МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«МУРМАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЁТ**

**ПО** **ПРЕДДИПЛОМНОЙ ПРАКТИКЕ**

Место прохождения практики – кафедра информационных технологий

(указать место прохождения практики в соответствии с приказом)

Сроки практики - с \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. по \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

(указать сроки прохождения практики в соответствии с приказом)

Объем практики зет (час) – \_\_\_ (\_\_\_\_\_)

**Выполнил:** обучающийся 5 курса группы ИВТб19з-1

направления подготовки: 09.03.01 Информатика и вычислительная техника

направленности (профиля): «Программное обеспечение вычислительной техники и автоматизированных систем» .

форма обучения: заочная

Мансуров Алишер Набижонович

(Фамилия, Имя, Отчество обучающегося)

**Руководитель практики от МГТУ**

(Фамилия, Имя, Отчество, должность, ученая степень)

Мурманск, 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc167222285)

[1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc167222286)

[1.1. Исследование и анализ ведения медицинской карты ребенка 6](#_Toc167222287)

[1.2. Описание бизнес-процесса 6](#_Toc167222288)

[1.3. Описание входной и результатной информации 7](#_Toc167222289)

[1.4. Выявленные проблемы в предметной области 9](#_Toc167222290)

[1.5. Алгоритмические зависимости 11](#_Toc167222291)

[1.6. Бизнес-правила 11](#_Toc167222292)

[1.7. Требования Заказчика к программному средству 12](#_Toc167222293)

[1.7.1. Требования к группам пользователей 12](#_Toc167222294)

[1.7.2. Функциональные требования 12](#_Toc167222295)

[1.7.3. Требования к данным 13](#_Toc167222296)

[1.7.4. Требования к окружению 13](#_Toc167222297)

[1.8. Обзор аналогов 13](#_Toc167222298)

[2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 15](#_Toc167222299)

[2.1. Описание архитектуры разрабатываемого программного средства 15](#_Toc167222300)

[2.2. Описание инструментария 16](#_Toc167222301)

[2.2.1. Физическое проектирование баз данных 18](#_Toc167222302)

[2.2.1.1. Создание базы данных 18](#_Toc167222303)

[2.2.1.2. Поддержка ограничений целостности 19](#_Toc167222304)

[2.3. Реализация API 21](#_Toc167222305)

[2.4. Реализация функций 22](#_Toc167222306)

[2.5. Реализация интерфейса 26](#_Toc167222307)

[3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 28](#_Toc167222308)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc167222309)

[*Приложение 1* 37](#_Toc167222310)

[*Приложение 2* 43](#_Toc167222311)

[*Приложение 3* 46](#_Toc167222312)

[*Приложение 4* 47](#_Toc167222313)

[*Приложение 5* 48](#_Toc167222314)

[*Приложение 6* 49](#_Toc167222315)

[*Приложение 7* 51](#_Toc167222316)

[*Приложение 8* 55](#_Toc167222317)

[*Приложение 9* 65](#_Toc167222318)

[*Приложение 10* 66](#_Toc167222319)

[*Приложение 11* 71](#_Toc167222320)

[*Приложение 12* 74](#_Toc167222321)

[*Приложение 13* 75](#_Toc167222322)

# ВВЕДЕНИЕ

В соответствии с приказом Минздрава РФ от 05.11.2013 N 822н [1], за каждой образовательной организацией на территории Российской Федерации, работающей с несовершеннолетними детьми, должен быть закреплен штатный медицинский работник. Это обусловлено тем, что медицинские работники, в числе которых врачи и медицинские сестры, обязуются правильно обследовать детей и принимать решения касательно профилактики и лечения различных заболеваний.

Основным документом, содержащим в себе все необходимые медицинские данные о ребенке, является медицинская карта ребенка для образовательных учреждений дошкольного, начального общего, основного общего, среднего (полного) общего образования, учреждений начального и среднего профессионального образования, детских домов и школ-интернатов (форма 026/у–2000)[[1]](#footnote-1), утвержденной приказом Минздрава РФ от 03.07.2000  
N 241 [2].

Медицинская карта заводится на каждого ребенка при поступлении в образовательное учреждение. Для ее заполнения требуется получение информации о состоянии ребенка, а также точного и полного описания травм, болезней, их осложнений, прививок и реакций после прививки, результатов анализов и исследований, данных об истории болезни родственников первого и второго поколения.

Также на основании данных в медкартах составляются различные отчеты.

На данный момент, каждая медкарта заполняется медсестрой вручную: обычной шариковой ручкой, в пустографных бумажных медицинских картах ребенка установленного образца.

Очевидно, что такой подход затрудняет процесс работы с информацией, в частности: поиск, ведение статистики, составление отчетов на основе обобщенных данных. В результате чего излишне расходуется рабочее время, снижается качество и достоверность предоставляемых данных. Разработанное программное средство поможет полностью автоматизировать указанные процессы.

**Практическая значимость**

Разрабатываемое программное средство может быть использовано для повышения эффективности работы медицинского работника дошкольного образовательного учреждения в части ведения медицинских карт детей.

**Новизна**

В ходе исследования предметной области, никаких программных средств, предназначенных для автоматизации ведения медицинских карт детей по установленной форме, в открытых источниках найдено не было. В основном имеются только информационные системы для врачей в поликлиниках и больницах, но их работа направлена на другие задачи.

Таким образом: разрабатываемое программное средство можно считать уникальным.

# ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## Исследование и анализ ведения медицинской карты ребенка

Как уже было отмечено ранее: медицинская карта в обязательном порядке заводится на каждого ребенка при его поступлении в образовательное учреждение.

Согласно приказу Минздрава РФ [2] указанный документ состоит из 11 разделов:

Раздел 1. Общие сведения о ребенке;

Раздел 2. Анамнестические сведения;

Раздел 3. Сведения о диспансерном наблюдении;

Раздел 4. Обязательные лечебно-профилактические мероприятия;

Раздел 5. Иммунопрофилактические мероприятия;

Раздел 6. Данные плановых профилактических медицинских осмотров;

Раздел 7. Результаты врачебной профессиональной консультации;

Раздел 8. Рекомендации по занятиям спортом, бальными или спортивными танцами;

Раздел 9. Сведения о подготовке юношей к военной службе;

Раздел 10. Данные текущего медицинского наблюдения;

Раздел 11. «Скрининг-программа» (дошкольный этап, этап основного общего, среднего (полного) общего образования).

Полная структура документа приведена в прил. 1.

Правила заполнения каждого раздела[[2]](#footnote-2) приведены в прил. 2 приказа [2].

## Описание бизнес-процесса

В ходе изучения предметной области, были выделен один бизнес-процесс – ведение медицинской карты ребенка. Данный процесс включает в себя следующие подпроцессы:

1. При поступлении ребенка в дошкольное образовательное учреждение на него заводится новая медицинская карта, если ранее она не заводилась. Все данные в нее вносятся на основании справки по форме 026/у из поликлиники, куда прикреплен ребенок. Данную справку предоставляют родители (опекуны). Если ребенок перевелся из другого детского сада и на него уже есть медицинская карта, то родители (опекуны) предоставляют ее.
2. Общий процесс заполнения медицинской карты включает в себя последовательное внесение данных во все разделы, согласно Правил.
3. При прохождении ребенком медицинских осмотров, вакцинации, сдаче анализов, либо иных мероприятий, которые необходимо учитывать в медицинской карте (применение антибиотиков, результаты УЗИ или Рентгенографии и т. д.) – все данные вносятся в соответствующие разделы медсестрой лично. Документами-основаниями для внесения данных могут служить различные справки от врачей.
4. При исключении ребенка из данного дошкольного образовательного учреждения – медкарта, подписанная медсестрой, выдается на руки одному из родителей (опекуну).
5. В январе каждого года составляется годовой отчет по состоянию на 31 декабря предыдущего года (структура документа приведена в прил. 2).
6. На основании данных в медкартах формируются данные по профилактическим прививкам (сколько вакцинировано, на каком этапе) за указанный период по группам (структура документа приведена в прил. 3).
7. На основании данных в медкартах формируются данные по дегельминтизации (проверено детей, из них: отрицательно, положительно) за указанный период по группам (структура документа приведена в прил. 4).
8. На основании данных в медкартах формируются туберкулинодиагностика (всего детей, сколько сделано Р-манту, из них: отрицательно, сомнительно, положительно) за указанный период по группам (структура документа приведена в прил. 5).
9. Все отчеты (включая годовой отчет) сдаются в поликлинику врачу (старшей медсестре), за которой закреплено данное дошкольное образовательное учреждение.

## Описание входной и результатной информации

На рис. 1 приведена информационная модель связи входных и результирующих документов.

Существует множество различных справок от врача: цитологический анализ, флюорография, профилактические мероприятия, скрининг, анализ крови и т. д. Данные по каждому виду справки вносятся в соответствующий раздел. На рис. 2 приведен образец справки после болезни.

Описание каждого вида справок в контексте данного исследования не требуется, поэтому сразу рассмотрим описание структуры самой медкарты (прил. 1). В ней рассмотрены только те атрибуты, которые относятся к учреждениям дошкольного образования.

Медкарта

Справка от врача

Отчет

Справка форма 026/у из поликлиники

Рис. 1. Информационная модель связи входных

и результирующих документов



Рис. 2. Образец справки после болезни

На каждого ребенка в дошкольном образовательном учреждении есть своя медкарта. В каждом учреждении в среднем по 6 групп, в каждой группе в среднем по 25 человек. Со всех групп в одном учреждении набирается около 10 различных справок от врача в неделю. Профилактический медосмотр проходят около 600 детей в год. Также каждый год производится набор около 100 детей, на которых заводятся новые медкарты. Около 100 детей ежегодно исключаются из детских садов и забирают с собой медкарты. В среднем за год между дошкольными образовательными учреждениями переводятся около 10 детей. Часто болеющих детей около 10 человек.

## Выявленные проблемы в предметной области

На текущий момент ведение медицинской карты ребенка осуществляется вручную в бумажном виде. Как было отмечено ранее: на каждого ребенка заводится своя карта, установленного образца, которая хранится у медицинского работника в данном учреждении.

Одной из проблем такого подхода является отсутствие оперативных и достоверных данных о состоянии здоровья и проведенных профилактических мероприятиях, как для отдельных детей, так и для групп и всей организации в целом. Это связано с тем фактом, что на должности медицинского работника дошкольного образовательного учреждения обычно работают студенты старших курсов высших образовательных учреждений медицинского образования, получившие сертификат медсестры. Обычно на пол ставки и не более года. Таким образом, медкарты:

* теряются;
* в них выцветает текст из-за использования гелиевых ручек (и т. п.);
* содержат данные, написанные неразборчиво;
* содержат не полную информацию, так как медсестра может не успевать сразу вносить данные из-за скопившегося большого количества «письменной» работы, а затем и вовсе забыть внести эти данные;
* не обеспечивают должную наглядность проведенных профилактических мероприятий (в том числе проставленных прививок).

В виду наличия указанных проблем может в значительной степени снизиться качество составляемых отчетов, таких как:

* отчеты по прививкам (например, на основании отчета по прививке от полиомиелита, детей временно распределяют по группам, в зависимости от текущей ревакцинации, так как используется «живая» прививка и есть риск заражения непривитых детей);
* отчет по дегельминтизации (может способствовать определению соблюдения санитарных норм в доме у ребенка и в образовательном учреждении в том числе);
* отчет по реакции Манту (может способствовать определению рисков заражения туберкулезом).

Также в конце каждого календарного года составляется годовой отчет, который представляет собой набор статистических данных по установленной форме (структура документа приведена в прил. 2). При этом медсестра собирает и считает данные по каждому пункту вручную, что непременно способствует наличию математических ошибок. Помимо этого, медсестра также может допустить неточность в категории, к которой относится ребенок. И множество других человеческих факторов, например: медсестре может стать, попросту, лень продолжать данный трудоемкий процесс, что приведет к заполнению указанного отчета недостоверными данными.

В связи с необходимостью решения выявленных проблем, разрабатываемое программное средство должно обеспечивать:

* единство и достоверность ведения данных в медкартах;
* оперативность получения первичной, обобщенной, аналитической и отчетной информации из медкарт.

Для этого надлежит создать единую систему учета и заполнения медкарт и обеспечить ее ведение, что включает в себя:

* создание медкарты;
* учет общих сведений о ребенке;
* учет анамнестических сведений;
* учет сведений о диспансерном наблюдении;
* учет обязательных лечебно-профилактические мероприятий;
* учет иммунопрофилактических мероприятий;
* учет данных плановых профилактических медицинских осмотров;
* учет результатов врачебной профессиональной консультации;
* учет рекомендаций по занятиям спортом, бальными или спортивными танцами;
* учет данных текущего медицинского наблюдения;
* учет базовой скрининг-программы (дошкольный этап);
* составление отчетов: по прививкам, по дегельминтизации, по реакции Манту (на каждую группу в дошкольном образовательном учреждении за указанный период, в соответствии со структурами документов, приведенными в прил. 3-5);
* составление годового отчета по организации (структура документа приведена в прил. 2).

## Алгоритмические зависимости

Одной из проблем, выявленных в предметной области, является заполнение статистических данных, представленных в виде годового отчета (прил. 2), вручную. Следствием этого могут явиться некорректные значения некоторых показателей, так как все данные в годовом отчете являются полностью расчетными, исходя из данных в медкартах (прил. 1).

В ходе поиска возможности автоматизации заполнения полей данного отчета были выявлены некоторые алгоритмические зависимости. Так, некоторые поля в годовом отчете содержат такие статистические показатели, как: «процент», «на 1 ребенка», «на 1000 детей».

Порядок расчетов данных показателей осуществляется по следующим формулам:

Ребенок считается частоболеющим, если в течение года им было перенесено определенное количество болезней (табл. 1)

*Таблица 1*

**Порядок определения частоболеющих детей**

| **Возраст** | **Количество болезней за год** |
| --- | --- |
| 1 – 3 года | 6 и более |
| 3 – 5 лет | 5 и более |
| более 5 лет | 4 и более |

## Бизнес-правила

Исходя из заданной предметной области, можно выделить следующие бизнес-правила:

* значения реквизитов документов должны быть из области допустимых значений;
* каждый ребенок может числиться только в одном дошкольном образовательном учреждении;
* каждый ребенок состоит в одной группе;
* каждая медкарта уникальна;
* на одного ребенка заводится только одна медкарта;
* после исключения из дошкольного образовательного учреждения медкарта выдается одному из родителей (опекуну);
* одна справка выдается только на одного ребенка;
* у одного ребенка может быть несколько справок от разных врачей;
* годовой отчет делается раз в год в январе по состоянию на 31 декабря предыдущего года;
* врач (старшая медсестра), за которой закреплено данное дошкольное образовательное учреждение имеет доступ ко всем отчетам и медкартам данного учреждения.

## Требования Заказчика к программному средству

Рассмотрим требования, выдвинутые Заказчиком к программному средству.

## Требования к группам пользователей

Разрабатываемое программное средство будет использоваться только медицинским работником конкретного дошкольного образовательного учреждения, поэтому выделять особые группы пользователей не требуется. Однако имеет место ограничить несанкционированный доступ к данным с помощью пароля.

