 1049]. Она имеет специальный вариант сборки, предназначенный для психотерапевтического диспансера. Эта сборка способна решать все задачи, интересующие организацию. С технической точки зрения эта система отличный вариант для полного управления работой центра. Из недостатков данной системы можно отметить только её большую (для данного центра) стоимость. Также присутствует перегруженность интерфейса функциями, которые для заказчика не нужны.

Результаты для данный систем можно обобщить:

- В некоторых существующих готовых решениях недостаточно функций для решения задач по организации работы исследуемого центра.
- Другие решения перегружены излишними функциями, которые не нужны для заказчика.
- Часть систем предоставляется платно, а так как данная организация обеспечивается из средств государственного бюджета, то у них просто нет финансовой возможности на покупку дорогостоящих систем.

1.3. Описание выбранного способа решения поставленных задач

В связи с тем, что готовые решения не подошли заказчику, было решено создавать собственную систему, которая удовлетворяет требованиям, описанным в разделе «Постановка задачи». Также было решено внедрить в готовую систему алгоритмы, позволяющие автоматизировать процесс построения расписаний для ускорения работы специалиста по расписанию.

В процессе разработки системы было решено создавать такой интерфейс, который одновременно отражает минимальный набор доступных пользователю действий. То есть, построить интерфейс так, чтобы уменьшить шанс на ошибку в выборе действия. Также эта концепция позволяет минимизировать время обучения пользователей.

Готовая система должна поддерживать возможность её расширения. А значит, архитектура приложения должна быть основана на идее модульности. То есть каждая подзадача (набор функций) встраивается в приложение, как автономная часть, минимально затрагивающая уже имеющиеся подзадачи.

Система должны быть избавлена от недостатков готовых решений. А значит:

- выполнять все заявленные требования;
- должна поставляться в организацию безвозмездно;
- не содержать избыточной функциональности, который усложняет обучение пользователей.

ГЛАВА 2. МАТЕМАТИЧЕСКИЕ АЛГОРИТМЫ КАК СПОСОБ АВТОМАТИЗАЦИИ ПОСТРОЕНИЯ РАСПИСАНИЙ

2.1. Предпосылки для применения автоматизации

На текущий момент в центре работают сотрудники с различными профессиональными обязанностями. Составление расписания для работы медицинского регистратора не содержит проблем (он отвечает за регистрацию пациентов в центре и запись их на прием). График работы данного персонала представляет собой чередование утренних и вечерних смен при пятидневной рабочей неделе. Но вместе с тем в центре работают психотерапевты, медицинские психологи, специалисты по социальной работе, медицинские сестры. Для них расписание так же представляет собой чередование утренних и вечерних смен. Но при этом вся смена разбита на события следующего типа:

- первичный прием пациента;
- повторный прием пациента;
- групповой прием пациентов;
- работа с документацией;
- диагностика;
- собрание.

Собрания назначаются на фиксированное время в течение длительного периода времени, поэтому их можно воспринимать как уменьшение доступного рабочего времени в течение смены. Первичный и повторный прием, а также работа с документацией и диагностика равноценны по времени и условиям проведения, а значит на первом этапе построения расписания их можно рассматривать как одинаковые события.

Групповые приемы вносят усложнения в процесс построения расписания. Их основная идея заключается в том, что проводить групповой

прием могут одновременно один или сразу два сотрудника. Поэтому при составлении индивидуальных расписаний сотрудников необходимо постоянно контролировать их согласованность. Также все другие события расписания не имеют «содержимого» на момент построения (не указывается пациент, для которого проходит прием) и поэтому между ними нет различий, а групповые приемы известны заранее, поэтому при их добавлении в расписание им приходится уделять большое внимание.

Описанные особенности приводят к тому, что специалисту по расписанию требуется несколько часов для построения расписания работы центра вручную. В этой работе используется идея применения алгоритмов теории расписаний для автоматизации их построения.

Теория расписаний изучена многими исследователями и с различных сторон. В рамках неё создано множество математических идей и аппаратов для решения задач различного рода. Эти идеи можно сгруппировать и применить для создания алгоритма, способного по входным параметрам построить рекомендуемое расписание для сотрудников центра. А специалисту необходимо проверить результат работы алгоритма и, возможно, внести правки в итоговое расписание. К примеру, отпуск сотрудника, праздничные и нерабочие дни и т.д.

2.2. Основные понятия теории расписаний

Определение: «*Теория расписаний* (**ТР**)– это раздел исследования операций, в котором строятся и анализируются математические модели календарного планирования (т.е. упорядочивания во времени) различных целенаправленных действий с учетом целевой функции и различных ограничений» [CITATION Лаз11 \l 1033].

Определение: *Работой* называется некоторое действие, которое необходимо обслужить. Работы состоят из *операций* – элементарных нераздельных частей. В силу того, что событие в расписании сотрудника

рассматривается как неделимое, то термины операции и работы для этой задачи эквивалентны.

Лазарев А.А. и Гафаров Е.Р. [CITATION Лаз11 \l 1049] приводят следующую классификацию задач **ТР**:

- **По типу искомого решения:**
  - Задачи *упорядочивания*. Распределение работ по исполнителям и время их выполнения уже задано. Необходимо составить расписание (или порядок) выполнения работ каждым исполнителем.
  - Задачи *согласования*. Основное внимание в этих задачах уделяется выбору продолжительности выполнения работ, времени поступления и другим параметрам.
  - Задачи *распределения* подразумевают поиск оптимального распределения работ по исполнителям.
- **По типу целевой функции:**
  - задачи с суммарными критериями оптимизации;
  - задачи с минимаксными критериями оптимизации;
  - многокритериальные задачи оптимизации;
  - задачи на построение допустимого расписания.
- **По способу задания входной информации:**
  - Детерминированные задачи (off-line). Для таких задач характерно, что все входные данные задачи точно известны, т.е. даны значения всех параметров до начала ее решения.
  - Динамические задачи (on-line). Для данных задач расписания строятся в режиме реального времени, т.е. перед началом решения задачи мы не знаем значения всех параметров. Расписание строится по частям по мере поступления новой информации. При этом в любой момент может быть понадобиться ответ о качестве построенного “частичного” расписания.
- **По разделу ТР:**
  - сетевое планирование или построение расписания для проекта, Project scheduling (PS);
  - календарное планирование или построение расписания для приборов, Machine scheduling (MS);
  - составление временных таблиц (Time Tabling);
  - доставка товаров в магазины (Shop-Floor Scheduling);

- составление расписаний движения транспортных средств (Transport Scheduling), циклические расписания для транспортных средств (Vehicle Routing); (1)
- составление расписаний спортивных мероприятий (Sports scheduling).

«Приведенные классификации задач ТР условны и лишь указывают на некоторые характерные особенности решаемых задач» [CITATION Лаз11 \l 1049].

Рассматриваемый здесь пример можно определить как детерминированную задачу упорядочивания календарного планирования с суммарными критериями оптимизации.

В качестве суммарного критерия применим суммарную ***взвешенную длительность прохождения*** работы в системе.

Обозначим p_i – длительность ***выполнения*** работы i; $p_{[i]}$ – аналогично, но *[i]* – индекс после упорядочения; W_i – длительность ***ожидания*** начала работы. Тогда $F_i = W_i + p_i$ – длительность ***прохождения*** работы i в системе.

Суммарная длительность прохождения работ в системе $\sum_{k=1}^{n} F_i$.

Если в системе определить ***вес*** работы (u_i), обозначающий приоритетность одной работы перед остальными, то можно определить ***суммарную взвешенную длительность*** прохождения работы:

$$F = \sum_{k=1}^{n} u_i F_i.$$

2.3. Применение ТР в рамках текущей задачи

Во-первых, стоит отметить, что расписание составляется на месяц, но для его составления достаточно шаблона на одну неделю, по которому формируется расписание для каждой недели месяца. Здесь не учитываются отпуска, больничные и иные несистемные события. Специалисту по расписанию оставлен доступ и к итоговому шаблону недели, и к

сформированному на месяц расписанию для внесения подобного рода корректировок.

Во-вторых, весь персонал разделен на две смены (группы): утреннюю и вечернюю. При этом порядок их работы устанавливается простым чередованием утро-вечер при переходе от одного дня недели к следующему. Поэтому для формирования расписания для обеих смен достаточно запустить алгоритм два раза, изменяя соответствующий параметр.

Выбранная математическая модель называется **SPT** (Shortest – processing – time sequencing) – упорядочением по минимуму длительностей работ [CITATION Кон75 \l 1049]. Для решения данной задачи эта модель дополняется учетом ***веса*** работы – условной единицы, обозначающей приоритет работы над остальными.

У смен есть особенность: вес работы утром противоположен весу работы вечером. Поэтому всё расписание создается по общему правилу, а затем расписание утренней смены симметрично отображается относительно середины рабочего времени.

Теорема [CITATION Кон75 \l 1033]: Минимальное значение суммарной взвешенной длительности прохождения (1) в системе достигается при расписании, для которого

$$\frac{p_{[1]}}{u_{[1]}} \leq \frac{p_{[2]}}{u_{[2]}} \leq \ldots \leq \frac{p_{[n]}}{u_{[n]}} \quad .$$

Графически этот метод представлен на рис. 1. Каждый прямоугольник – работа, а его ширина – длительность, высота – вес. Суммарная взвешенная длительность является суммой площадей прямоугольников и площади под ними, поэтому для минимизации $\boldsymbol{F}$ необходимо упорядочить работы в порядке уменьшения модуля угла вектора, лежащего на диагонали прямоугольника, к горизонтальной оси. Коэффициент угла равен $\frac{u_{[i]}}{p_{[i]}}$.

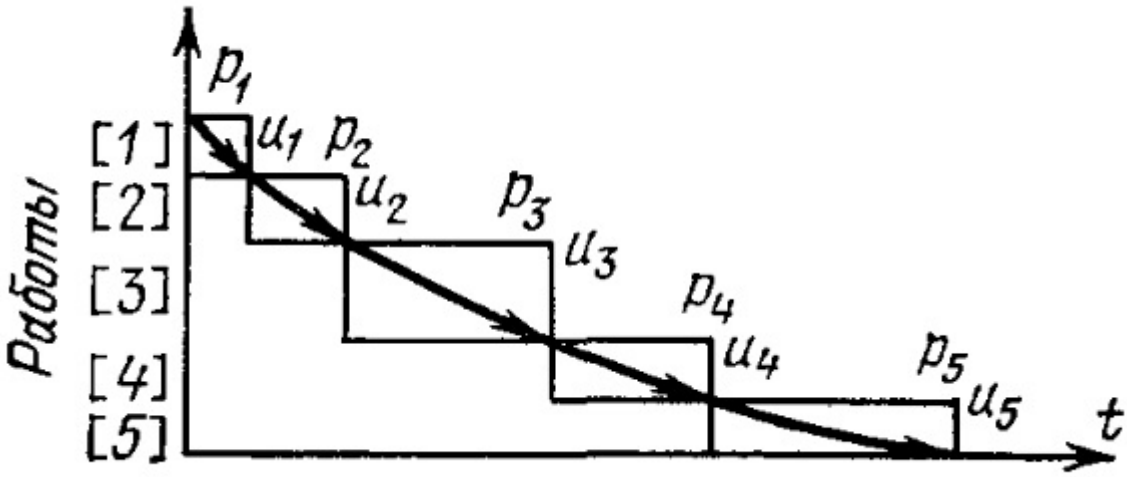


Рис. 1. Геометрическое представление

Для решения проблемы групповых занятий используем теорию, описанную Brucker' о м [CITATION 1Br07 \l 1049]. Для решения мультипроцессорных задач (т.е. задач, требующих одновременного выполнения на нескольких процессорах) он вводит следующую модификацию начальных данных.

Пусть в системе с m процессорами задано n мультипроцессорных задач $i=1,\ldots,n$. Сопоставим задаче i множество μ_i процессоров, необходимых для её выполнения. Это множество определяет **тип** задачи i. Выделим $R \le 2^m - 1$ типов задач и сгруппируем все задачи по этим типам [CITATION 1Br07 \l 1049]. Здесь под типом задачи понимается характеристика взаимно однозначная структуре порождающего множества процессоров. Тип может быть порожден как отдельным сотрудником, так и парой сотрудников.

В итоге получился набор «сущностей» (множеств, порождающих типы). Для этих «сущностей» нет требования на одновременное выполнение задачи. Есть только требования предшествования вида: групповой прием, исполняемый составной «сущностью», должен стоять после (или до) первичного приема, исполняемого одним из сотрудников, состоящим в этой «сущности». Можно использовать ту же теорию, что и в классической задаче по множеству независимых исполнителей с ограничением на последовательность выполнения.

Далее можно применить идеи группового упорядочивания, описанные в [CITATION Кон75 \l 1049]. Проведем упорядочивание между этими группами и внутри них.

2.4. Описание алгоритма построения расписания сотрудников

С учетом условий, описанных в начале параграфа 2.3, опишем алгоритм составления шаблона расписания на одну неделю.

Входными параметрами для алгоритма являются:

- таблица нагрузок сотрудника на неделю по всем видам события расписания, кроме групповых приемов (данные указываются на примере одного события, для остальных событий этого типа данные аналогичны);
- данные по групповым приемам с указанием ведущих и необходимых нагрузок на неделю (данные указываются для каждой группы);
- данные о весе событий расписания.

Предполагается, что суммарная длительность работ не превышает ограничений длительности рабочего времени. Для упорядочивания применяется взвешенная SPT модель, описанная выше.

На первом этапе создаются «сущности» − пары сотрудников. Создание расписания для них сложнее, чем для отдельных сотрудников, поэтому начнем с них.

Для каждой пары сотрудников упорядочим групповые приемы и добавим их в личное расписание обоих сотрудников с учетом сохранения условия, указанного в теореме. Причем начинать стоит с групповых приемов с наибольшим коэффициентом $\frac{u_{[i]}}{p_{[i]}}$, чтобы не возникало конфликтов. На данном этапе строится только порядок групповых приемов, без указания времени начала. В конце этого этапа проводится разбиение всего списка на дни недели.

На втором этапе для разбитых на дни групповых приемов устанавливается время начала с учетом согласованности расписаний

связанных сотрудников. Свободное время заполняется оставшимися событиями других типов. При необходимости, для соблюдения правила упорядочивания, можно изменить начало групповых приемов.

Для дней с утреней сменой делается зеркальное отражение расписания по времени.

Далее каждому событию назначается доступный кабинет.

В итоге получаем расписание, не всегда оптимальное, но допустимое и близкое к оптимальному.

ГЛАВА 3. ЭТАПЫ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

3.1. Способы и средства разработки

Для создания приложения было решено использовать язык программирования C# (.Net Framework 4.5.2) [CITATION Net \l 1049] с применением WPF-технологии (Windows Presentation Foundation) и шаблоном проектирования MVVM (Model-View-ViewModel) [CITATION Smi09 \l 1049]. Для хранения данных был использован MS SQL Server 2012 Express [CITATION MSS \l 1033]. Работа с ними производилась посредством EntityFramwork 6.1.3[CITATION Ent \l 1033] совместно с Linq.

Приложение построено на клиент-серверной архитектуре, где в качестве сервера выступает СУБД, а в качестве клиента – настольное приложение.

Далее более подробно описаны этапы разработки приложения.

3.2. Проектирование структуры данных

Проектирование структуры базы данных началось с создания двух основных «узлов»: «Сотрудник» и «Пациент». Далее, в процессе работы над программой структура всё больше усложнялась, приобретая новые таблицы и связи между ними. Итоговый вид базы данных представлен на рис. 2 и рис. 3.

Описание блоков (см. рис. 3):

1. Данные сотрудников Организации: ФИО, должность, контакты, логин, пароль.
2. Данные о пациенте: номер карты, ФИО, телефон, адрес, данные о лечащих врачах. История карты: сдача карты в архив и возвращение в работу.
3. Права доступа в системе. Шаблоны для групп прав («Регистратор», «Врач» и т.д.) и таблица назначенных прав пользователей.
4. Лист задач. Заметки сотрудника о необходимых в отношении пациента действиях
5. Расписания. Готовые расписания сотрудников, которые хранят полную информацию о типе и времени события и данные, зависящие от типа события (ссылка на пациента, группу, собрание и т.д.)