## Функциональные требования

Должны быть реализованы следующие функции:

* редактирования данных Пользователя системы (ФИО, пароль)
* создания новой медкарты;
* удаления медкарты;
* импорта медкарты;
* экспорта медкарты;
* вывода на печать медкарты;
* внесения данных в медкарту (как новых, так и изменения существующих);
* формирования годового отчета (структура документа приведена в прил. 2);
* формирования отчета по профилактическим прививкам за указанный период по группам (структура документа приведена в прил. 3);
* формирования отчета по дегельминтизации за указанный период по группам (структура документа приведена в прил. 4);
* формирования туберкулинодиагностики за указанный период по группам (структура документа приведена в прил. 5).

## Требования к данным

В качестве входных данных Пользователи будут вводить с клавиатуры информацию по каждому из пунктов медкарты, а так же импортировать данные из файла.

Выходными данными будут являться сформированные медкарты, файлы экспорта, а также следующие отчеты:

* годовой отчет;
* отчет по профилактическим прививкам за указанный период (по возрасту);
* отчет по профилактическим прививкам за указанный период (по группам);
* отчет по дегельминтизации за указанный период (по группам);
* туберкулинодиагностика за указанный период (по возрасту);
* туберкулинодиагностика за указанный период (по группам).

## Требования к окружению

Необходимо, чтобы системные требования разрабатываемого программного средства были совместимы с заданными:

* Частота процессора 1600 МГц
* ОЗУ – 2048 Мб
* 2 Гб свободного места на жёстком диске
* Разрешение экрана 1024x768 точек
* ОС Windows 7 или AstraLinux 1.6

## Обзор аналогов

Как уже было отмечено ранее: в ходе рассмотрения наличия существующих систем, подходящих для реализации выдвинутых Заказчиком требований, аналогов в открытом доступе (как платных, так и бесплатных) выявлено не было.

# ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## Описание архитектуры разрабатываемого программного средства

Архитектуру разрабатываемого программного средства определяют ряд требований и ограничений, выявленных на данный момент:

* исходя из требований к группам пользователей (п 1.7.1) видно, что особых групп Пользователей не требуется, разделение доступа также не требуется (только защита от несанкционированного доступа путем установки пароля). Соответственно Пользователь в программе будет только один;
* исходя из функциональных требований (п. 1.7.2) видно, что требуется выполнять в основном задачи по добавлению, чтению, обновлению и удалению различных данных (Далее – «CRUD-задачи»);
* исходя из требований к данным (п. 1.7.3) видно, что необходимо хранить только текстовые данные;
* технические характеристики оборудования, на котором будет использоваться данное программное средство указаны в п. 1.7.4;

Вывод: необходимо реализовать архитектуру:

* с наименьшими требованиями к сопровождению и инфраструктуре;
* позволяющую создать кроссплатформенное программное средство;
* без особой необходимости распределения нагрузки на систему.

При этом, также имеет смысл заложить в данное программное средство потенциал, как для вертикального (расширение функциональных возможностей, за счет увеличения ресурсов), так и горизонтального (объединение нескольких детских садов в единую систему) масштабирования.

Таким образом, решено использовать web-архитектуру в виде модульного монолита с одной базой данных. Кроссплатформенность будет обеспечиваться за счет адаптивного web-интерфейса и публичного API.

На данном этапе жизненного цикла разрабатываемого программного средства, предполагается разработка однопользовательской десктопной версии, в виде Progressive Web App (Далее – «PWA») с локальной базой данных SQLite.

Использование базы данных в данном программном средстве обусловлено тем, что она является хорошим инструментом для решения CRUD-задач.

## Описание инструментария

Выбор инструментов для разработки программного средства основан на анализе архитектурных требований программного средства, его целей и особенностей. Данные ограничения перекладываются и на выбор основного языка программирования. В данном случае этот выбор был сделан в пользу Python 3.12, благодаря его простоте и читаемости кода, большому и активному сообществу разработчиков, которое поддерживает множество библиотек, фреймворков и инструментов для разработки, а также подкреплен использованием FastAPI, который является одним из наиболее эффективных фреймворков для разработки веб-приложений на Python.

Таким образом в основной инструментарий разработки состоит из:

* **FastAPI:**

FastAPI был выбран в качестве основного фреймворка для разработки по ряду причин. Прежде всего, он предлагает высокую производительность и эффективность благодаря использованию асинхронного подхода. Это особенно важно для веб-приложений, так как обеспечивает быстрый отклик и масштабируемость. Кроме того, FastAPI интегрируется с автоматической документацией Swagger, что облегчает процесс разработки и поддержки API.

В качестве альтернатив рассматривались еще Flask и Django.

Flask так же использует библиотеку Starlette, но при этом не поддерживает асинхронное программирование из коробки.

Django предлагает полноценное решение, включающее ORM, административный интерфейс и множество инструментов для разработки веб-приложений. Однако, для нашего проекта, где требуется высокая производительность и поддержка асинхронных запросов, выбор остановился на более легком и быстром FastAPI.

* **Uvicorn**

Uvicorn выбран в качестве ASGI-сервера для запуска FastAPI-приложения. Альтернативы не рассматривались, так как FastAPI-документация сама рекомендует использовать именно uvicorn в качестве сервера для развертывания своих приложений, что подтверждает его эффективность и надежность.

* **Pydantic:**

Pydantic был выбран для типизации и валидации данных в запросах и ответах API. Его использование обеспечивает безопасность и надежность обработки данных, что особенно важно в веб-приложениях. В сравнении с альтернативами, такими как Marshmallow или Dataclasses, Pydantic предлагает более простой синтаксис и интегрированн с FastAPI.

* **JinjaTemplates:**

JinjaTemplates были выбраны для удобного и гибкого создания HTML-шаблонов. Они позволяют разделять логику и представление, что способствует более чистому и поддерживаемому коду. Кроме того, JinjaTemplates поддерживают наследование шаблонов и макросы, что делает процесс разработки веб-интерфейса более гибким и эффективным.

В качестве альтернатив рассматривались: Mako Templates и Chameleon Templates. Но по сравнению с данными альтернативами JinjaTemplates имеет чистый и простой синтаксис, который легко читается и понимается. Он напоминает синтаксис Python, что делает работу с ним интуитивно понятной с самого начала.

* **SQLAlchemy:**

SQLAlchemy выбран как ORM-инструмент (Object-Relational Mapping) для работы с базой данных. Его гибкость и мощные возможности позволяют легко создавать и управлять сущностями базы данных на уровне объектов Python. Это значительно упрощает процесс взаимодействия с базой данных и обеспечивает безопасность и надежность операций с данными.

Большинство ORM-инструментов схожи между собой по синтаксису и скорости работы с данными в рамках разрабатываемого программного средства. SQLAlchemy выбран благодаря наличию асинхронности и поддержке различных типов баз данных, включая SQLite, PostgreSQL, MySQL и других. Это обеспечивает гибкость в выборе базы данных в зависимости от требований проекта.

* **Pytest:**

Pytest выбран в качестве фреймворка для тестирования. Хотя Python предлагает стандартный модуль unittest, который также предоставляет средства для написания и запуска тестов, Pytest позволяет писать тесты более эффективно и удобно, что делает его предпочтительным выбором. Также его плюсом являются понятные и информативные отчеты о прохождении тестов, что упрощает анализ результатов и обнаружение проблем и цветной вывод в консоль, для большей наглядности.

* **SQLite:**

В качестве базы данных была выбрана SQLite из-за её легкости, простоты в использовании и хорошей поддержки в Python.

При выборе базы данных рассматривались различные варианты, включая: PostgreSQL, MySQL, MariaDB и MongoDB.

На фоне остальных, SQLite предоставляет все необходимые возможности для хранения и обработки данных, при этом не требует настройки сервера и администрирования, что удовлетворяет текущие требования проекта. В случае дальнейшей масштабируемости, легко можно будет перейти на другую базу данных, благодаря SQLAlchemy.

Также в проекте задействованы ряд других библиотек для шифрования паролей, импорта и экспорта файлов, работы с переменными окружения и так далее, но основной функционал приходится именно на описанные инструменты (полное дерево зависимостей приведено в прил. 6). В качестве среды разработки использован VS Code. Для реализации контроля версий, развернут git-репозиторий (<https://github.com/Exence/medical_record/>).

Эти инструменты были выбраны после анализа и оценки их соответствия требованиям проекта. Они обеспечивают высокую производительность, гибкость и надежность при разработке программного средства.

Перед написанием самого приложения рассмотрим структуру базы данных.

## Физическое проектирование баз данных

## Создание базы данных

Для использования в проекте, была создана база данных medical\_record, включающая в себя следующие таблицы:

* childrens (данные о детях);
* clinics (данные о поликлиниках)
* users (данные о пользователях);
* medical\_certificates (данные о справках от врачей);
* allergyes (данные об аллергиях);
* extra\_classes (данные о дополнительных занятиях);
* hospitalizations (данные о госпитализациях);
* spa\_treatments (данные о санаторно-курортных лечениях);
* skiping\_by\_disease (данные о пропусках занятий по болезни);
* dispensaryes (данные о диспансерных наблюдениях);
* visit\_specialist\_controls (контроль посещений специалистов);
* dewormings (данные о дегельминтизации);
* oral\_sanations (данные о санациях полости рта);
* prevaccination\_checkups (данные осмотров перед профилактическими прививками);
* vaccinations (данные о прививках);
* vac\_names (наименования прививок);
* tuberculosis\_vaccinations (данные о прививках против туберкулеза (БЦЖ));
* gamma\_globulin\_injections (данные о введении гамма-глобулина);
* mantoux\_test (данные о реакциях Манту);
* medical\_examinations (данные о медосмотрах);
* ongoing\_medical\_supervisions (данные о текущем медицинском наблюдении)
* screenings (данные скринингов).

Полный SQL-код, использованный для создания таблиц, приведен в прил. 8.

Следует отметить следующие особенности:

* Атрибуты сущности «СКРИНИНГ»: «Выявление невротических расстройств», «Мышление и речь», «Моторное развитие», «Внимание и память», «Социальные контакты», имеющие в качестве допустимых значений «Норма» либо «Отклонение» приведены в таблице в булевом формате;
* Так как SQLite не поддерживает тип данных для хранения дат, он заменен на «TEXT(10)», где дата будет представлена в виде «ГГГГ-ММ-ДД».
* Для удобства работы с данными в таблице «dispensaries» был введен суррогатный ключ «id», который является внешним ключом для поля «dispensary\_id» таблицы «visit\_specialist\_controls».

Физическая схема баз данных представлена на рис. 3.

## Поддержка ограничений целостности

Для обеспечения поддержки целостности были использованы различные средства: задание доменов (прил. 7), средства языка DDL, триггеры. Все задействованные средства приведены в табл. 2.

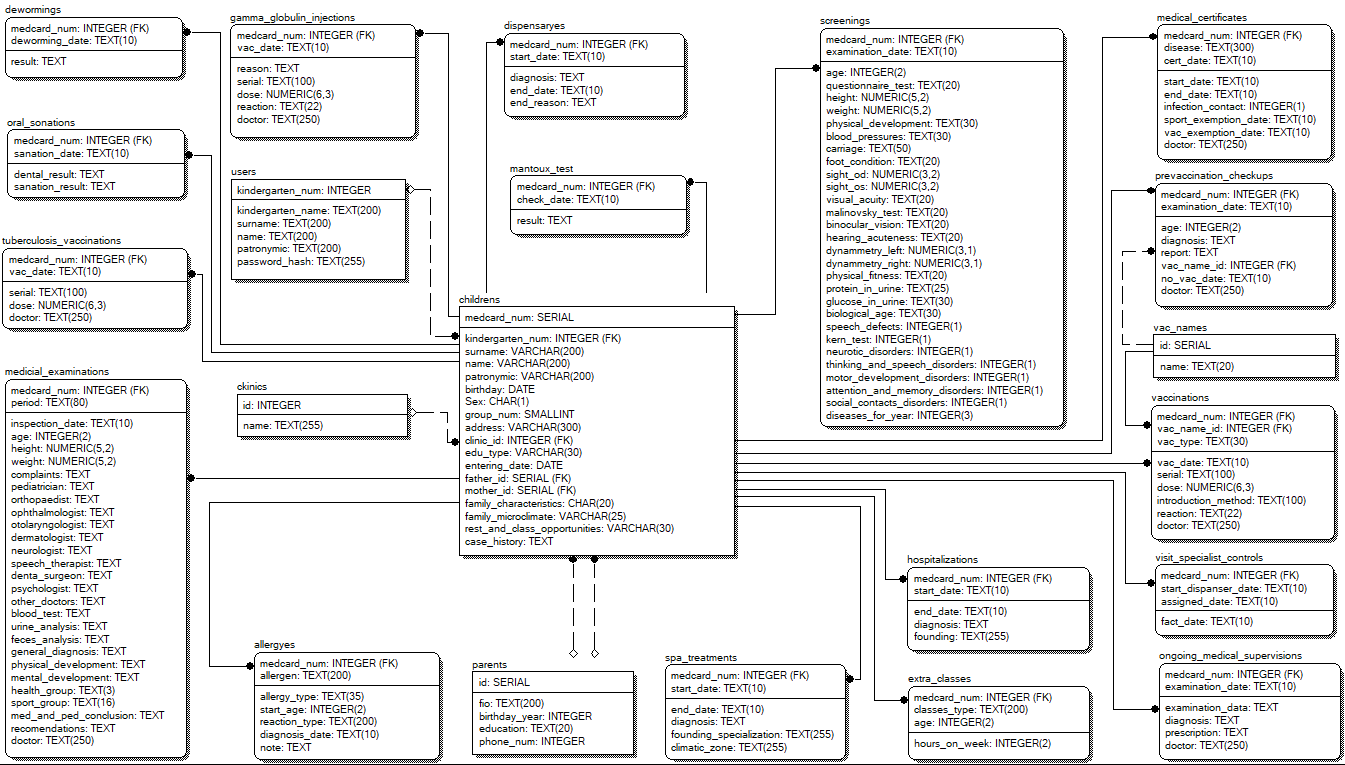


Рис. 3. Физическая схема баз данных

*Таблица 2*

**Обеспечение ограничение целостности**

|  |  |
| --- | --- |
| **Ограничение целостности** | **Средства поддержки ограничения целостности** |
| Задание диапазона значений атрибутов:  parents.education;  parents.phone\_num;  childrens.sex;  childrens.group\_num;  childrens.family\_characteristics;  childrens.family\_microclimate;  childrens.rest\_and\_class\_opportunities;  users.access\_level;  allergyes.allergy\_type;  vaccinations.vac\_type;  vaccinations.reaction;  gamma\_globulin\_injections.reaction;  medical\_examinations.period;  medical\_examinations.health\_group;  medical\_examinations.sport\_group;  screenings.questionnaire\_test;  screenings.physical\_development;  screenings.carriage;  screenings.foot\_condition;  screenings.visual\_acuity;  screenings.malinovsky\_test;  screenings.binocular\_vision;  screenings.hearing\_acuteness;  screenings.physical\_fitness;  screenings.protein\_in\_urine;  screenings.glucose\_in\_urine;  screenings.biological\_age. | Заданы ограничительные условия на значения атрибута в операторе CREATE TABLE (прил. 8) |
| Проверка корректности связи между таблицами | Созданы внешние ключи для таблиц  (прил. 8) |
| Расчет возраста в годах для атрибутов:  prevaccination\_checkups.age;  medical\_examinations.age;  screenings.age.  Расчет количества болезней за год для атрибута screenings.diseases\_for\_year | Созданы триггеры (прил. 9) |

## Реализация API

При реализации API принято решение руководствоваться следующими принципами:

* Для создания, чтения, обновления и удаления данных использовать методы POST, GET, PUT и DELETE соответственно.
* URL для API начинаются со слова «api» и далее идет версия (на случай дальнейшего масштабирования).
* Далее идет имя ресурса. Для более удобной и понятной работы с API созданы ресурсы по именам таблиц базы данных, к которым обращается данный ресурс (например: «children», «parents», «medical\_certificates» и так далее), а также отдельный ресурс «auth» – для действий, связанных с авторизацией.
* После ресурса может идти динамический параметр, по которому производится конкретная выборка элемента(-ов) из заданного ресурса (например: номер медкарты для «children»).
* После чего, может идти следующий ресурс, в который передается полученная выборка, и так далее.