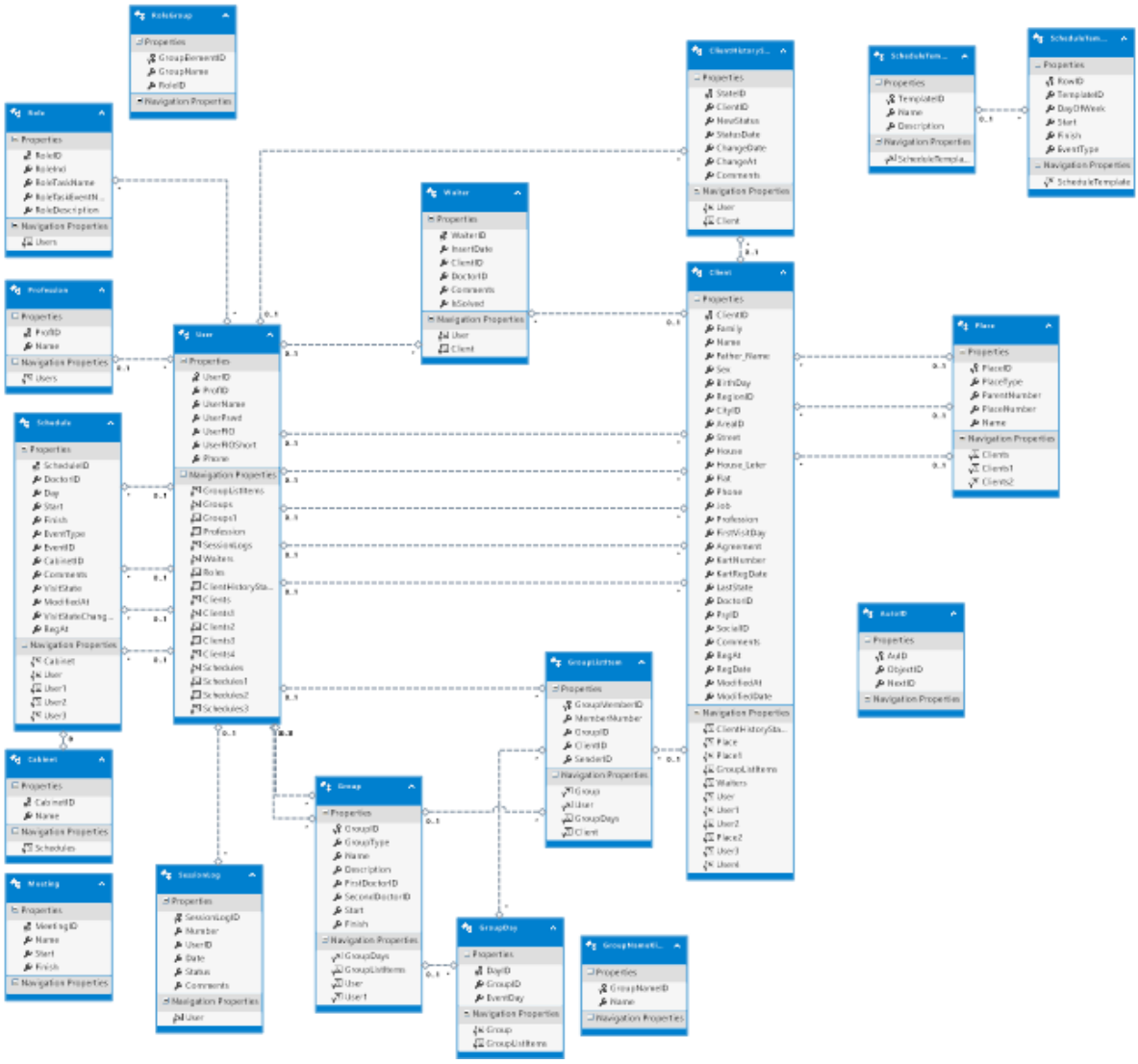


Рис. 2. Структура базы данных разработанной системы

6. Шаблоны расписаний. Шаблон расписания на неделю. Применяются для генерации расписаний из п.5 (но возможна и ручное создание расписаний без применения шаблонов). Шаблоны создаются пользователем вручную или при помощи алгоритма построения расписаний.

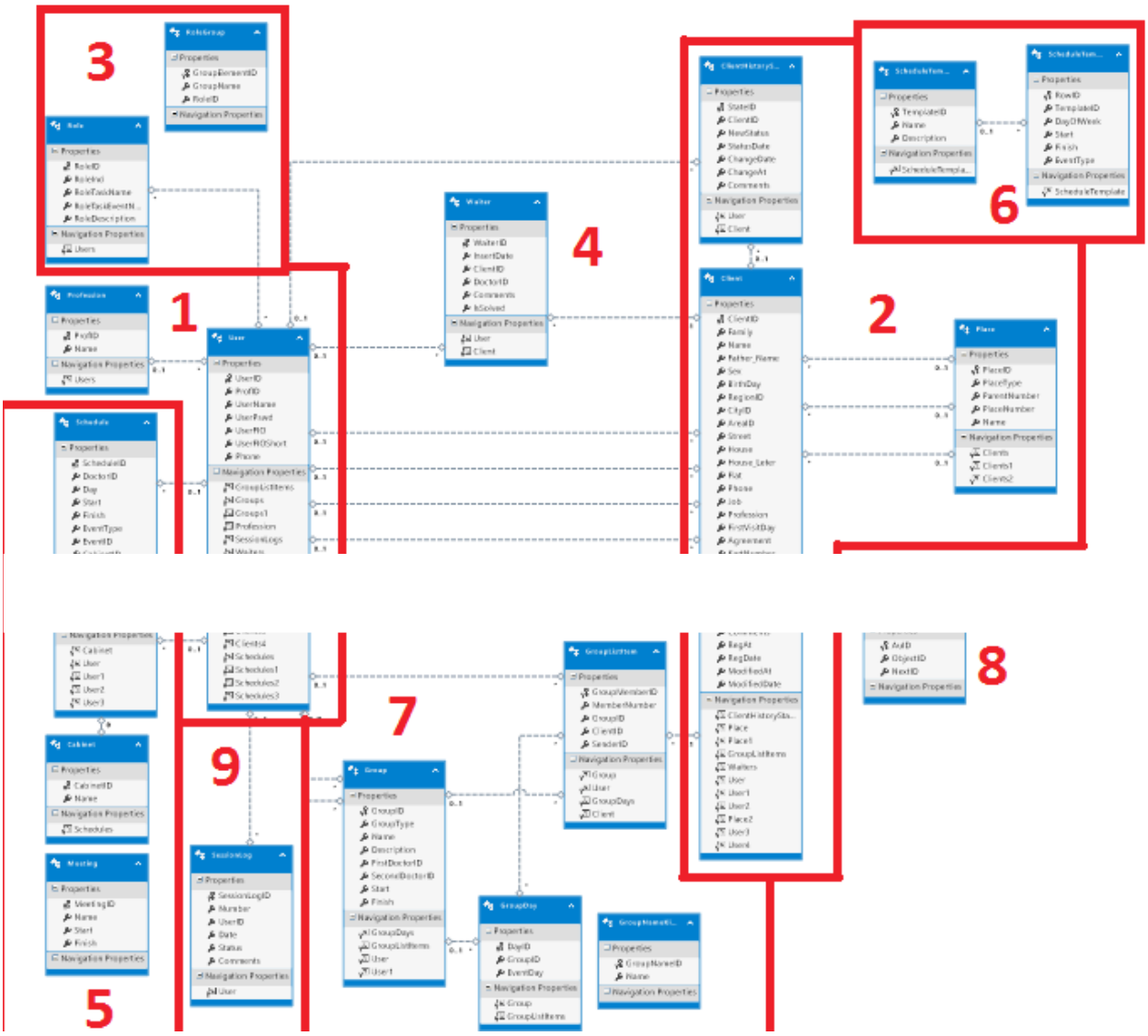
7. Групповые приемы: данные о сотрудниках, ведущих группу, данные о пациентах и посещаемости.

8. Таблица, способная автоматически создавать новый номер для различных структур (номер карты пациента, порядковый номер нового города/района в справочнике адресов, номера для системных ошибок и т.д.) Необходима для контроля над уникальностью этих номеров, чтобы при одновременном многопользовательском доступе не возникало конфликтов.

9. Данные о входе/выходе пользователей и системных ошибках.

Как можно видеть по схеме, основными структурами (наиболее связанные) в данной системе оказались «Сотрудник» и «Пациент», потому

Рис. 3. Структура базы данных разработанной системы (с выделением структурных блоков)



что основная задача данной системы именно в организации их взаимодействия.

В большинстве случаев при создании внешних ключей для таблиц достаточно было задать каскадные методы классическим способом:

```
ON DELETE CASCADE
```

Или

```
ON DELETE SET NULL
```

Но данный подход не работает для таблиц, связанных более, чем одной связью. К примеру, у пациента одновременно три лечащих врача, и при попытке создать три внешних ключа с действием `ON DELETE SET NULL` приводят к ошибке. MS SQL Server просто не позволяет создать такой ключ. Такая же проблема встречается ещё в нескольких местах.

Для решения данной проблемы был применен подход с использованием триггеров типа: `INSTEAD OF DELETE`. Особенностью данного типа

триггеров является то, что он выполняется вместо удаления записи, а не вместе с ней, как в BEFORE (AFTER) DELETE триггерах. При создании этих триггеров было организована обработка всех необходимых действия со связанными записями и только потом удаляется целевая запись. Конечно, этот подход работает гораздо медленнее классического каскадного удаления. Но так как каскадное удаление недоступно, то этот подход гораздо безопаснее, чем оставить ON DELETE NO ACTION и пытаться контролировать удаление в коде программы. Очевидно, что такой способ крайне небезопасен. К тому же удаление данных требуется не часто, а поэтому потери в скорости не существенны.

3.3. Создание базовых функциональных элементов

В приложении активно используется Data Binding (Связывание данных), поэтому создано два базовых класса, реализующих интерфейс *INotifyPropertyChanged*.

```
public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(params string[] propertyNames)
{
     var handler = PropertyChanged;
     if (handler == null) return;
     Helper.DoInMainThread(() =>
     {
         foreach (var property in propertyNames)
        {
            handler(this, new
PropertyChangedEventArgs(property));
         }
     });
}
```

Основная цель данного интерфейса заключается в явном уведомлении клиентов об изменении значения свойства. Один из классов (*NotifyProperty*) является базовым для классов данных, таких как классы модели данных и классы контекстов данных. Второй класс (*NotifyControl*) используется для наследования пользовательскими элементами управления, он наследуется от *UserControl* и расширяет его возможности при помощи данного интерфейса.

Далее был изменен «T4 Text Template» − файл (.tt) . EntityFramework использует данный файл для генерации классов модели из таблиц базы данных. В этом файле добавлено наследование от *NotifyProperty* для того, чтобы изменение свойств сразу же отражалось в интерфейсе программы. Все свойства классов модели приведены от вида:

```
public int Data { get; set; }
```

К виду:

```
partial void OnDataChanged();
private int _Data;
public int Data
{
	get { return _Data; }
	set
	{
		if(Equals(this._ Data, value)) return;
		this._ Data = value;
		this.OnPropertyChanged("Data");
		this.OnDataChanged();
	}
}
```

При вызове *OnPropertyChanged* происходит обновление полей интерфейса, связанных со свойством Data. Метод *OnDataChanged* используется для создания связей в коде, к примеру, при помощи делегатов.