Рассмотрим на примере: API версии 1 для получения списка медицинских справок ребенка с номером медкарты 5 будет выглядеть следующим образом:

GET /api/v1/children/5/medical\_certificates/

версия ресурс динамический параметр следующий ресурс

Таким образом будет сформирован единообразный, хорошо читаемый и понятный API. Далее перейдем к реализации функций.

## Реализация функций

Согласно функциональных требований (п. 1.7.2) реализуем следующие функции:

* **Редактирования данных Пользователя системы (ФИО, пароль)**

Данная функция редактирует запись в таблице «Users». Для этого будет проверена полнота и правильность заполнения всех необходимых атрибутов (благодаря типизации и использованию Pydantic).

Для реализации методов, связанных с пользователем создан отдельный сервис, который при инициализации автоматически открывает SQLAlchemy-сессию для работы с базой данных. Данный сервис содержит функцию «update\_user\_by\_ kindergarten\_num», которая принимает на вход номер детского сада (так как это первичный ключ) и обновленные данные пользователя с типом «UserModel», которые полностью совместимы с таблицей «users».

В случае, если пользователь с указанным номером детского сада не найден в базе данных, будет вызвана HTTP-ошибка с кодом 404.

Подобная логика реализована для всех методов получения, обновления и удаления данных из базы.

class UserService():

def \_\_init\_\_(self, session: Session = Depends(get\_session)):

self.session = session

…

def update\_user\_by\_kindergarten\_num(self, kindergarten\_num: int, updated\_data: UserModel) -> UserModel:

user = self.\_get(current\_data.kindergarten\_num)

if user:

for key, value in updated\_data.dict().items():

setattr(user, key, value)

self.session.commit()

user\_dict = {column.name: getattr(user, column.name) for column in user.\_\_table\_\_.columns}

return UserModel(\*\*user\_dict)

else:

raise HTTPException(

status\_code=status.HTTP\_404\_NOT\_FOUND,

detail='User is not found'

)

* **Создания новой медкарты**

Функции для добавления записи в таблицы с использованием ORM довольно тривиальны и не требуют дополнительных пояснений:

def add\_new\_medcard(self, child\_data: ChildCreate):

medcard = Child(\*\*child\_data.dict())

self.session.add(medcard)

self.session.commit()

self.session.refresh(medcard)

return medcard

Стоит только отметить, что экземпляр класса «ChildCreate» на входе не имеет поля «medcard\_num» переменная «medcard» получает значение «medcard\_num» от ORM благодаря «self.session.refresh(medcard)».

* **Удаления медкарты**

Удаление записи из таблицы, так же тривиально:

def delete\_medcard(self, medcard\_num: int):

medcard = self.\_get(medcard\_num)

self.session.delete(medcard)

self.session.commit()

Благодаря ограничениям «ON DELETE CASCADE» удаление данных из таблицы «childrens» запускает каскадное удаление данных из всех ссылающихся таблиц.

* **Импорта медкарты**

Данные в программу импортируются из файла MS Excel, для этого используется библиотека «openpyxl».

Все страницы данного табличного файла проименованы согласно названиям моделей (классов), хранящихся в них данных. По названию этих моделей создается экземпляр класса, в который передаются данные с этой страницы (первая строка таблицы содержит заголовки). Далее этот экземпляр передается в ORM и затем записывается в базу данных.

Функция для создания генератора объектов класса, заданного названием листа выглядит следующим образом:

def create\_obj\_generator\_from\_ws(ws: Worksheet, tab\_name: str):

enum\_table = {

"TableBaseModel": TableBaseModel

}

obj\_gen = ws.values

headers = next(obj\_gen)

for data in obj\_gen:

data\_as\_dict = dict(zip(headers, data))

obj = enum\_table["TableBaseModel"][tab\_name].value(\*\*data\_as\_dict)

yield obj

В данном примере:

ws – лист из MS Excel преобразованный в объект «openpyxl»

tab\_name – имя листа (имя класса, в который преобразуются данные на этом листе)

TableBaseModel – класс, в котором в качестве значений Enum хранятся ссылки на конкретные классы

dict(zip(headers, data)) – преобразует два кортежа: заголовков и данных в словарь, который затем распаковывается в экземпляр конкретного класса

Полный текст модуля приведен в прил. 10

* **Экспорта медкарты**

Экспорт медкарты решено производить в MS Excel файл, при помощи библиотеки «openpyxl».

Все страницы данного табличного файла именуются согласно названиям моделей (классов), экспортируемых в них данных. На каждую страницу первой строкой записываются заголовки, а далее построчно идут данные. Одна строка – одна запись из базы данных. Экспортированных файл содержит в обязательном порядке страницу с данными ребенка, а остальные страницы записываются согласно найденным связанным данным. В случае отсутствия в какой либо таблице связанных данных – пустая страница не создается.

Таким образом, в модуле экспорта реализованы две функции:

Создания документа с данными ребенка:

def create\_file\_with\_child\_tab(filename: str, child\_data: Child) -> None:

wb = Workbook()

ws = wb['Sheet']

ws.title = BaseTableNameModel.Child.value

headers = [column.name for column in child\_data.\_\_table\_\_.columns]

ws.append(headers)

row = [getattr(child\_data, column.name) for column in child\_data.\_\_table\_\_.columns]

ws.append(row)

wb.save(filename)

на вход передается имя файла и объект с данными ребенка

Добавления страницы в созданный:

def append\_data\_in\_xlsx\_file(data: list[BaseModel], tab\_name: BaseTableNameModel, filename: str):

if data:

wb = load\_workbook(filename)

ws = wb.create\_sheet(title=tab\_name.value)

headers = [column.name for column in data[0].\_\_table\_\_.columns]

ws.append(headers)

for item in data:

if item:

row = [getattr(item, column.name) for column in data[0].\_\_table\_\_.columns]

ws.append(row)

wb.save(filename)

на вход передается список объектов из базы для записи, имя класса, к которому относятся данные и путь к файлу.

Стоит отметить, что все модели из базы данных унаследованы от BaseModel и поэтому совместимы с данной функцией.

* **Внесения данных в медкарту (как новых, так и изменения существующих);**

Все функции добавления, обновления и удаления данных разработаны по аналогии с уже рассмотренными.

* **Формирования годового отчета (структура документа приведена в прил. 2);**

Формирование годового отчета сводится к обычным математическим действиям с отфильтрованными данными из базы.

Полный текст модуля по формированию отчетов приведен в прил. 11.

* **Формирования отчета по профилактическим прививкам, дегельминтизации и туберкулинодиагностике**

Формирование данных отчетов реализовано фильтрацией данных из базы.

Полный текст модуля по формированию отчетов приведен в прил. 12.

## Реализация интерфейса

Как уже было отмечено ранее, программное средство имеет графический web-интерфейс.

Общий замысел по использованию программного средства следующий:

* Если Пользователь не авторизован в системе, то при запуске программного средства он попадает на главное окно входа в систему, где вводит свой пароль и нажимает кнопку «Войти».
* В случае неудачной попытки входа в систему, Пользователь получает сообщение об ошибке: «Ошибка авторизации! Проверьте правильность пароля!»
* При первом входе в систему автоматически создается Пользователь со стандартным паролем и значениями полей по умолчанию.
* В случае успешной авторизации Пользователь попадает в Личный кабинет, откуда ему доступны переходы по пунктам меню, редактирование сведений о Пользователе и выход из системы.
* Дальнейшая работа строится примерно одинаковым образом: отображение данных из базы производится в виде форматированного текста, редактирование и добавление – во всплывающих окнах со специальными формами для добавления/редактирования.

При реализации web-интерфейса основные усилия были сосредоточены на создании интуитивно-понятного, не перегруженного информацией рабочего пространства, с минимальным (необходимым в данный момент, конкретным) набором отображаемых данных, но при этом с возможностью быстрого перехода между всеми доступными текущему Пользователю данными.

Также в интерфейсе присутствует общность стилей и цветовой гаммы на всех отображаемых элементах.

Функциональная структура главной страницы с приведена на рис. 8.

Интерфейс различных web-страниц программы приведен в прил. 13.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | ГЛАВНАЯ СТРАНИЦА | | |  | | |
|  | |  | | |  | |  | |
| **Работ с базой данных** | | **Работа с**  **медкартами** | | | **Отчеты** | | **Пользователи** | |
| Очистка базы | | Создание | | | Годовой отчет | | Выход | |
| **Ведение**  **справочников** > | | Импорт из файла | | | Отчет по прививкам | | **Личный кабинет** > | |
| Экспорт всех медкарт | | | Туберкулинодиагностика | |  |  |
|  |  | **Просмотр/редактирование** > | | |  | | Редактирование данных | |
| Наименования прививок | |  | |  |  | |
| Экспорт текущей медкарты | | |  | |
| Поликлиники | | Печать текущей медкарты | | |  | |  | |
| Общие сведения о ребенке | | |  | |  | |
|  | | Аллергия | | |  | |  | |
|  | | Дополнительные занятия | | |  | |  | |
|  | | Госпитализация | | |  | |  | |
|  | | Санаторно-курортное лечение | | |  | |  | |
|  | | Пропуск занятий по болезни | | |  | |  | |
|  | | Диспансерное наблюдение | | |  | |  | |
|  | | Контроль посещений специалиста | | |  | |  | |
|  | | Дегельминтизация | | |  | |  | |
|  | | Санация полости рта | | |  | |  | |
|  | | Осмотр перед  профилактическими прививками | | |  | |  | |
|  | | Прививка | | |  | |  | |
|  | | Прививка против туберкулеза (БЦЖ) | | |  | |  | |
|  | | Введение гамма-глобулина | | |  | |  | |
|  | | Реакция Манту | | |  | |  | |
|  | | Медосмотр | | |  | |  | |
|  | | Текущее медицинское наблюдение | | |  | |  | |
|  | | Скрининг | | |  | |  | |

Рис. 8. Функциональная структура главной формы программного средства

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного средства играет важную роль в обеспечении его качества, надежности и корректного функционирования. Оно направлено на выявление ошибок, дефектов и несоответствий в функциональности программного средства на ранних этапах разработки.

За основу тестирования в текущем проекте взяты юнит-тесты, охватывающие разработанные сервисы. Для поддержания актуальности этих тестов введен pre-push скрипт, который автоматически запускает pytest перед каждой попыткой выполнить git push. Если хотя бы один из тестов не проходит успешно, операция push отменяется.

Для оптимизации времени выполнения тестов, замедляющихся из-за взаимодействия с базой данных, эта часть заменена на моки. В юнит-тестах также использованы фикстуры для изоляции тестовых наборов и данных.

Тестовые наборы сервиса аутентификации и сервиса для работы с данными по аллергиям приведены в прил. 14. Остальные сервисы тестируются схожим образом, поэтому примеры кода не приводятся отдельно.

Все файлы тестовых модулей в проекте начинаются с префикса «test\_», при желании можно ознакомиться с ними в репозитории.

Тестирование графического интерфейса производится вручную, а также доступна возможность ручного тестирования API с использованием развернутого SWAGGER-UI (рис. 4).

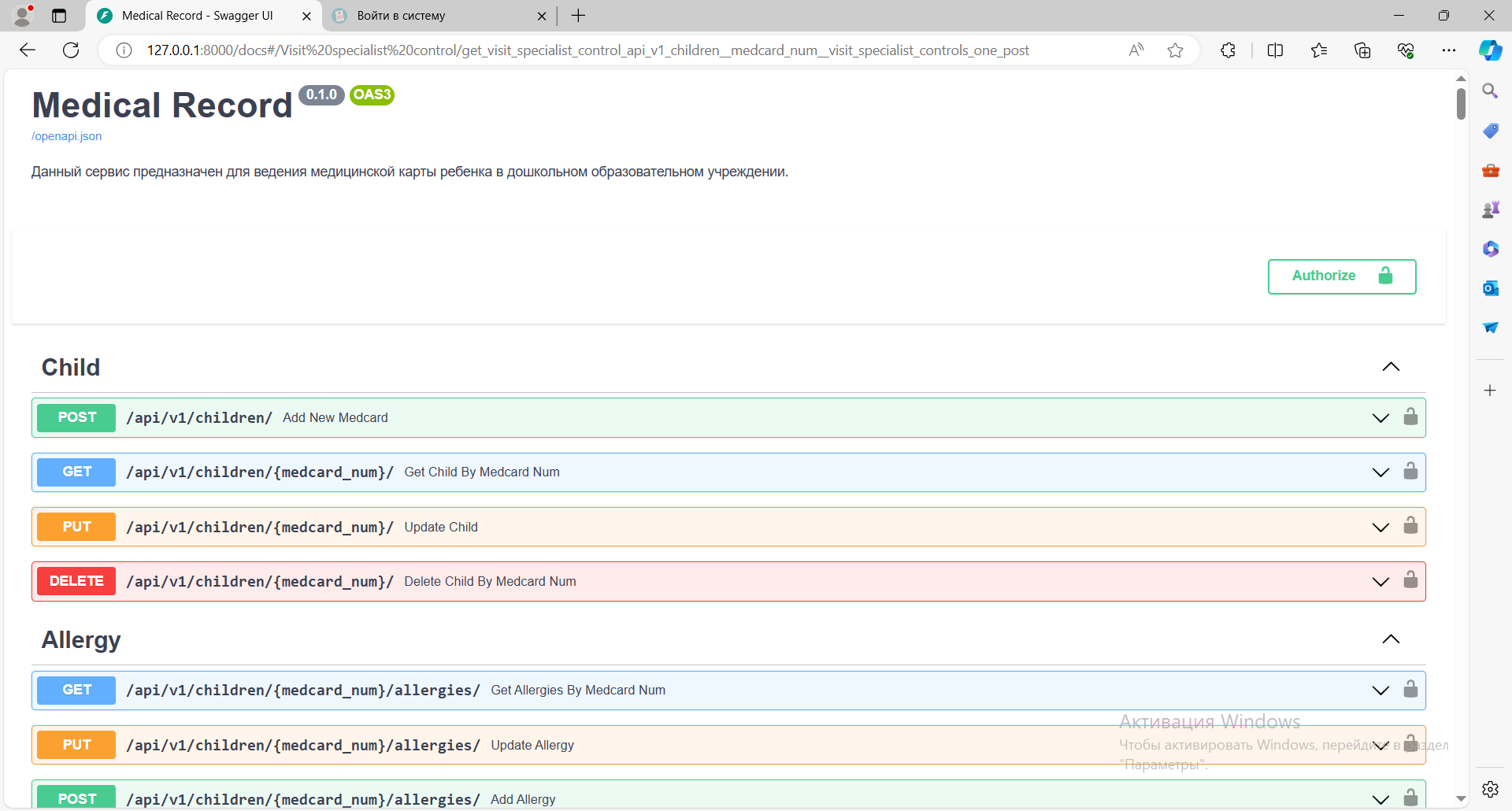


Рис. 4 – Общий вид SWAGGER-UI

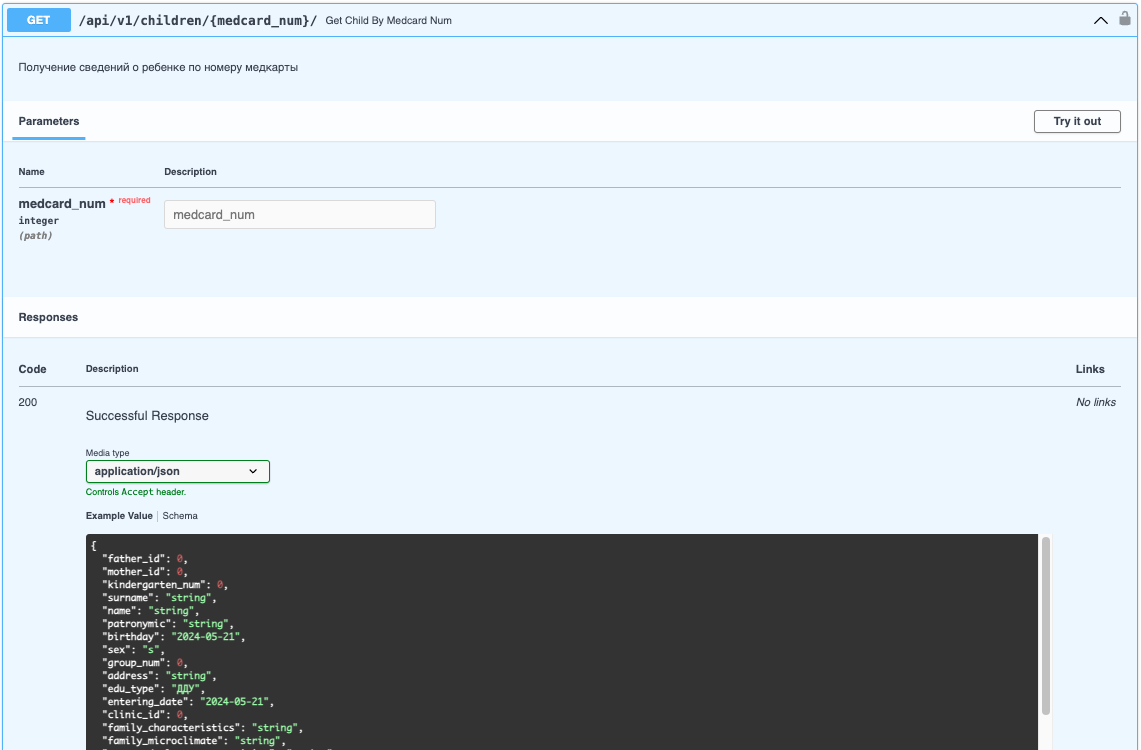
В таблице 3 представлен тестовый набор проверки получения данных ребенка из медкарты с номером 1.

Таблица 3

**Тестовый набор**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тип интерфейса** | **Ввод корректного значения** | **Ответ на корректное значение** | **Ввод некорректного значения** | **Ответ на некорректное значение** |
| API | Обращение к GET /api/v1/children/1/ авторизованного Пользователя | HTTP-cтатус ответа: 200  В теле ответа возвращаются данные ребенка с медкартой номер 1 | Обращение к GET /api/v1/children/1/ Пользователя, не прошедшего авторизацию | HTTP-cтатус ответа: 401  Ошибка аутентификации |
| APP | Обращение к странице /medical\_record/ children/1/ авторизованного Пользователя | Отображение данных ребенка с медкартой номер 1 | Обращение к странице /medical\_record/ children/1/ Пользователя, не прошедшего авторизацию | Переадресация на страницу входа в систему |

Для тестирования API найдем его в SWAGGER-UI и развернем (Рис. 5)

Рис. 5 – Тестирование конкретного API в SWAGGER-UI

В данном окне можно просмотреть описание, необходимые параметры запроса и их типы, а также схемы ответов (Рис. 6)

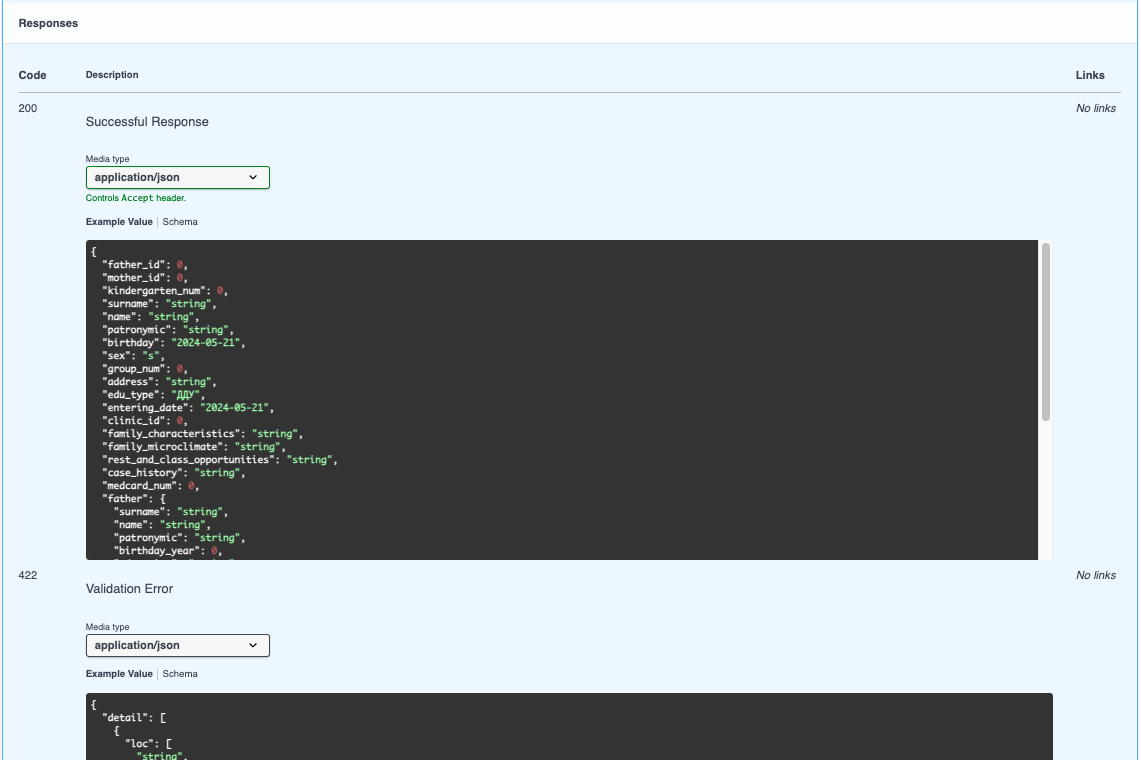


Рис. 6 – Вывод схемы ответа

При нажатии кнопки «Try it out» и становятся активным поле для ввода medcard\_num и появляется кнопка «Execute» (Рис. 7).

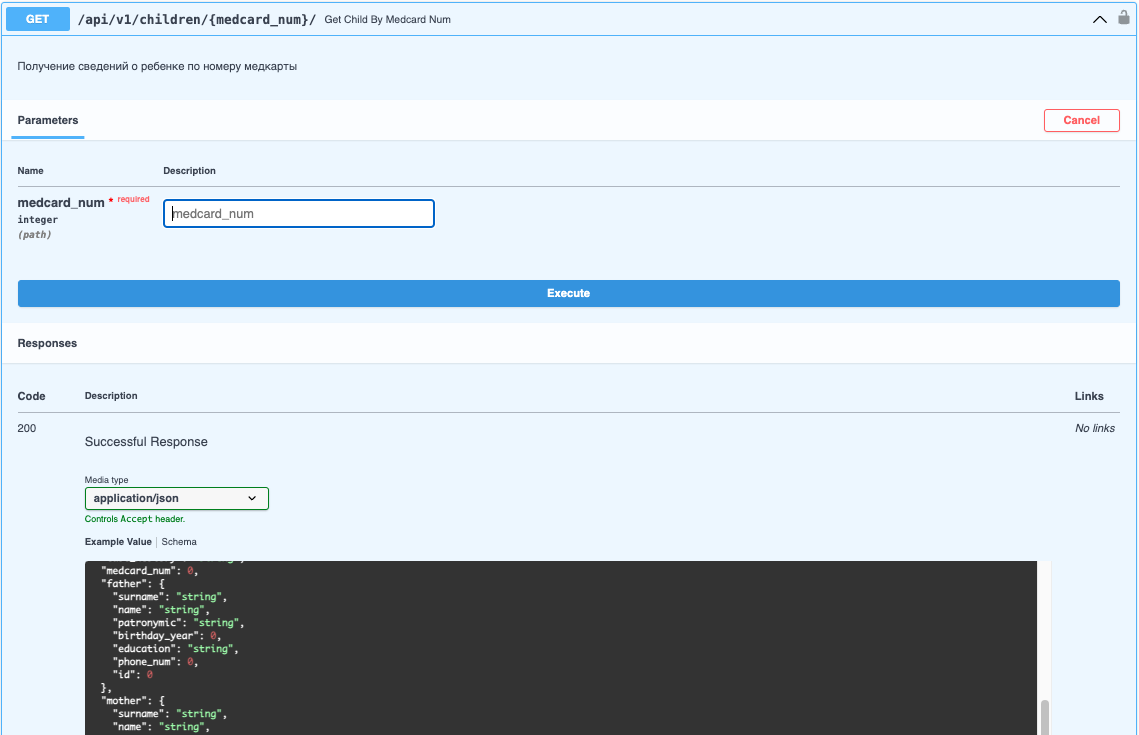


Рис. 7 – Подготовка к тестированию API

Вводим значение medcard\_num = 1 и нажимаем «Execute». В результате чего получим ошибку аутентификации (Рис. 8)

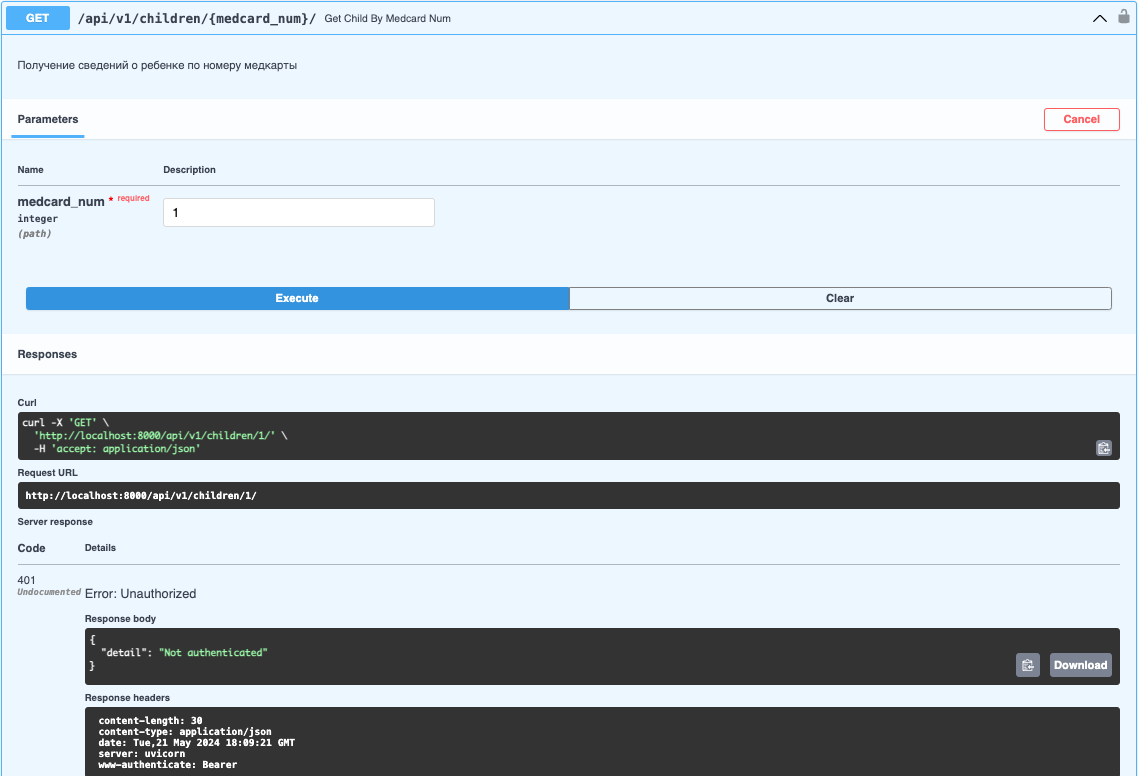
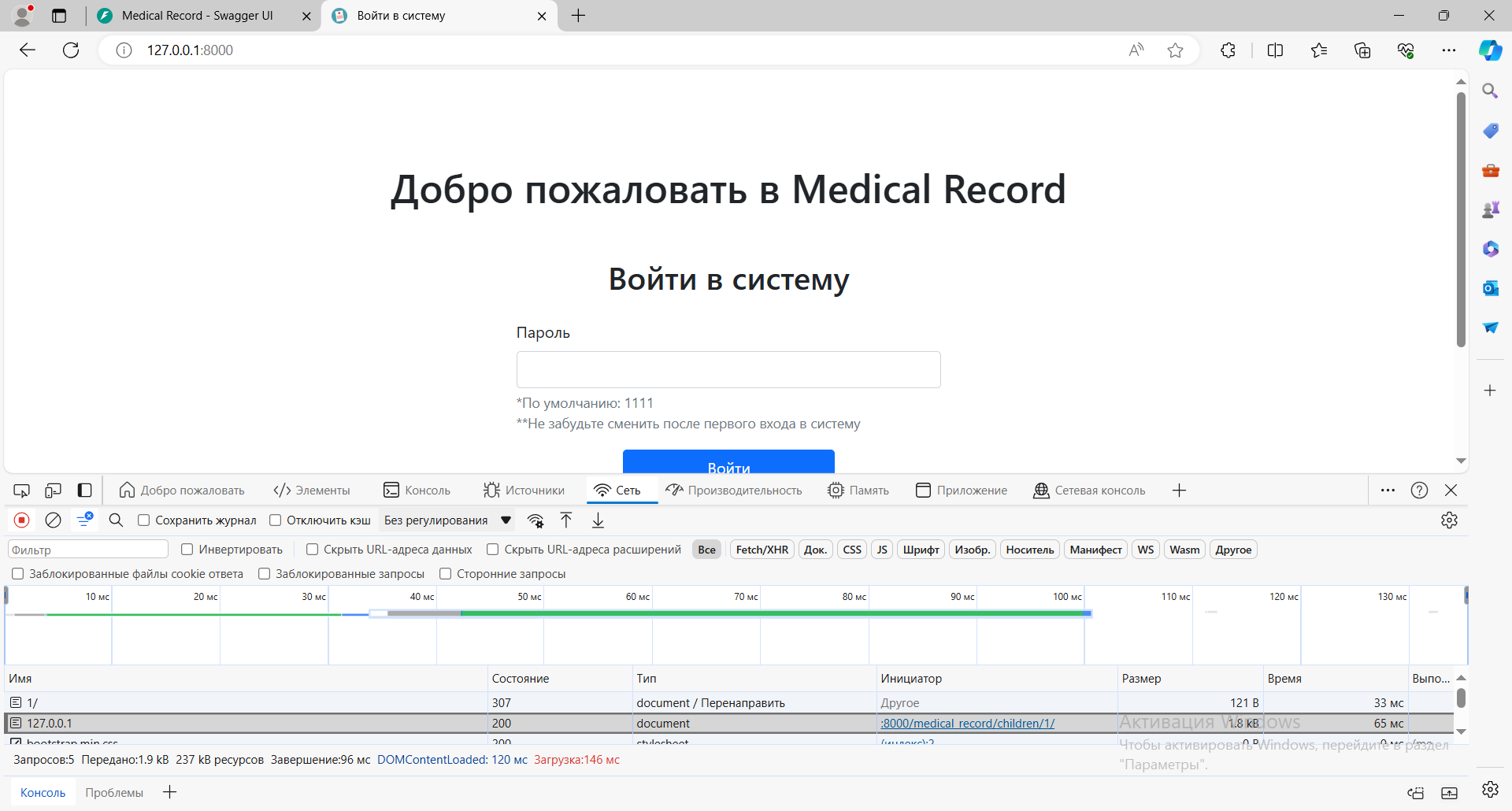