Создание *NotifyProperty, NotifyControl* и изменение вида свойств модели предназначается для полноценного использования MVVM архитектуры.

Далее для организации *View* ↔ *ModelView* взаимодействия создано 3 базовых класса:

- *PageContext* – *ModelView*-класс, который является посредником между интерфейсом и моделью. В нем определены методы загрузки данных из БД, для создания новых записей и сохранения их в БД, для изменения существующих данных.
- *PageControl* – базовый *View*-класс, который расширяет возможности обычного *UserControl*. Он используется для создания элементов управления, которые предназначаются для отображения в главном окне программы
- *ModalPageControl* – предназначается для создания элементов управления, отображаемых в модальном окне.

Для загрузки данных из БД создан класс, представленный ниже в сокращенной форме.

```
public class PsyList<T> : NotifyProperty
{
	public List<T> SourceCollection { get; private set; }
	public ObservableCollection<T> Items { get; private set; }
	public T _selectedItem;
	public T SelectedItem
	{
		get { return _selectedItem; }
		set
		{
			if (Equals(value, _selectedItem)) return;
			_selectedItem = value;
			if (SelectedChanged != null) SelectedChanged();
			OnPropertyChanged("SelectedItem");
		}
	}
	public Func<List<T>> Loading { get; set; }
	public event Action Loaded;
	public event Action SelectedChanged;
	private bool _isBusy;
	public bool IsBusy
	{
		get { return _isBusy; }
		private set
		{
			_isBusy = value;
			OnPropertyChanged("IsBusy");
		}
	}
	public async void LoadAsync()
	{
		if (IsBusy) return;
```

```
                IsBusy = true;
                if (Loading == null) return;
                await Task.Factory.StartNew(() =>
                {
                        SourceCollection = Loading();
                        Helper.DoInMainThread(() =>
                        {
                                SetItems(SourceCollection);
                                IsBusy = false;
                                if (Loaded != null) Loaded();
                        });
                });
                IsBusy = false;
        }
    }
```

Использование двух коллекций данных предназначено для того, чтобы в *SourceCollection* хранились данные, загруженные из БД. А к *Items* устанавливалась привязка интерфейса. Эту коллекцию можно сортировать и фильтровать в зависимости от необходимости. При этом все фильтры можно сбросить и поместить в *Items* полный набор из *SourceCollection* без перезагрузки данных.

Класс для работы с подключением (*ConnectionManager*) при старте приложения просматривает конфигурационный файл на наличие данных подключения. На основе этих данных тестирует подключение и предоставляет метод для получения контекста данных (*CreateContext*). Контекст данных реализует интерфейс *IDisposable*, поэтому построенный контекст можно применять в блоке using:

Всё общение с БД в приложении построено при помощи такого блока.

```
using (var ctx = ConnectionManager.CreateContext())
{
}
```

Класс для работы с сессией (*SessionModel*) дает возможность работы с сессией: открытие и закрытие сессии, проверка данных входа и контроль за правами доступа при помощи *AttachedProperty*. В классе определены для свойства *AccessTaskName* (Имя подзадачи) и

AccessTaskEventName (Имя действия в подзадаче). В БД хранятся права доступа пользователей, каждое правило представляет собой данные вида: id пользователя, имя подзадачи и имя действия (наличие записи в БД означает доступ пользователя к действию, отсутствие – запрет на доступ). У системных кнопок, для которых необходимы права указываются оба эти свойства. На изменения этих свойств срабатывает следующий обработчик:

```
private static void OnTaskChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
{
      var button = d as FrameworkElement;
      if (button == null) return;
      button.IsEnabled = SessionModel.HasAccess(GetAccessTaskName(button),
                                          GetAccessTaskEventName(button));
}
```

HasAccess проверяет наличие записи в БД. Свойство *Button.Visibility* привязано к свойству *Button.IsEnabled*, поэтому установка *IsEnabled = false* сразу скрывает кнопку. Все эти права доступны для настройки администратором.

Класс *Helper* содержит статические функции, необходимые для различных целей в разных частях программ. Один из основных методов - *DoInMainThread* отправляет задание на выполнение в главном потоке. Он необходим в моменты асинхронной работы, когда требуется повлиять на интерфейс.

3.4. Построение архитектуры приложения

Классы подключения и сессии, совместно с PageContex, PageControl и ModalPageControl были использованы для построения базовой архитектуры приложения. Оно организовано так, что для того, чтобы создать новую подзадачу в системе достаточно определить классы, наследующие PageContex и PageControl, а при необходимости использовать модальные окна в подзадаче и от ModalPageControl. Сформированная подзадача добавляется в список

существующих и дальше система сама способна обработать и отобразить эту подзадачу.

Как уже упоминалось ранее, для разработки приложения был выбран паттерн Model-View-ViewModel (Модель-Представление-Модель представления).

При запуске стартового окна выполняется следующий код:

```
public partial class MainWindow : Window
{
	public MainWindow()
	{
		Helper.Init();
		ConnectionManager.InitConnectionManager();
		Constants.Init();
		AppDomain.CurrentDomain.UnhandledException  +=
new UnhandledExceptionEventHandler(MyHandler);
		InitializeComponent();
		SessionModel.CreateSession(
			() => Content = new Watcher(),
			() => Content = new LoginControl()
		);
	}
}
```

Производится инициализация служебных классов и создание сессии, при этом новой сессии передаются два действия на смену содержимого стартового окна. Если пользователь не в сети, то отображается *LoginControl,* если произведен успешный вход, то появляется главное окно приложения *Watcher*.

Watcher – основной класс архитектуры интерфейса. В классе *WatcherContext* задаются подзадачи, существующие в системе. После создания подзадачи необходимо только добавить её в *WatcherContext* и она появится в интерфейсе.

3.5. Логика работы и интерфейс приложения

Для организации главного окна приложения из всего списка требований были выделены подзадачи:

- личный кабинет;

- регистратура;
- сотрудники;
- группы;
- лист задач;
- настройки.

После открытия главное окно выглядит следующим образом:

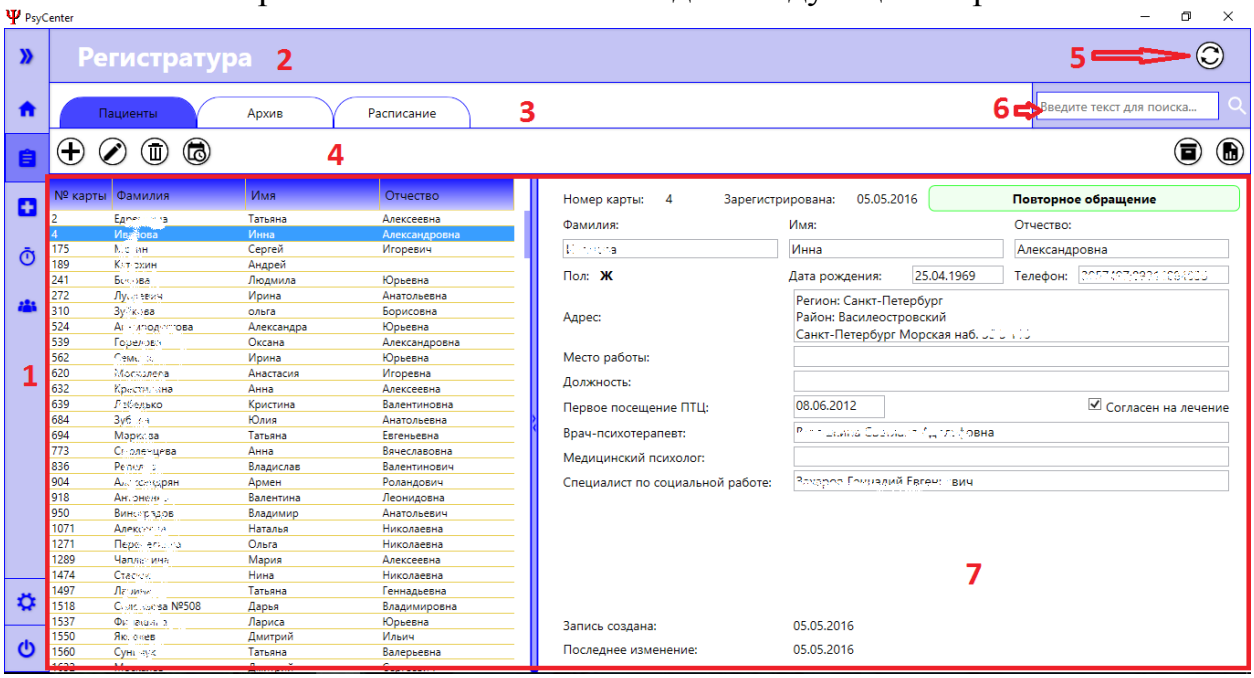


Рис. 4. Главное окно программы (1 – меню для навигации по подзадачам; 2 – заголовок текущей подзадачи; 3 – вкладки текущей подзадачи; 4 – строка с доступными для данной подзадачи действиями; 5 – обновление данных текущей подзадачи; 6 – строка поиска; 7 – окно с основной информацией)

Слева располагается меню для навигации по подзадачам, сверху – название текущей подзадачи и доступные для неё действия. Остальное пространство занято её содержимым.