Рис. 8 – Ошибка аутентификации

При попытке перехода на любую страницу неавторизованным пользователем, он перенаправляется а окно входа (Рис. 9).



Происходит перенаправление

Пройдем авторизацию в графическом интерфейсе, либо нажав на кнопку «Authorize» в SWAGGER-UI (Рис. 10) и повторим запрос.

Видим, что в заголовках появился токен пользователя и система возвращает корректный ответ от API (Рис. 11) и в графическом интерфейсе (Рис. 12).

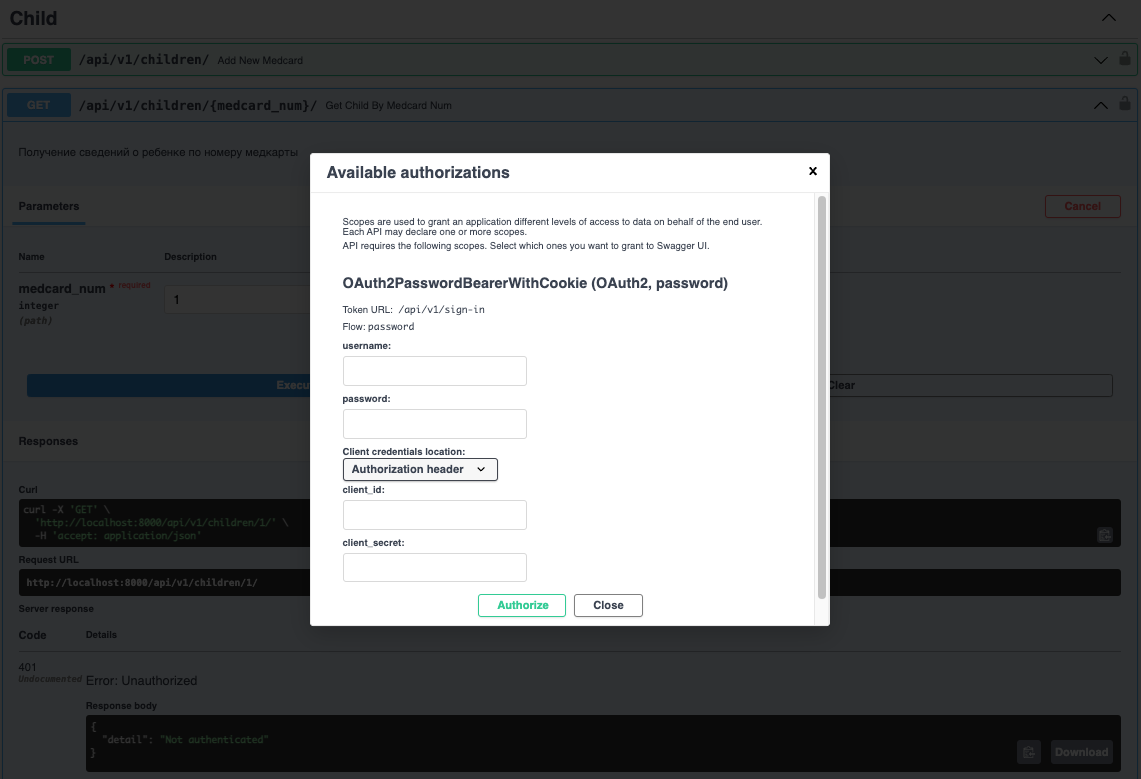


Рис. 10 – Окно авторизации SWAGGER-UI

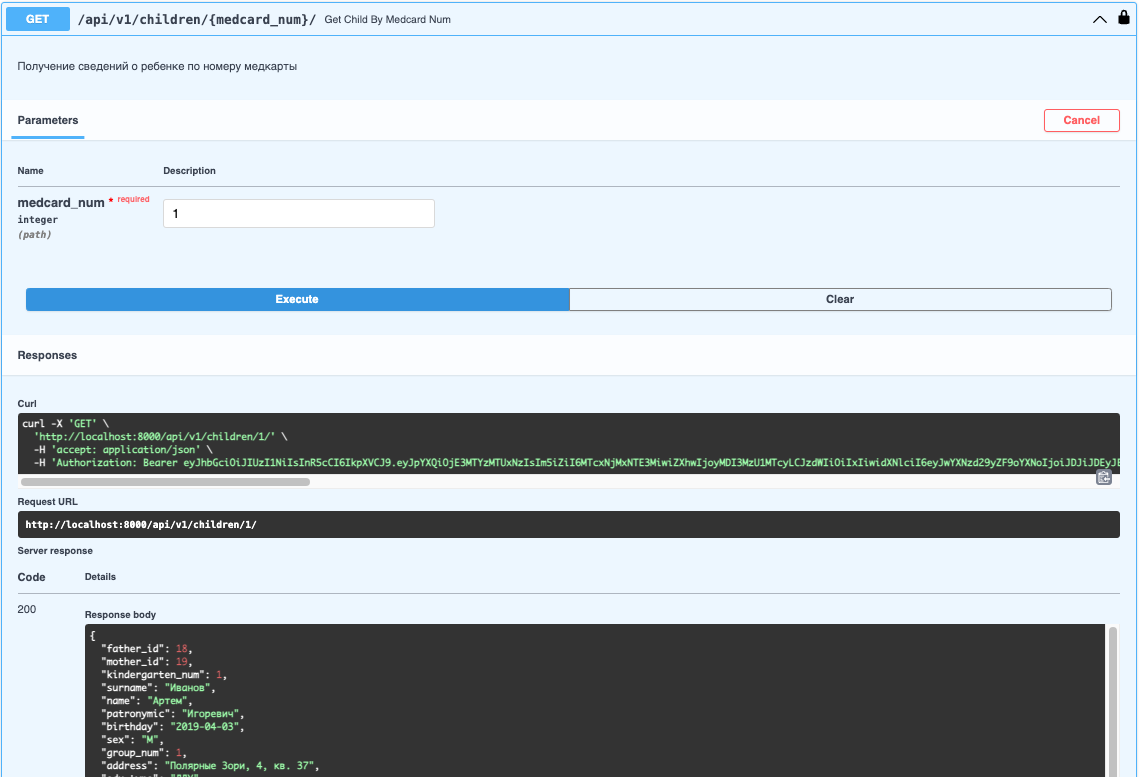


Рис. 11 – Ответ для авторизованного пользователя в SWAGGER-UI

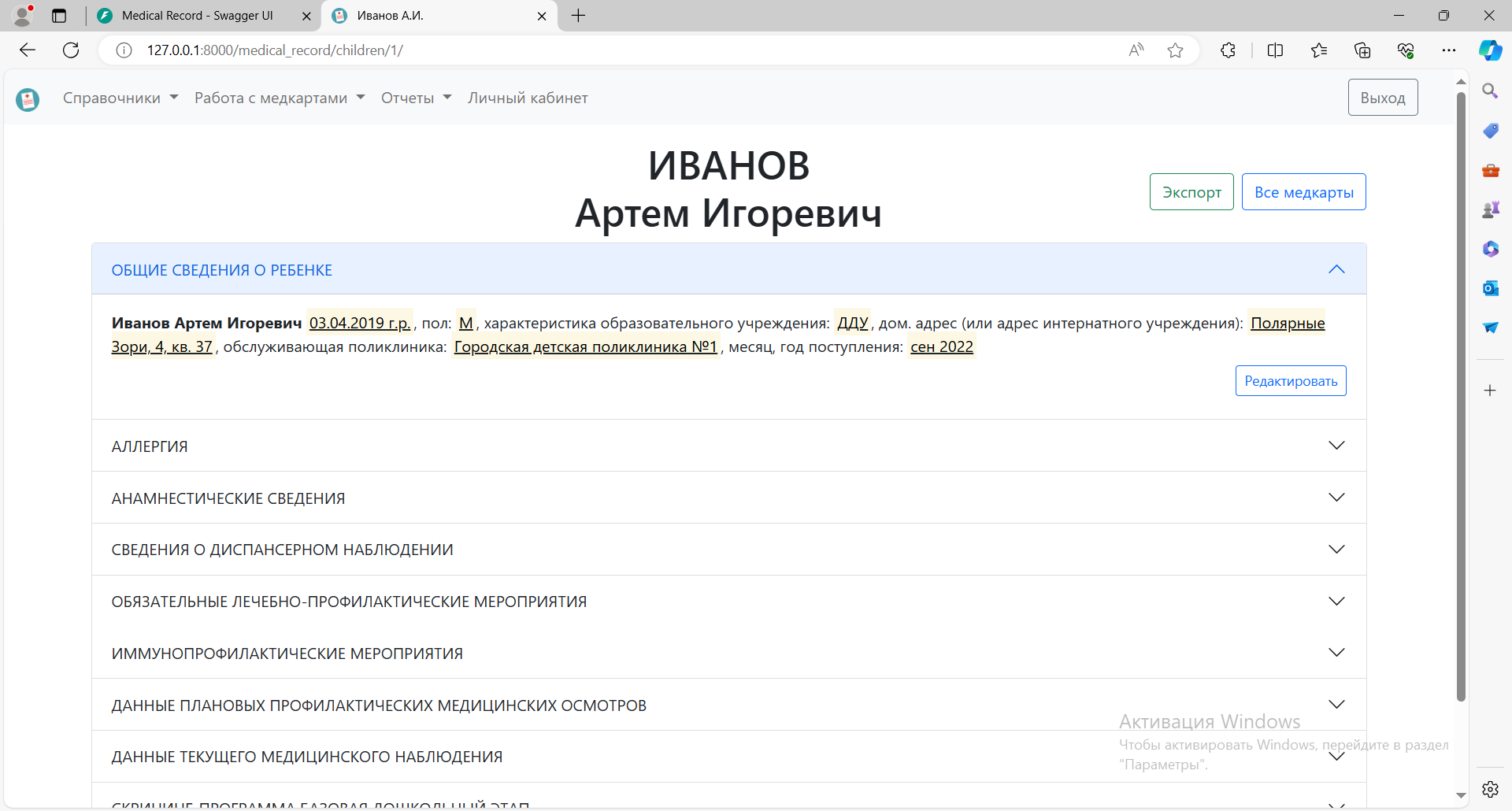


Рис. 12 – Отображение выбранной медкарты для авторизованного Пользователя

В случае ввода некорректного значения «medcard\_num», получим ошибку 404 в API (Рис. 12) и в графическом интерфейсе (Рис. 13)

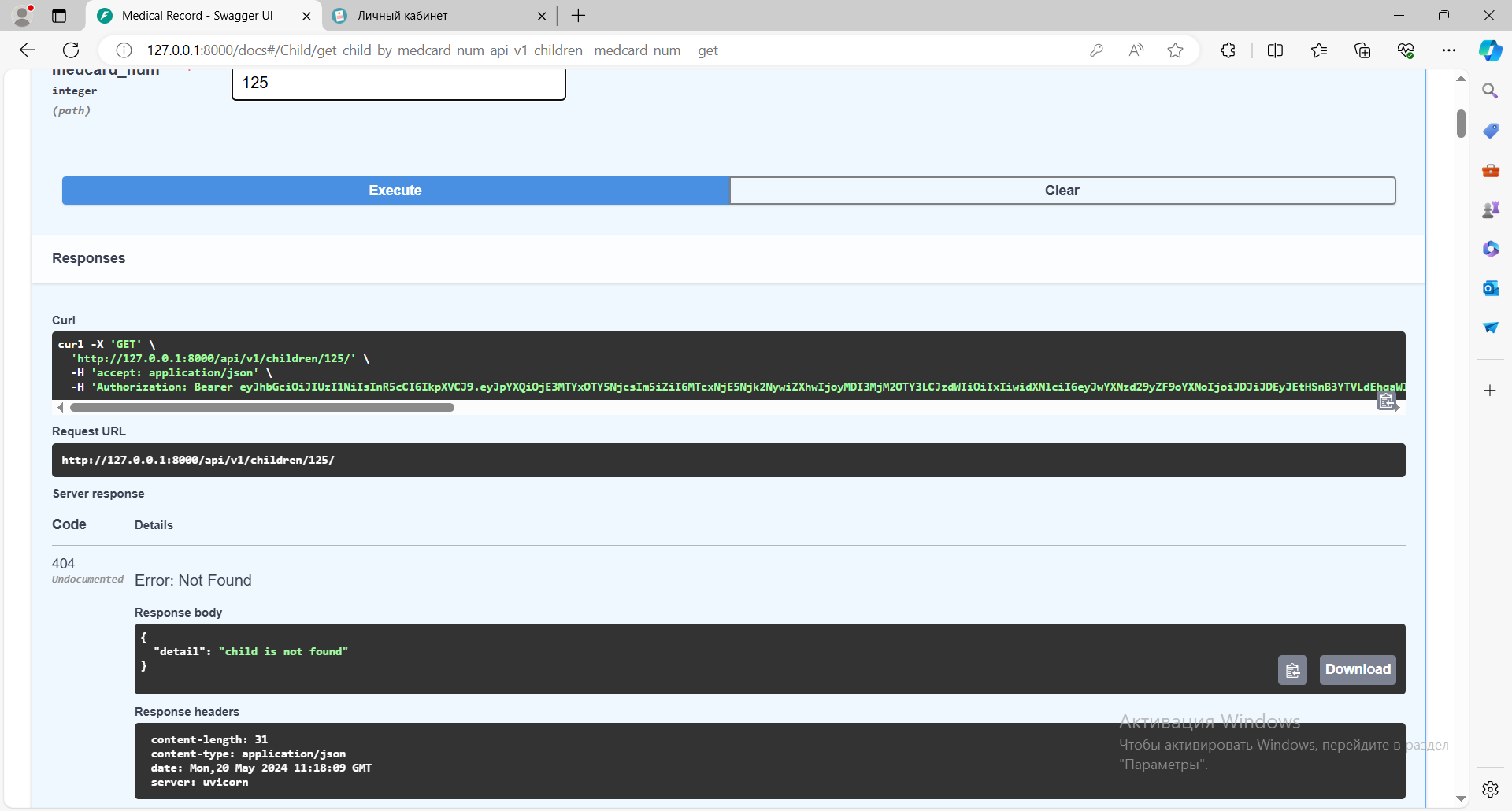


Рис. 13 – Ответ от API при некорректном значении medcard\_num

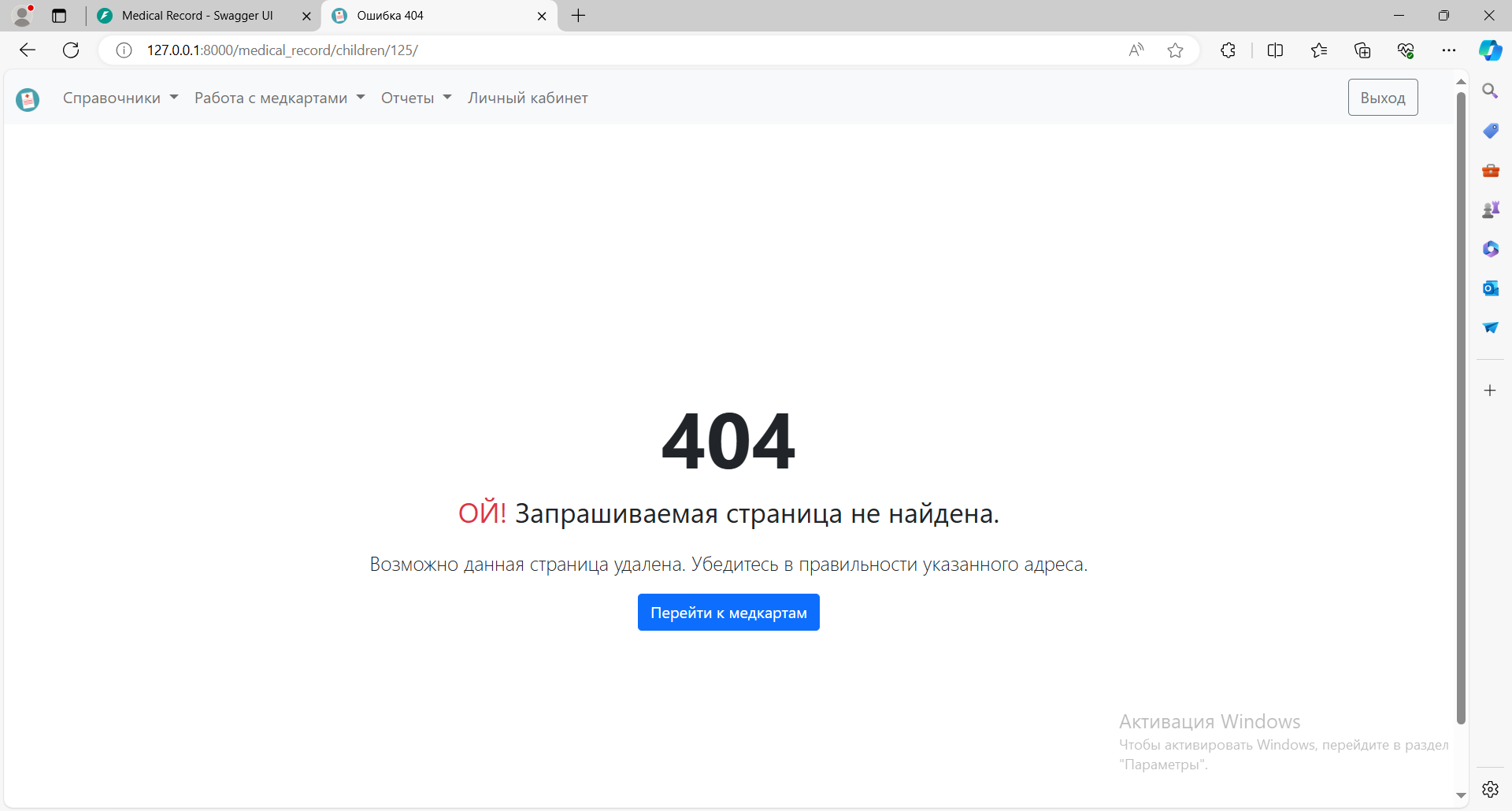


Рис. 14 – Отображение ошибки в графическом интерфейсе

Так как текущая функциональность API практически полностью покрыта юнит-тестами, то рассмотрим только тестовые наборы для валидации формы создания новой медкарты в графическом интерфейсе  
(табл. 3).

Таблица 3. Тестовый набор заполнения формы

| **Ввод корректного значения** | **Ответ на корректное значение** | **Ввод некорректного значения** | **Ответ на некорректное значение** |
| --- | --- | --- | --- |
| Фамилия: Иванов  Имя: Иван  Отчество: Иванович | Ошибок нет | Ввод пустого значения | Ошибка, форма не отправляется |
| Ввод цифр |
| Ввод количества символов свыше 140 | Дальнейший ввод символов невозможен |
| Адрес: г. Москва, Красная Площадь, д.1 | Ошибок нет | Ввод пустого значения | Ошибка, форма не отправляется |
| Дата рождения: 01.01.2022 | Ошибок нет | Ввод пустого значения | Ошибка, форма не отправляется |
| Дата поступления:  01.09.2023 | Ошибок нет | Ввод пустого значения | Ошибка, форма не отправляется |
| Фамилия отца: Иванов  Имя отца: Иван  Отчество отца: Иванович | Ошибок нет | Ввод пустого значения | Ошибка, форма не отправляется |
| Ввод цифр |
| Ввод количества символов свыше 140 | Дальнейший ввод символов невозможен |
| Год рождения отца: 1983 | Ошибок нет | Ввод пустого значения | Ошибка, форма не отправляется |
| Ввод символов, отличных от цифр |
| Ввод количества символов свыше 4 | Дальнейший ввод символов невозможен |
| Номер телефона отца:  1234567890 | Ошибок нет | Ввод пустого значения | Ошибка, форма не отправляется |
| Ввод символов, отличных от цифр |
| Ввод количества символов свыше 4 | Дальнейший ввод символов невозможен |

Примеры форм, содержащих валидные и не валидные данные приведены на Рис. 15-16. Общий стиль и шаблоны валидации схожи и для других частей программного средства.

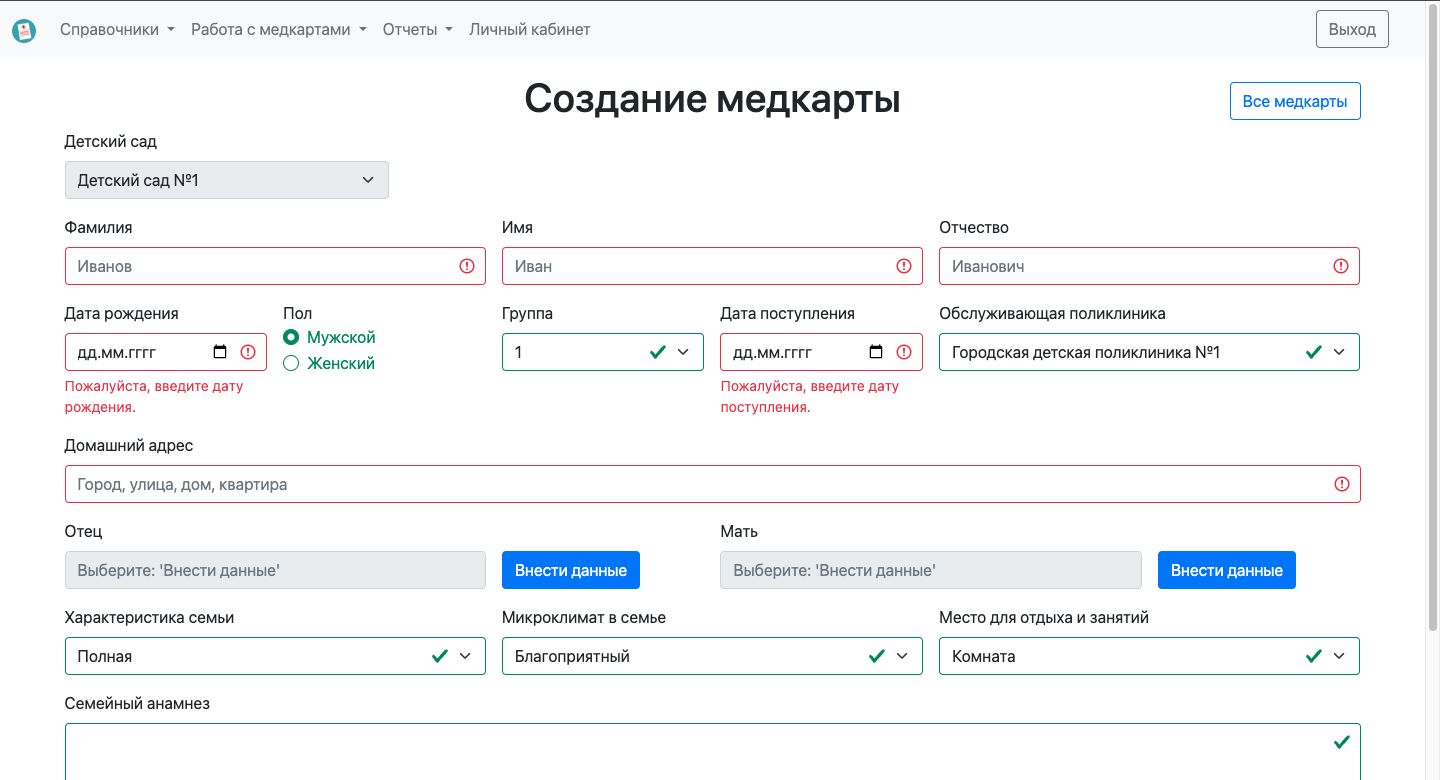


Рис. 15 – Пример формы, содержащей корректные и не корректные данные

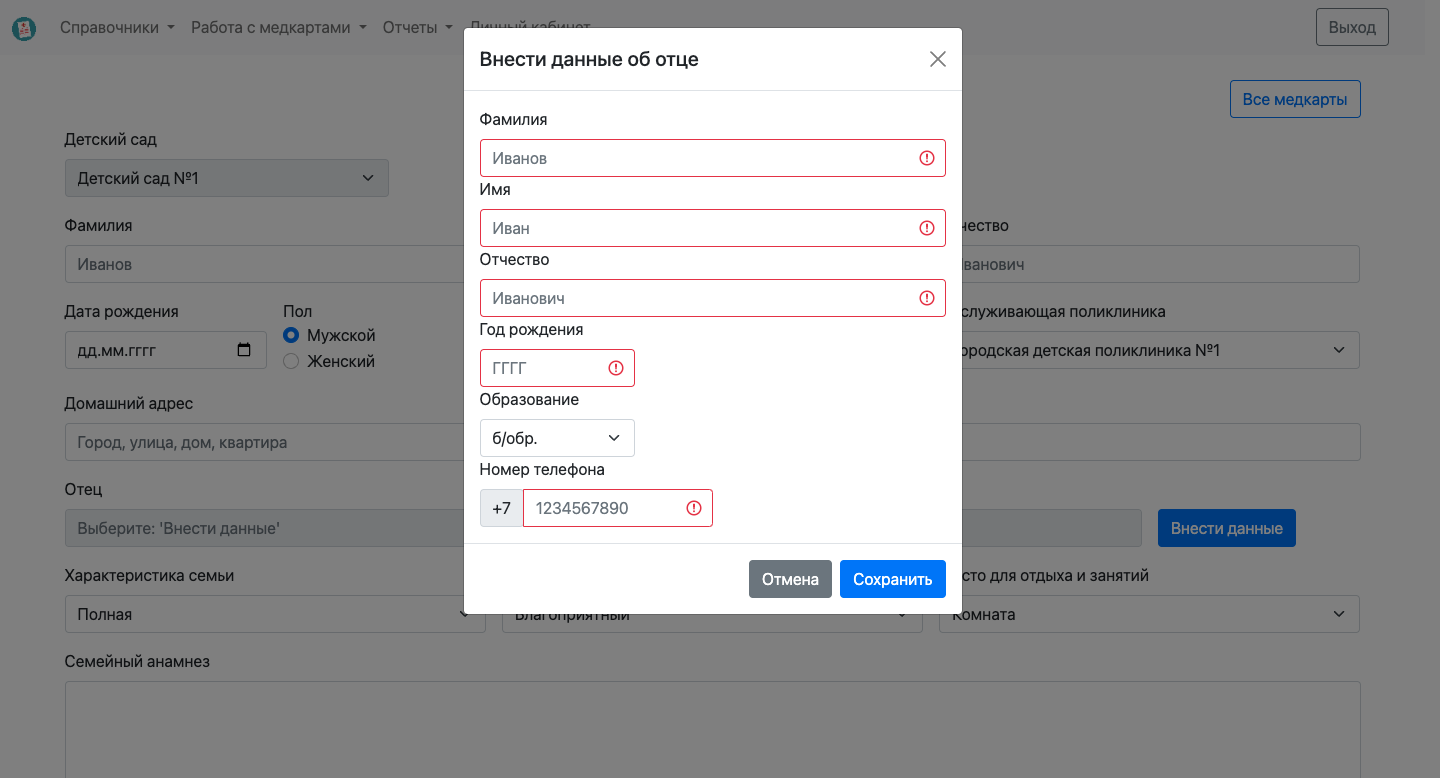


Рис. 15 – Пример формы, содержащей пустые значения

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Приказ Минздрава РФ от 05.11.2013 N 822н (ред. от 21.02.2020) «Об утверждении Порядка оказания медицинской помощи несовершеннолетним, в том числе в период обучения и воспитания в образовательных организациях»
2. Приказ Минздрава РФ от 03.07.2000 N 241 «Об утверждении «Медицинской карты ребенка для образовательных учреждений» (вместе с «Инструкцией о порядке ведения учетной формы N 026/у-2000 «Медицинская карта ребенка для образовательных учреждений дошкольного, начального общего, основного общего, среднего (полного) общего образования, учреждений начального и среднего профессионального образования, детских домов и школ-интернатов»)