Некоторые подзадачи разбиты на ещё несколько подзадач и сгруппированы в окне главной подзадачи про помощи вкладок.

К примеру, «Регистратура» содержит в себе вкладки «Пациенты», «Архив», «Расписание».

Основное предназначение главного окна – отображение информации и вызов действий, которые открывают модальные окна, предназначенные для общения с пользователем.

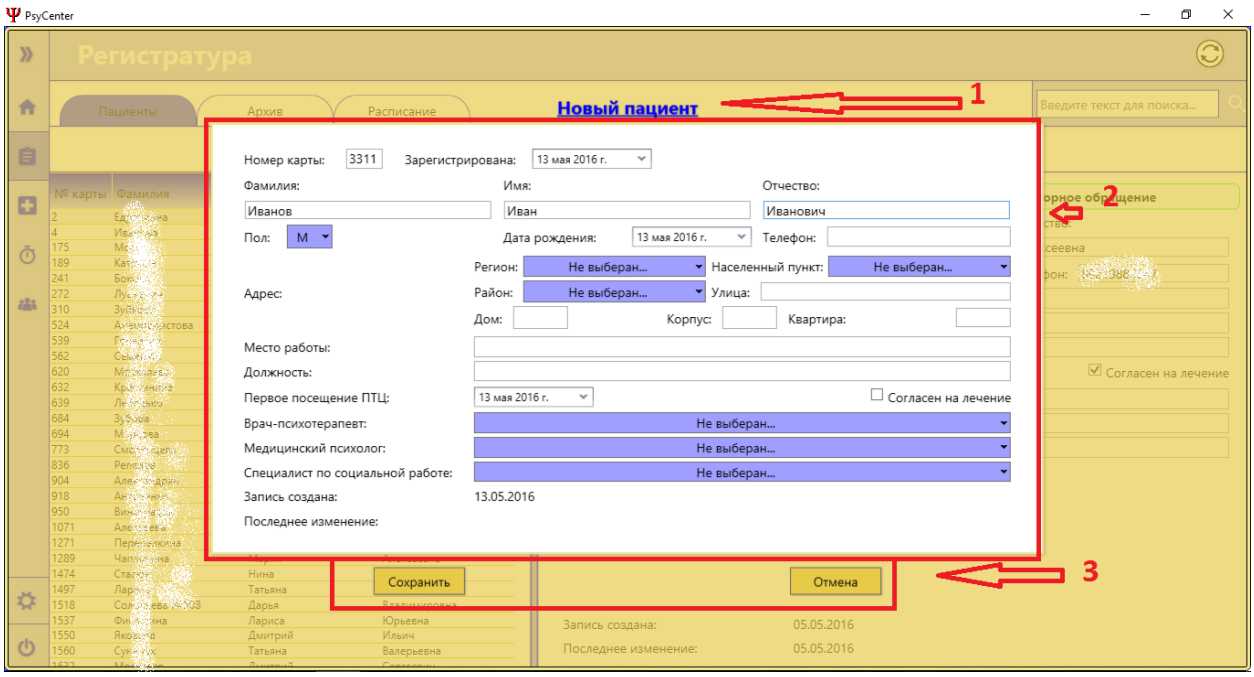


Рис. 5. Модальное окно программы (1 – заголовок модального окна; 2 – содержимое модального окна; 3 – доступные для данного окна действия)

Модальное окно открывается поверх основного и оно предназначено для различного взаимодействия с пользователем (на рисунке можно видеть пример окна ввода и редактирования информации о пациенте).

Модальные окна применяются, как для обычных действий по созданию и изменению элементов различной структуры, так и для отображения некоторых подзадач (к примеру, «Расписание») и для иных действий («Формирование отчетов», «Состояния карты»).

Раздел «Регистратура» состоит из трех подзадач «Пациенты», «Архив» и «Расписание». «Пациенты» и «Архив» выполнены при помощи одного элемента управления, но с разными настройками, что позволяет уменьшить объем кода.

Для карты пациента возможно четыре состояния: принят на учет, лечение окончено, повторное обращение и карта сдана в архив. Для смены этих состояний используется модальное окно:

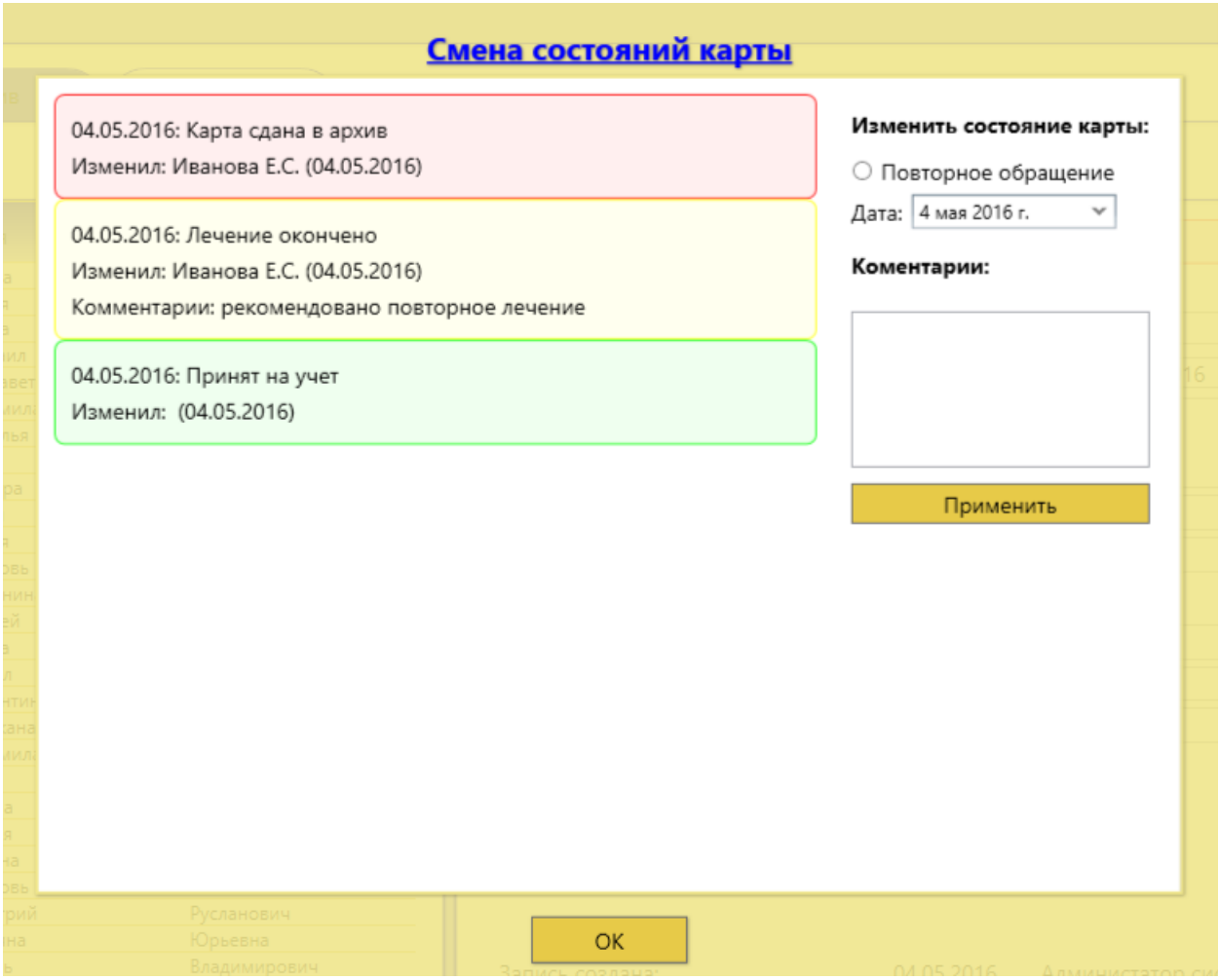


Рис. 6. Смена состояний карты

Слева отображается список изменений состояния карты с информацией о дате изменений; новом статусе; данных о том, кто и когда изменил; комментариями. Справа форма для внесения нового статуса. Отображаются только доступные статусы. В текущем примере доступно только «повторное обращение».

Подзадача «Расписания» используется в трех местах программы (как и в случае с «Пациентами» - один элемент, разные настройки. Эта подзадача отображается в «Личном кабинете» для просмотра личного расписания, в разделе «Регистратура» для записи пациентов на прием, в разделе «Сотрудники» для построения структуры специалистом по расписанию.