# *Приложение 1*

**Структура документа**

**«Медицинская карта ребенка для образовательных учреждений дошкольного, начального общего, основного общего, среднего (полного) общего образования, учреждений начального и среднего профессионального образования, детских домов и школ-интернатов»**

| **Реквизит**  **документа** | **Тип**  **данных** | **Кол-во разрядов в**  **знаках** | **Формат**  **представления в документах** | **Допустимое**  **значение** | **Допустимость пустых значений (Да/Нет)** | **Примечание** |
| --- | --- | --- | --- | --- | --- | --- |
| **ОБЩИЕ СВЕДЕНИЯ О РЕБЕНКЕ** | | | | | | |
| ФИО | Текст | 255 |  |  | Нет |  |
| Дата рождения | Дата | 8 | ДД.ММ.ГГ |  | Нет |  |
| Пол | Текст | 1 |  | 1. М  2. Ж | Нет |  |
| Дом. адрес (или адрес интернатного учреждения) | Текст | 255 |  |  | Нет |  |
| Тел. м/жит | Число | 11 | 81234567890 |  |  |  |
| Обслуживающая поликлиника | Текст | 255 |  |  | Нет |  |
| Характеристика образовательного учреждения | Текст | 3 |  | 1. ДДУ | Нет |  |
| Месяц, год поступления | Дата | 8 | ДД.ММ.ГГ |  | Нет |  |
| **АЛЛЕРГИЯ** | | | | | | |
| Вакцинальная, лекарственная, аллергические заболевания | Текст | 50 |  |  |  |  |
| Аллерген | Текст | 100 |  |  |  |  |
| Возраст начала | Число | 2 |  |  |  |  |
| Тип реакции | Текст | 100 |  |  |  |  |
| Год уст. диагноза | Число | 4 |  |  |  |  |
| Примечания | Текст | 255 |  |  |  |  |
| **АНАМНЕСТИЧЕСКИЕ СВЕДЕНИЯ** | | | | | | |
| Родители | Текст | 4 |  | 1. Мать  2. Отец |  |  |
| ФИО | Текст | 255 |  |  |  |  |
| Год рождения | Числовой | 4 |  |  |  |  |
| Образование | Числовой | 1 |  | 1. б/обр.,  2. н/ср.,  3. ср.,  4. ср.-спец.,  5. н/высш.,  6. высш. |  |  |
| Характеристика семьи | Текст | 8 |  | 1. Полная,  2. Неполная |  |  |
| Микроклимат в семье | Текст | 15 |  | 1. Благоприятный  2. Неблагоприятный |  |  |
| Наличие у ребенка места для отдыха и занятий | Текст | 20 |  | 1. Комната  2. Индивидуаль-ный стол  3. Нет | Нет |  |
| Семейный анамнез (заболевания у родственников первого и второго поколения) | Текст | 255 |  |  |  |  |
| Вид занятий | Текст | 100 |  | 1. Спорт (указать какой, в  т.ч. танцы)  2. Музыка  3. Иностранный язык  4. Другие занятия (указ.) |  |  |
| Возраст/час. в неделю | Текст | 10 | XX/YY,Y |  |  |  |
| Перенесенные заболевания | Текст | 50 |  |  |  |  |
| Дата начала заболевания | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Дата окончания заболевания | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Сведения о госпитализации (включая травмы, операции) | | | | | | |
| Дата начала | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Дата окончания | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Диагноз, вид вмешательства | Текст | 255 |  |  |  |  |
| Учреждение | Текст | 255 |  |  |  |  |
| Сведения о санаторно-курортном (и приравненном к нему) лечении | | | | | | |
| Дата начала | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Дата окончания | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Диагноз | Текст | 255 |  |  |  |  |
| Профиль учреждения | Текст | 50 |  |  |  |  |
| Климатическая зона учреждения | Текст | 50 |  |  |  |  |
| Пропуск занятий по болезни | | | | | | |
| Дата начала | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Дата окончания | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Диагноз | Текст | 255 |  |  |  |  |
| **СВЕДЕНИЯ О ДИСПАНСЕРНОМ НАБЛЮДЕНИИ** | | | | | | |
| Диагноз | Текст | 255 |  |  |  |  |
| Специалист | Текст | 100 |  |  |  |  |
| Дата взятия | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Дата снятия | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Причина снятия | Текст | 255 |  |  |  |  |
| Контроль посещений специалиста | | | | | | |
| Назначено | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Явка | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| **ОБЯЗАТЕЛЬНЫЕ ЛЕЧЕБНО-ПРОФИЛАКТИЧЕСКИЕ МЕРОПРИЯТИЯ** | | | | | | |
| Дегельминтизация | | | | | | |
| Дата | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Результат | Текст | 255 |  |  |  |  |
| Санация полости рта | | | | | | |
| Дата | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Данные осмотра стоматологом | Текст | 255 |  |  |  |  |
| Результаты санации | Текст | 255 |  |  |  |  |
| **ИММУНОПРОФИЛАКТИЧЕСКИЕ МЕРОПРИЯТИЯ** | | | | | | |
| Осмотр перед профилактическими прививками | | | | | | |
| Дата | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Возраст | Число | 1 |  |  |  |  |
| Диагноз | Текст | 255 |  |  |  |  |
| Заключение | Текст | 11 |  | 1. Разрешено  2. Неразрешено  3. Отказ |  |  |
| Прививка | Текст | 100 |  |  |  |  |
| Мед. отвод до | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Врач | Текст | 255 |  |  |  |  |
| Профилактические прививки | | | | | | |
| Прививка | Текст | 100 |  | 1. Полиомиелит  2. Дифтерия, коклюш, столбняк  3. Паротит  4. Корь  5. Гепатит «В»  6. Краснуха |  |  |
| Вакцинация I | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Вакцинация II | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Вакцинация III | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Ревакцинация I | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Ревакцинация II | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Ревакцинация III | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Ревакцинация IV | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Серия, доза | Текст | 100 |  |  |  |  |
| Способ введения | Текст | 100 |  |  |  |  |
| Реакция | Текст | 15 |  | 1. Немедленная  2. Замедленная |  |  |
| Врач | Текст | 255 |  |  |  |  |
| Прививки по эпидпоказаниям | | | | | | |
| Название |  |  |  |  |  |  |
| Вакцинация I | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Вакцинация II | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Вакцинация III | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Ревакцинация I | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Ревакцинация II | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Ревакцинация III | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Ревакцинация IV | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Серия, доза | Текст | 100 |  |  |  |  |
| Способ введения | Текст | 100 |  |  |  |  |
| Реакция | Текст | 15 |  | 1. Немедленная  2. Замедленная |  |  |
| Врач | Текст | 255 |  |  |  |  |
| Введение гамма-глобулина (по показаниям) | | | | | | |
| Дата | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Причина | Текст | 255 |  |  |  |  |
| Серия, доза | Текст | 100 |  |  |  |  |
| Реакция | Текст | 15 |  | 1. Немедленная  2. Замедленная |  |  |
| Врач | Текст | 255 |  |  |  |  |
| Реакция Манту | | | | | | |
| Дата | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Результат | Текст | 255 |  |  |  |  |
| Врач | Текст | 255 |  |  |  |  |
| Прививка против туберкулеза (БЦЖ) | | | | | | |
| Дата | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Доза | Текст | 100 |  |  |  |  |
| Серия | Число | 6 |  |  |  |  |
| Врач | Текст | 255 |  |  |  |  |
| Данные плановых профилактических медицинских осмотров | | | | | | |
| Период обследования | Текст | 42 |  | 1. Перед поступлением в ясли-сад, детский сад  2. За 1 год до  школы  3. Перед школой |  |  |
| Дата обследования | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Возраст | Число | 3 |  |  |  | лет, месяцев |
| Длина тела, см | Число | 3 |  |  |  |  |
| Масса тела, кг | Число | 4 |  |  |  |  |
| Жалобы | Текст | 255 |  |  |  |  |
| Педиатр | Текст | 255 |  |  |  |  |
| Ортопед | Текст | 255 |  |  |  |  |
| Офтальмолог | Текст | 255 |  |  |  |  |
| Отоларинголог | Текст | 255 |  |  |  |  |
| Дерматолог | Текст | 255 |  |  |  |  |
| Невролог | Текст | 255 |  |  |  |  |
| Логопед | Текст | 255 |  |  |  | с 3-х лет |
| Стоматолог | Текст | 255 |  |  |  |  |
| Психолог | Текст | 255 |  |  |  |  |
| Другие | Текст | 255 |  |  |  |  |
| Анализ крови | Текст | 255 |  |  |  |  |
| Анализ мочи | Текст | 255 |  |  |  |  |
| Анализ кала | Текст | 255 |  |  |  |  |
| Заключительный диагноз | Текст | 255 |  |  |  |  |
| Оценка физического развития | Текст | 255 |  |  |  |  |
| Оценка нервно-психического развития | Текст | 255 |  |  |  |  |
| Группа здоровья | Текст | 3 |  | I - IV |  |  |
| Мед. гр. для занятий физкультурой | Текст | 8 |  | 1. Основная  2. ЛФК |  |  |
| Медико-педагогическое заключение | Текст | 255 |  |  |  |  |
| Рекомендации | Текст | 255 |  |  |  |  |
| **ДАННЫЕ ТЕКУЩЕГО МЕДИЦИНСКОГО НАБЛЮДЕНИЯ** | | | | | | |
| Дата | Дата | 8 | ДД.ММ.ГГ |  |  |  |
| Данные осмотра | Текст | 255 |  |  |  |  |
| Диагноз | Текст | 255 |  |  |  |  |
| Назначения | Текст | 255 |  |  |  |  |
| Врач | Текст | 255 |  |  |  |  |
| **СКРИНИНГ-ПРОГРАММА БАЗОВАЯ ДОШКОЛЬНЫЙ ЭТАП** | | | | | | |
| Возраст | Число | 1 |  | 3-5 |  |  |
| Анкетный тест | Текст | 10 |  | 1. Норма,  2. Отклонение |  |  |
| Рост, см | Число | 3 |  |  |  |  |
| Масса, кг | Число | 4 |  |  |  |  |
| Физическое  развитие | Текст | 15 |  | 1. Нормальное  2. Низкий рост  3. Дефицит массы  4. Избыток массы |  |  |
| АД, мм.рт.ст. | Текст | 7 |  |  |  |  |
| Осанка | Текст | 25 |  | 1. Нормальная  2. Незначительные  отклонения  3. Значительные  нарушения |  |  |
| Состояние стопы | Текст | 10 |  | 1. Нормальная  2. Уплощена  3 Плоская |  |  |
| Зрение, ОД | Число | 3 |  |  |  |  |
| Зрение, OS | Число | 3 |  |  |  |  |
| Острота зрения | Текст | 10 |  | 1. Нормальная  2. Снижена |  |  |
| Тест Малиновского | Текст | 10 |  | 1. Нормальная  2. Предмиопия |  |  |
| Бинокулярное зрение | Текст | 9 |  | 1. Норма  2. Нарушение |  |  |
| Острота слуха | Текст | 7 |  | 1. Норма  2. Снижена |  |  |
| Динамометрия,  Правая рука | Число | 5 |  |  |  |  |
| Динамометрия,  Левая рука | Число | 5 |  |  |  |  |
| Физическая  подготовленность | Текст | 8 |  | 1. Норма  2. Снижена  3. Повышена |  |  |
| Определение белка в моче | Текст | 12 |  | 1. Норма  2. Следы белка  3. Белок в моче |  |  |
| Определение глюкозы в моче | Текст | 14 |  | 1. Норма  2. Глюкоза в моче |  |  |
| Расширенная скрининг-программа | | | | | | |
| Биологический возраст | Текст | 13 |  | 1. Соответствует  2. Опережает  3. Отстает |  |  |
| Дефекты речи | Текст | 4 |  | 1. Есть  2. Нет |  |  |
| Тест  Керна-Иерасика | Число | 5 |  |  |  |  |
| Выявление невротических  расстройств | Текст | 10 |  | 1. Норма  2. Отклонение |  |  |
| Нервнопсихическое  развитие | Текст | 19 |  | 1. Мышление и речь  2. Моторное  развитие  3. Внимание и  память  4. Социальные  контакты |  |  |
| Оценка нервнопсихического  развития | Текст | 10 |  | 1. Норма  2. Отклонение |  |  |
| Число заболеваний за год | Число | 2 |  |  |  |  |

# *Приложение 2*

**Структура документа «Годовой отчет»**

| **Реквизит**  **документа** | **Тип**  **данных** | **Кол-во разрядов в**  **знаках** | **Формат**  **представления в документах** | **Допустимое**  **значение** | **Допустимость пустых значений (Да/Нет)** | **Примечание** |
| --- | --- | --- | --- | --- | --- | --- |
| Отчетный год | Число | 4 | ГГГГ |  |  |  |
| Возраст | Текст |  |  | 1. 1-3 года  2. 4-6 лет  3. Поступ. в 1 класс |  |  |
| Численность детей | Число | 3 |  |  | Нет |  |
| Количество частоболеющих детей (абсолютно) | Число | 3 |  |  | Да |  |
| Количество частоболеющих детей (процент) | Число | 3 |  |  | Да | Расчетный |
| Индекс здоровья (абсолютно) | Число | 3 |  |  | Да |  |
| Индекс здоровья (процент) | Число | 3 |  |  | Да | Расчетный |
| **Общая заболеваемость** | | | | | | |
| Число случаев (абсолютно) | Число | 3 |  |  | Да |  |
| Число случаев (на 1000 детей) | Число | 3 |  |  | Да | Расчетный |
| Число дней, пропущенных по болезни (абсолютно) | Число | 3 |  |  | Да |  |
| Число дней, пропущенных по болезни (на 1 ребенка) | Число | 3 |  |  | Да | Расчетный |
| **Острая заболеваемость** | | | | | | |
| Число случаев (абсолютно) | Число | 3 |  |  | Да |  |
| Число случаев (на 1000 детей) | Число | 3 |  |  |  | Расчетный |
| Число дней, пропущенных по болезни (абсолютно) | Число | 3 |  |  |  |  |
| Число дней, пропущенных по болезни (на 1 ребенка) | Число | 3 |  |  |  | Расчетный |
| **В том числе** | | | | | | |
| Простудные заболевания (абсолютно) | Число | 3 |  |  |  |  |
| Простудные заболевания  (на 1000 детей) | Число | 3 |  |  |  | Расчетный |
| **Группы здоровья** | | | | | | |
| Первая | Число | 3 |  |  |  |  |
| Вторая | Число | 3 |  |  |  |  |
| Третья | Число | 3 |  |  |  |  |
| **Положительные туб. пробы** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| в том числе туб. вираж | Число | 3 |  |  |  |  |
| **Дефекты речи** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| **Нарушение осанки** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| **Сколиоз** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| **Отставание в физическом развитии** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| **Детей с хроническими заболеваниями** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| **Болезни кровообращения** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| ВПС | Число | 3 |  |  |  |  |
| Ревматизм | Число | 3 |  |  |  |  |
| Функц. шум | Число | 3 |  |  |  |  |
| **Болезни нервной системы и органов чувств** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| Заболевания ПНС | Число | 3 |  |  |  |  |
| Заболевания ЦНС | Число | 3 |  |  |  |  |
| Хронический отит | Число | 3 |  |  |  |  |
| Заболевания органов слуха | Число | 3 |  |  |  |  |
| Заболевания органов зрения | Число | 3 |  |  |  |  |
| **Болезни дыхательных путей** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| Болезни миндалин | Число | 3 |  |  |  |  |
| Хронический фарингит, синусит | Число | 3 |  |  |  |  |
| Хроническая пневмония | Число | 3 |  |  |  |  |
| Бронхиальная астма | Число | 3 |  |  |  |  |
| **Болезни органов пищеварения** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| Хронический гастрит | Число | 3 |  |  |  |  |
| Хронический гепатит | Число | 3 |  |  |  |  |
| ДЖВП | Число | 3 |  |  |  |  |
| **Заболевания мочеполовой системы** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| Хронический пиелонефрит | Число | 3 |  |  |  |  |
| **Эндокринные заболевания** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| Сахарный диабет | Число | 3 |  |  |  |  |
| **Заболевания крови** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| **Заболевания кожи** | | | | | | |
| Всего | Число | 3 |  |  |  |  |
| Атопический дерматит | Число | 3 |  |  |  |  |
| **Аллергические заболевания** | | | | | | |
| Всего | Число | 3 |  |  |  |  |