Ниже приведен пример отображения данной подзадачи в разделе «Регистратура»:

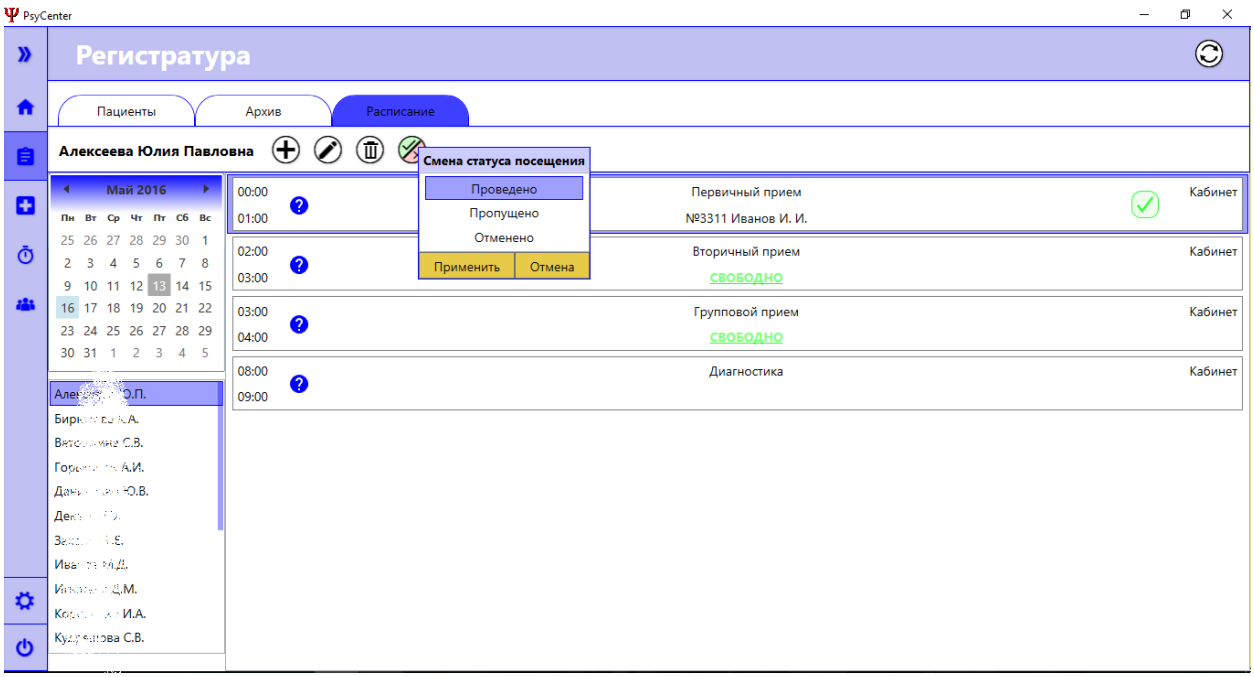


Рис. 7. Расписание

Для записей доступна смена состояния посещения. Выбранный статус отображается специальным знаком справа. При наведении на вопросительный знак в левой части элемента расписания можно увидеть информацию о том, кто и когда менял запись и статус её посещения.

У специалиста по расписанию есть доступ к подзадаче «Шаблоны расписания» (Рис. 8).

Идея шаблона – создать расписание на неделю и в дальнейшем применить его на весь выбранный месяц для одного сотрудника.

Все шаблоны хранятся в базе и могут быть использованы многократно. Шаблон можно создать двумя методами: вручную и автоматизировано, путем заполнения необходимых для алгоритма параметров. Единственное различие в том, что алгоритм создает шаблоны сразу для всех должностей сотрудников. Итоговый вид шаблонов, созданных этими способами, совпадает. Они сохраняются в базе и становятся доступными на редактирование.

После применения шаблона на определенный месяц специалист по расписанию может, не меняя шаблон, откорректировать полученное расписание на месяц.

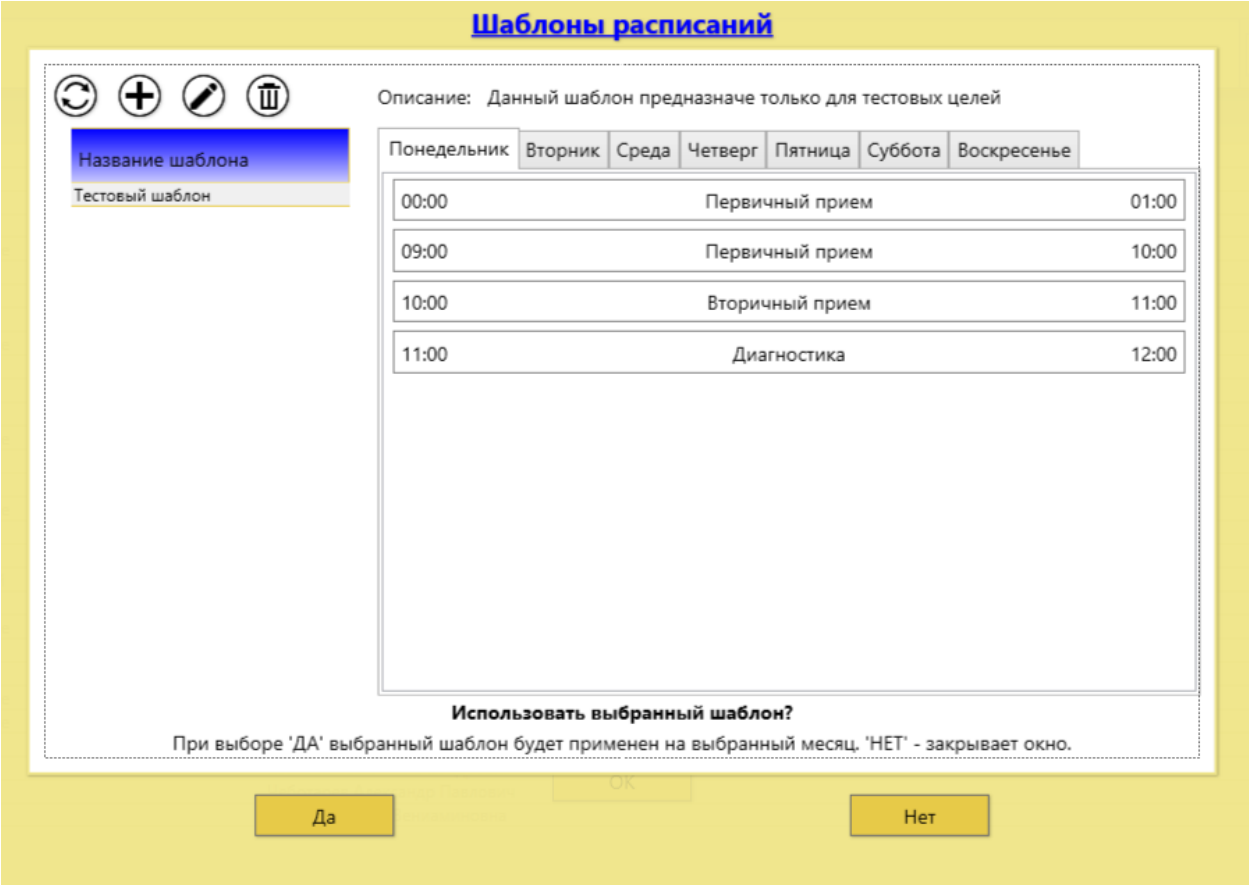


Рис. 8. Шаблоны расписаний

3.6. Создание дополнительной функциональности

Дополнительно были созданы:

- Пополняемые справочники
    - Справочник по регионам, районам и городам для введения адреса пациента.
    - Справочник по типам собраний.
    - Справочник по названиям групповых приемов.
    - Справочник по кабинетам для приема.
- Средства для настройки прав доступа пользователей.
- Окно для смены пароля.
- В некоторых подзадачах используется строка поиска для фильтрации списков данных.

ГЛАВА 4. ТЕСТИРОВАНИЕ СИСТЕМЫ

4.1. Тестирование заявленных функций

Тестирование проходило в два этапа:

- внутреннее тестирование при разработке;
- тестирование персоналом при внедрении.

Большинство ошибок было найдено и исправлено на первом этапе, также были добавлены методы по выявлению программных ошибок и записи данных о них. Сразу же после внедрения первой версии была найдена ошибка в базовой архитектуре приложения и в одном из модулей. Ошибки оказались критичными, но легко исправляемыми. После установки второй версии новых программных ошибок не возникало.

Для тестирования модуля учета пациентов были использованы реальные данные психотерапевтического центра. 3310 записей о картах пациентов в таблице Excel.

Для переноса данных было реализовано две вспомогательные программы:

- первая программа переносила общие данные такие, как ФИО, адрес, дата рождения и т.д.
- вторая программа переносила данные об истории изменений состояния карты.

При использовании этих двух программ было потеряно только 2 из 3310 записей. Они были перенесены вручную. Также были утеряны некоторые комментарии, которые сотрудники устанавливали в случайных столбцах таблицы Excel, не соблюдая заголовки столбцов. Но потеря этих данных оказалась не критичной.

Одна из проблем организации хранения данных в Excel состояла в том, что при поиске по таблице с большим объемом данных, возникали зависания интерфейса. Разработанный программный комплекс оказался лишен этого недостатка. Во время внутреннего тестирования генератором было создано

более пяти тысяч записей, поиск по ним производился моментально. На реальных данных центра – аналогично.

Все остальные заявленные требования также успешно прошли испытания.

4.2. Тестирование модульности

Одной из изначальных идей при построении базовый архитектуры приложения было стремление к модульности. Модульность в виде подключаемых библиотек не рассматривалась, так как данный проект не предполагает частичной продажи, как аналоги, рассмотренные в главе 1. Поэтому достаточно создать возможность к расширению в рамках одной библиотеки классов. В главе 3 описаны базовые классы, за счет которых была реализована эта идея.

Для проверки расширяемости было создано несколько набросков подзадач на основе базовых классов, описанных в параграфе 3.3. Для добавления их в систему достаточно в классе `WatcherContext` добавить всего одну строку:

```
AddPage(typeof(NewControl));
```

Где `NewControl` – это основной класс интерфейса подзадачи (в нем так же содержится информация о контексте для этого элемента управления и модальных окнах), `WatcherContext` умеет анализировать эту информацию и предоставлять её `Watcher` (класс интерфейса основного окна приложения) для отображения.

Тестовые модули успешно отобразились в системе. При этом создание пустого наброска нового модуля требует малого количества действий:

- создать класс-наследник от `PageControl`;
- создать класс-наследник от `PageContext` и привязать его к первому классу;
- добавить строку подключения модуля в `WatcherContext`;
- добавить в БД права на доступ к этому модулю.

4.3. Тестирование вспомогательных систем

В программу были внедрены методы контроля ошибок. Полностью проработаны эти методы оказались только на контроле данных сессии. Большая нехватка этих методов наблюдается в проверке стабильности подключения к серверу. При проблемах соединения чаще всего срабатывал обработчик событий, перехватывающий исключения, которые не отловили все типовые обработчики. Этот обработчик снабжен минимальным набором действий и в большинстве ситуаций стремится закрыть приложение с системной ошибкой, так как он предполагается именно для нетипичных ситуаций, которые не способны обработать методы, созданные для известных типов ошибок.

Данный факт в большей степени касается именно ошибок сети. Эту проблему можно решить, но она не является приоритетной, так как программа предназначена для локальной сети и её стабильность возлагается на системного администратора.

4.4. Тестирования раздела построения расписаний

Специалисты успешно обучились составлять расписания в ручном режиме и при помощи шаблонов. Шаблоны успешно формировались как в ручном режиме, так и при помощи алгоритма автоматизации. Большинство шаблонов использовались многократно.

ВЫВОДЫ

Был проведен анализ существующей системы учета данных, применяемой в организации. В ней были найдены как удачные решения, которые нашли свое применение в системе, так и ошибки, которые были учтены при разработке.

Проведено исследование существующих методов разработки многопользовательских систем с учетом необходимой функциональности. Была выбрана технология, предложенная Microsoft в рамках .Net Framework.

Спроектирована структура базы данных и основных элементов системы.

Изучены и реализованы методы автоматизированного построения расписаний.

Была создана система, содержащая все необходимые для работы центра функции. При этом не возникло ситуации, как с аналогами, когда система имеет неиспользуемые функции.

Персонал был успешно обучен работе с системой. Она была признана комфортной всеми сотрудниками. Основным показателем комфортности являлась простая навигация в системе и малое количество одновременно доступных действий (если возможность быстро прочитать всплывающие к каждому действию подсказки).

Анализ работы центра при внедрении системы показал, что в первое время работа усложнилась, так как персонал поначалу продолжал пользоваться старой системой совместно с новой. Но со временем отмечено ускорение работы персонала, особенно в работе специалиста по составлению расписаний.

ЗАКЛЮЧЕНИЕ

Стоит отметить, что данная система, хоть и уступает крупным коммерческим проектам, но со своими задачами она успешно справляется.

Помимо удобства, одним из важных её преимуществ является расширяемость. В ней легко изменять уже существующие модули и добавлять новые.

В то же время системе требуется доработка в части обработки исключений и подключения к серверу. В перспективе можно также выделить работу с БД в отдельный серверный api. Сейчас за работу с БД отвечают `ViewModel` классы программы.

Управление сервером отдано под ответственность системного администратора для настройки архивации базы данных и контроль за стабильностью её работы.

В итоге получился готовый работоспособный проект с возможностью дополнения.

По соглашению с организацией автором осуществляется поддержка существующих модулей. Обсуждение новых пока отложено, хотя идеи для из реализации существуют.

В данный момент система используется организацией в полном объеме (Приложение 2).

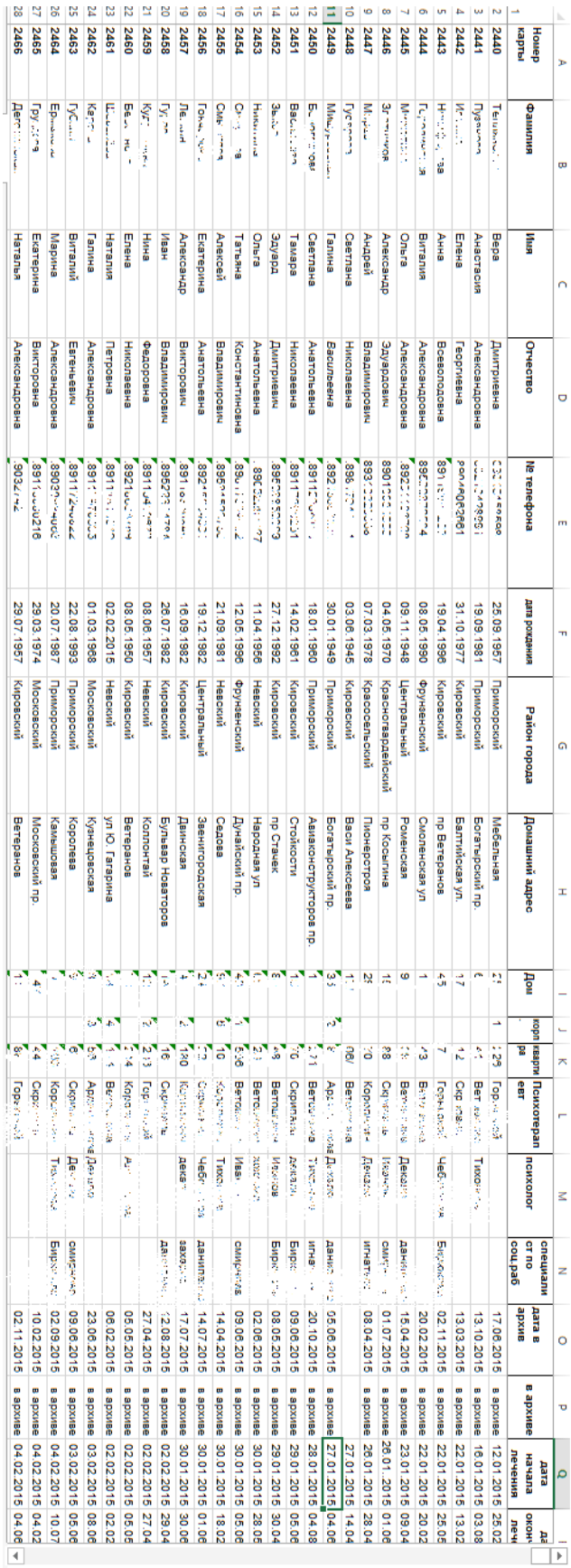
Список литературы

x

1. Конвей Р.В., Максвелл В.Л., Миллер Л.В. Теория расписаний. Наука-е изд. М. 1975.
2. Лазарев А.А., Гафаров Е.Р. Теория расписаний. Задачи и алгоритмы. М.: МГУ, 2011.
3. Brucker P. Scheduling Algorithms. Springer Verlag, 2007. 317-320 pp.
4. Smith J. Patterns - WPF Apps With The Model-View-ViewModel Design Pattern // MSDN Magazine, February 2009.
5. ЗАО «КОСТА». Медицинская информационная система «Авиценна», http://kostasoft.ru/produkty/reshenie-dlya-psihiatrii/
6. ООО «Решение». Медицинская Информационная Система «Ариадна», http://www.reshenie-soft.ru/
7. ООО «САМСОН Групп». Медицинская Информационная Система «Комплекс Программных Средств «Система Автоматизации Медико-Страхового Обслуживания Населения (САМСОН)», http://samson-rus.com/
8. .NET Framework, https://ru.wikipedia.org/wiki/.NET_Framework
9. EntityFramework, https://msdn.microsoft.com/ru-ru/data/ef.aspx
10. Microsoft Office Excel, https://products.office.com/ru-ru/excel
11. MS SQL Server Express, https://www.microsoft.com/ru-ru/server-cloud/products/sql-server-editions/sql-server-express.aspx

x

ПРИЛОЖЕНИЕ 1



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Номер карты	Фамилия	Имя	Отчество	№ телефона	дата рождения	Район города	Домашний адрес	Дом	корпуса	квартира	Психотерапевт	психолог	специалист по соцраб	Дата в архив	в архиве	Дата начала лечения	Дата окончания лечения
2	2440	Т[illegible]	Вера	Дмитриевна	[illegible]	25.09.1967	Приморский	Мебельная	2[illegible]	1	2[illegible]29	Г[illegible]		17.06.2015	в архиве	12.01.2015	25.02	
3	2441	П[illegible]	Анастасия	Александровна	[illegible]	19.09.1981	Приморский	Богатырский пр.	6		[illegible]	Вет[illegible]	Тихонов[illegible]		13.10.2015	в архиве	16.01.2015	03.08
4	2442	И[illegible]	Елена	Георгиевна	8[illegible]2861	31.10.1977	Кировский	Балтийская ул.	17		12	Скр[illegible]			13.03.2015	в архиве	22.01.2015	13.02
5	2443	Н[illegible]	Анна	Всеволодовна	891[illegible]	19.04.1996	Кировский	пр Ветеранов	45		7	Горь[illegible]	Чеб[illegible]	Бир[illegible]	02.11.2015	в архиве	22.01.2015	25.05
6	2444	Г[illegible]	Виталия	Александровна	89[illegible]74	08.06.1990	Фрунзенский	Смоленская ул	1		73	Бат[illegible]			20.02.2015	в архиве	22.01.2015	20.02
7	2445	М[illegible]	Ольга	Александровна	892[illegible]	09.11.1948	Центральный	Роменовая	9		[illegible]	Вет[illegible]	Дем[illegible]	дан[illegible]	15.04.2015	в архиве	23.01.2015	09.04
8	2446	Зг[illegible]	Александр	Эдуардович	8901[illegible]	04.05.1970	Красногвардейский	пр Косыгина	15		88	Скр[illegible]	Пет[illegible]	смир[illegible]	01.07.2015	в архиве	26.01.2015	01.06
9	2447	М[illegible]	Андрей	Владимирович	893[illegible]	07.03.1978	Красносельский	Пионерстроя	25		70	Кор[illegible]	Дем[illegible]	игнат[illegible]	08.04.2015	в архиве	28.01.2015	28.04
10	2448	Г[illegible]	Светлана	Николаевна	898[illegible]	03.06.1945	Кировский	Васи Алексеева	1[illegible]		66/	Вет[illegible]					27.01.2015	14.04
11	2449	М[illegible]	Галина	Васильевна	892[illegible]	30.01.1949	Приморский	Богатырский пр.	35	3	8	Арх[illegible]	Дав[illegible]	давн[illegible]	05.06.2015	в архиве	27.01.2015	04.08
12	2450	Б[illegible]	Светлана	Анатольевна	8911[illegible]	18.01.1960	Приморский	Авиаконструкторов пр.	1		71	Вет[illegible]	Тих[illegible]	игнат[illegible]	20.10.2015	в архиве	28.01.2015	04.08
13	2451	Ва[illegible]	Тамара	Николаевна	8911[illegible]	14.02.1961	Кировский	Стойкости	1[illegible]		70	Скр[illegible]	Дем[illegible]	Бир[illegible]	09.06.2015	в архиве	29.01.2015	05.06
14	2452	За[illegible]	Эдуард	Дмитриевич	89[illegible]3	27.12.1992	Кировский	пр Стачек	3		8	Вет[illegible]	Ива[illegible]	Бир[illegible]	08.06.2015	в архиве	29.01.2015	30.04
15	2453	Н[illegible]	Ольга	Анатольевна	89[illegible]07	11.04.1956	Невский	Народная ул	4[illegible]		21	Вет[illegible]			02.06.2015	в архиве	30.01.2015	28.05
16	2454	С[illegible]	Татьяна	Константиновна	89[illegible]2	12.05.1996	Фрунзенский	Дунайский пр.	45		5/6	Вет[illegible]	Ива[illegible]	смир[illegible]	09.06.2015	в архиве	30.01.2015	05.06
17	2455	См[illegible]	Алексей	Владимирович	895[illegible]	21.09.1981	Невский	Седова	5[illegible]	3	10	К[illegible]	Тих[illegible]		14.04.2015	в архиве	30.01.2015	18.02
18	2456	Го[illegible]	Екатерина	Анатольевна	892[illegible]	19.12.1982	Центральный	Звенигородская	24		73	Скр[illegible]	Чеб[illegible]	давн[illegible]	14.07.2015	в архиве	30.01.2015	01.06
19	2457	Ле[illegible]	Александр	Викторович	891[illegible]	16.09.1982	Кировский	Двинская	4	2	120	К[illegible]	дека[illegible]	захар[illegible]	17.07.2015	в архиве	30.01.2015	30.06
20	2458	Гу[illegible]	Иван	Владимирович	895[illegible]	26.07.1982	Кировский	Бульвар Новаторов	1[illegible]		16	Скр[illegible]		дав[illegible]	12.08.2015	в архиве	02.02.2015	29.04
21	2459	Ку[illegible]	Нина	Федоровна	8911[illegible]	08.06.1957	Невский	Коллонтай	1[illegible]	2	213	Гор[illegible]			27.04.2015	в архиве	02.02.2015	27.04
22	2460	Ев[illegible]	Елена	Николаевна	8921[illegible]	08.06.1950	Кировский	Ветеранов	1[illegible]	4	14	Кор[illegible]	Ан[illegible]		05.06.2015	в архиве	02.02.2015	05.05
23	2461	Ш[illegible]	Наталия	Петровна	8911[illegible]	02.02.2015	Невский	ул Ю. Гагарина	[illegible]	4	14	Вет[illegible]			06.02.2015	в архиве	02.02.2015	02.02
24	2462	Ка[illegible]	Галина	Александровна	8911[illegible]	01.03.1968	Московский	Кузнецовская	3[illegible]	3	5/3	Ар[illegible]			23.06.2015	в архиве	03.02.2015	08.06
25	2463	Гу[illegible]	Виталий	Евгеньевич	8911[illegible]	22.08.1993	Приморский	Королева	[illegible]		8	Скр[illegible]	Дем[illegible]	смир[illegible]	09.06.2015	в архиве	03.02.2015	05.06
26	2464	Ер[illegible]	Марина	Александровна	8902[illegible]	20.07.1987	Приморский	Камышовая	[illegible]		[illegible]	Кор[illegible]	Тих[illegible]	Бир[illegible]	02.09.2015	в архиве	04.02.2015	10.07
27	2465	Гр[illegible]	Екатерина	Викторовна	8911[illegible]216	29.03.1974	Московский	Московский пр.	4[illegible]		44	Скр[illegible]			10.02.2015	в архиве	04.02.2015	04.02
28	2466	Де[illegible]	Наталья	Александровна	902[illegible]4	29.07.1957	Кировский	Ветеранов	1[illegible]		8/	Гор[illegible]			02.11.2015	в архиве	04.02.2015	04.06

Часть таблицы с данными пациентов

ПРИЛОЖЕНИЕ 2

ПРАВИТЕЛЬСТВО САНКТ-ПЕТЕРБУРГА
КОМИТЕТ ПО ЗДРАВООХРАНЕНИЮ
Санкт-Петербургское государственное
бюджетное учреждение здравоохранения
«Городской Психоневрологический
диспансер № 7 (со стационаром) »
Психотерапевтический центр

Санкт-Петербург, 190008
ул.Канонерская, д.12,
Тел./факс (812) 714-34-14,
ОКПО 34 336 510 ОКОГУ 23340
ОГРН 1027810359565
ИНН/КПП 78 09 016 374 / 782 601 001
ptcentr@bk.ru
28 АПР 2016 **2016г. №** 656

Акт внедрения
программного обеспечения
Система учета «PsyCenter»

Настоящий Акт свидетельствует, что программное обеспечение Система учета пациентов и работы сотрудников «PsyCenter» (далее Система), разработанная Сахаровым Алексеем Александровичем (далее Разработчик) в рамках работы над дипломной работой «Разработка многопользовательской системы учета с возможностью автоматизированного построения расписаний», внедрена в «Городской психоневрологический диспансер № 7 (со стационаром)» подразделение Психотерапевтический центр (далее Организация).

Процесс внедрения системы учета проводился поэтапно с мая 2015 года по март 2016 года. Проведено тестирование системы, выполнено обучение персонала. В настоящий момент сервер Системы установлен в Организации, к нему подключены и полноценно функционируют 5 ПК (рабочих мест).

Заявленные характеристики системы предполагали наличие следующих основных возможностей:

- Многопользовательский доступ к Системе.
- Контроль доступа сотрудников к Системе и данным, хранящимся в ней.
- Учет пациентов Организации.
- Снятие пациентов с учета (отправка карты в архив) и возобновление лечения (возвращение карты из архива), доступ к истории этих изменений.
- Формирование групп пациентов для проведения групповых приемов с возможностью фиксации посещаемости.

- Ведение журнала задач для учета заметок о действиях сотрудника, которые ему требуется произвести по отношению к тому или иному пациенту.
- Административный раздел для учета пользователей и их прав доступа.
- Ведение раздела расписаний работы сотрудников, функции данного раздела зависят от должности текущего пользователя системы:
    - специалисту по расписаниям доступны инструменты автоматизированного и ручного построения расписаний;
    - специалисту регистратуры – запись пациентов на прием в уже созданное расписание;
    - всем сотрудникам – просмотр личного расписания.

  Элементами расписания выступают:

    - индивидуальные приемы пациентов;
    - прием группы пациентов, сформированной в Системе;
    - иные события.
- Автоматизированное построение расписания (с возможностью дальнейшей коррекции) с учетом заданных ограничений.
- Формирование отчетов по работе сотрудников и приему пациентов на основании расписаний, хранимых в Системе.

В ходе эксплуатации программы подтверждено, что она обладает всеми заявленными возможностями и позволяет в полном объеме вести учет пациентов Организации.

И.о. главного врача Колчев А.И.

МП