# *Приложение 3*

**Структура документа**

**«Отчет по профилактическим прививкам»**

| **Реквизит**  **документа** | **Тип**  **данных** | **Кол-во разрядов в**  **знаках** | **Формат**  **представления в документах** | **Допустимое**  **значение** | **Допустимость пустых значений (Да/Нет)** | **Примечание** |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование детского сада | Текст | 255 |  |  | Нет |  |
| Номер группы | Число | 1 |  |  | Нет |  |
| Кол-во детей в группе | Число | 2 |  |  | Нет |  |
| Наименование прививки | Текст | 255 |  |  | Нет |  |
| Тип прививки | Текст | 255 |  |  | Нет |  |
| Кол-во привитых детей в указанный период | Число | 2 |  | Меньше чем «Кол-во детей в группе» | Нет |  |
| Период начало | Дата | 8 | ДД.ММ.ГГ |  | Нет |  |
| Период конец | Дата | 8 | ДД.ММ.ГГ | Больше чем «Период начало» | Нет |  |

# *Приложение 4*

**Структура документа**

**«Отчет по дегельминтизации»**

| **Реквизит**  **документа** | **Тип**  **данных** | **Кол-во разрядов в**  **знаках** | **Формат**  **представления в документах** | **Допустимое**  **значение** | **Допустимость пустых значений (Да/Нет)** | **Примечание** |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование детского сада | Текст | 255 |  |  | Нет |  |
| Номер группы | Число | 1 |  |  | Нет |  |
| Кол-во детей в группе | Число | 2 |  |  | Нет |  |
| Всего проверено в указанный период | Число | 2 |  |  | Нет |  |
| Кол-во отрицательных результатов | Число | 2 |  | Отрицательно  +  Положительно  =  Всего | Нет |  |
| Кол-во положительных результатов | Число | 2 |  | Нет |  |
| Период начало | Дата | 8 | ДД.ММ.ГГ |  | Нет |  |
| Период конец | Дата | 8 | ДД.ММ.ГГ | Больше чем «Период начало» | Нет |  |

# *Приложение 5*

**Структура документа**

**«Туберкулинодиагностика»**

| **Реквизит**  **документа** | **Тип**  **данных** | **Кол-во разрядов в**  **знаках** | **Формат**  **представления в документах** | **Допустимое**  **значение** | **Допустимость пустых значений (Да/Нет)** | **Примечание** |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование детского сада | Текст | 255 |  |  | Нет |  |
| Номер группы | Число | 1 |  |  | Нет |  |
| Кол-во детей в группе | Число | 2 |  |  | Нет |  |
| Всего проверено в указанный период | Число | 2 |  |  | Нет |  |
| Кол-во отрицательных результатов | Число | 2 |  | Отрицательно  +  Сомнительно  +  Положительно  =  Всего | Нет |  |
| Кол-во сомнительных результатов | Число | 2 |  | Нет |  |
| Кол-во положительных результатов | Число | 2 |  | Нет |  |
| Период начало | Дата | 8 | ДД.ММ.ГГ |  | Нет |  |
| Период конец | Дата | 8 | ДД.ММ.ГГ | Больше чем «Период начало» | Нет |  |

# *Приложение 6*

**Полное дерево зависимостей пакетов Python,**

**использованных при разработке программного средства**

**anyio** 3.7.1 High level compatibility layer for multiple asynchronous event loop implementations

├── **idna** >=2.8

└── **sniffio** >=1.1

**bcrypt** 4.1.3 Modern password hashing for your software and your servers

**cffi** 1.16.0 Foreign Function Interface for Python calling C code.

└── **pycparser** \*

**click** 8.1.7 Composable command line interface toolkit

└── **colorama** \*

**cryptography** 40.0.2 cryptography is a package which provides cryptographic recipes and primitives to Python developers.

└── **cffi** >=1.12

└── pycparser \*

**ecdsa** 0.18.0 ECDSA cryptographic signature library (pure python)

└── **six** >=1.9.0

**envparse** 0.2.0 Simple environment variable parsing

**fastapi** 0.95.1 FastAPI framework, high performance, easy to learn, fast to code, ready for production

├── **pydantic** >=1.6.2,<1.7 || >1.7,<1.7.1 || >1.7.1,<1.7.2 || >1.7.2,<1.7.3 || >1.7.3,<1.8 || >1.8,<1.8.1 || >1.8.1,<2.0.0

│ └── **typing-extensions** >=4.2.0

└── **starlette** >=0.26.1,<0.27.0

└── **anyio** >=3.4.0,<5

├── **idna** >=2.8

└── **sniffio** >=1.1

**h11** 0.14.0 A pure-Python, bring-your-own-I/O implementation of HTTP/1.1

**idna** 3.7 Internationalized Domain Names in Applications (IDNA)

**jinja2** 3.1.4 A very fast and expressive template engine.

└── **markupsafe** >=2.0

**markupsafe** 2.1.5 Safely add untrusted strings to HTML/XML markup.

**openpyxl** 3.1.2 A Python library to read/write Excel 2010 xlsx/xlsm files

└── **et-xmlfile** \*

**psycopg2-binary** 2.9.9 psycopg2 - Python-PostgreSQL Database Adapter

**pyasn1** 0.4.8 ASN.1 types and codecs

**pycparser** 2.22 C parser in Python

**pydantic** 1.10.15 Data validation and settings management using python type hints

└── **typing-extensions** >=4.2.0

**pytest** 8.2.0 pytest: simple powerful testing with Python

├── **colorama** \*

├── **iniconfig** \*

├── **packaging** \*

└── **pluggy** >=1.5,<2.0

**pytest-mock** 3.14.0 Thin-wrapper around the mock package for easier use with pytest

└── **pytest** >=6.2.5

├── **colorama** \*

├── **iniconfig** \*

├── **packaging** \*

└── **pluggy** >=1.5,<2.0

**python-dotenv** 1.0.1 Read key-value pairs from a .env file and set them as environment **variables**

**python-jose** 3.3.0 JOSE implementation in Python

├── **ecdsa** !=0.15

│ └── **six** >=1.9.0

├── **pyasn1** \*

└── **rsa** \*

└── **pyasn1** >=0.1.3

**python-multipart** 0.0.6 A streaming multipart parser for Python

**rsa** 4.9 Pure-Python RSA implementation

└── **pyasn1** >=0.1.3

**six** 1.16.0 Python 2 and 3 compatibility utilities

**sniffio** 1.3.1 Sniff out which async library your code is running under

**sqlalchemy** 2.0.30 Database Abstraction Library

├── **greenlet** !=0.4.17

└── **typing-extensions** >=4.6.0

**starlette** 0.26.1 The little ASGI library that shines.

└── **anyio** >=3.4.0,<5

├── **idna** >=2.8

└── **sniffio** >=1.1

**typing-extensions** 4.11.0 Backported and Experimental Type Hints for Python 3.8+

**uvicorn** 0.21.1 The lightning-fast ASGI server.

├── **click** >=7.0

│ └── **colorama** \*

└── **h11** >=0.8

# *Приложение 7*

**Документирование доменов**

| **Название**  **домена** | **Описание**  **домена** | **Название**  **атрибута** | **Название типа**  **сущности** | **Набор**  **допустимых**  **значений** | **Размер и формат** |
| --- | --- | --- | --- | --- | --- |
| Пол ребенка | Содержит два возможных значения половой принадлежности | Пол | Ребенок | М, Ж | Один символ, буквенный, буква заглавная |
| Номер группы в детском саду | Содержит возможные номера группы в детском саду  (от 1 до 6) | Номер группы в детском саду | Ребенок | 1-6 | Одна цифра |
| Характеристика образовательного учреждения | Содержит единственно-возможный вариант «ДДУ» | Характеристика образовательного учреждения | Ребенок | ДДУ | Значение «ДДУ» |
| Характеристика семьи | Содержит две возможные характеристики семьи | Характеристика семьи | Ребенок | 1. Полная  2. Неполная | Одно слово, первая буква заглавная |
| Микроклимат в семье | Содержит два возможных варианта микроклимата в семье | Микроклимат в семье | Ребенок | 1. Благоприятный  2. Неблагоприятный | Одно слово, первая буква заглавная |
| Наличие у ребенка места для отдыха и занятий | Содержит возможные варианты мест для отдыха и занятий ребенка | Наличие у ребенка места для отдыха и занятий | Ребенок | 1. Комната  2. Индивидуальный стол  3. Нет | Символьная строка, первая буква заглавная |
| Тип родства | Содержит два возможных типа родства родителей (опекунов) | Тип родства | Родитель | 1. Отец  2. Мать | Одно слово, первая буква заглавная |
| Образование | Содержит типы образования родителей | Образование | Родитель | 1. б/обр.  2. н/ср.  3. ср.  4. ср.-спец.  5. н/высш.  6. высш. | Символьная строка, все буквы строчные |
| Номер телефона | Содержит одиннадцатизначный номер телефона | Номер телефона | Родитель | 80000000000 - 89999999999 | Одиннадцатизначное число, начинается с цифры 8 |
| Тип аллергии | Содержит наименование типа аллергии | Тип | Аллергия | 1. Вакцинальная  2. Лекарственная  3. Аллергические заболевания | Символьная строка, первая буква заглавная |
| Тип прививки | Содержит наименование типа прививки | Тип прививки | Прививка | 1. Вакцинация I  2. Вакцинация II  3. Вакцинация III  4. Ревакцинация I  5. Ревакцинация II  6. Ревакцинация III  7. Ревакцинация IV | Символьная строка, первая буква заглавная |
| Реакция на прививку | Содержит результат реакции на прививку | Реакция | Прививка,  Введение гамма-глобулина | 1. Немедленная  2. Замедленная | Одно слово, первая буква заглавная |
| Возрастной период проведения медосмотра | Содержит наименование периода проведения медосмотра | Период | Медосмотр | 1. Перед поступ-лением в ясли - сад, детский сад  2. За 1 год до школы  3. Перед школой | Символьная строка, первая буква заглавная |
| Группа здоровья | Содержит перечень групп здоровья | Группа здоровья | Медосмотр | I - IV | Одна римская цифра от I до IV |
| Мед. гр. для занятий физкультурой | Содержит перечень мед. гр. для занятий физкультурой | Мед. гр. для занятий физкультурой | Медосмотр | 1. Основная  2. Лечебная | Одно слово, первая буква заглавная |
| Результаты анкетного теста | Содержит результаты анкетного теста | Анкетный тест | Скрининг | 1. Норма  2. Отклонение | Одно слово, первая буква заглавная |
| Оценка физического развития | Содержит оценку физического развития | Физическое  развитие | Скрининг | 1. Нормальное  2. Низкий рост  3. Дефицит массы  4. Избыток массы | Символьная строка, первая буква заглавная |
| Тип осанки | Содержит три типа осанки | Осанка | Скрининг | 1. Нормальная  2. Незначительные отклонения  3. Значительные нарушения | Символьная строка, первая буква заглавная |
| Состояние стопы | Содержит три типа состояния стопы | Состояние стопы | Скрининг | 1. Нормальная  2. Уплощена  3. Плоская | Одно слово, первая буква заглавная |
| Острота зрения | Содержит два типа остроты зрения | Острота зрения | Скрининг | 1. Нормальная  2. Снижена | Одно слово, первая буква заглавная |
| Результаты теста Малиновского | Содержит результаты теста Малиновского | Тест Малиновского | Скрининг | 1. Нормальная  2. Предмиопия | Одно слово, первая буква заглавная |
| Бинокулярное зрение | Содержит два типа бинокулярного зрения | Бинокулярное зрение | Скрининг | 1. Норма  2. Нарушение | Одно слово, первая буква заглавная |
| Оценка остроты слуха | Содержит оценку остроты слуха | Острота слуха | Скрининг | 1. Норма  2. Снижена | Одно слово, первая буква заглавная |
| Оценка физической подготовленности | Содержит оценку физической подготовленности | Физическая  подготовленность | Скрининг | 1. Норма  2. Снижена  3. Повышена | Одно слово, первая буква заглавная |
| Определение белка в моче | Содержит результат анализа мочи на белок | Определение белка в моче | Скрининг | 1. Норма  2. Следы белка  3. Белок в моче | Символьная строка, первая буква заглавная |
| Определение глюкозы в моче | Содержит результат анализа мочи на глюкозу | Определение глюкозы в моче | Скрининг | 1. Норма  2. Глюкоза в моче | Символьная строка, первая буква заглавная |
| Биологический возраст | Содержит результат сравнения биологического возраста с проведенными тестами | Биологический возраст | Скрининг | 1. Соответствует  2. Опережает  3. Отстает | Одно слово, первая буква заглавная |
| Выявление невротических расстройств | Содержит заключение по выявлению невротических расстройств | Выявление невротических расстройств | Скрининг | 1. Норма  2. Отклонение | Одно слово, первая буква заглавная |
| Мышление и речь | Содержит заключение по выявлению дефектов мышления и речи | Мышление и речь | Скрининг |
| Моторное развитие | Содержит заключение по выявлению отклонений моторного развития | Моторное развитие | Скрининг |
| Внимание и память | Содержит заключение по выявлению отклонений внимания и памяти | Внимание и память | Скрининг |  |  |
| Социальные контакты | Содержит заключение по выявлению социальных контактов | Социальные контакты | Скрининг |

# *Приложение 8*

**SQL-код для определения таблиц в базе данных medical\_record**

CREATE TABLE users

(

kindergarten\_num INTEGER NOT NULL,

kindergarten\_name TEXT (200) NOT NULL UNIQUE,

surname TEXT (200) NOT NULL,

name TEXT (200) NOT NULL,

patronymic TEXT (200) NOT NULL,

password\_hash TEXT (255) NOT NULL,

PRIMARY KEY (kindergarten\_num)

);

CREATE TABLE parents

(

id INTEGER NOT NULL,

surname INTEGER (200) NOT NULL,

name TEXT (200) NOT NULL,

patronymic INTEGER (200) NOT NULL,

birthday\_year INTEGER NOT NULL,

education TEXT (10) NOT NULL

CHECK (education = 'б/обр.' OR

education = 'н/ср.' OR

education = 'ср.' OR

education = 'ср.-спец.' OR

education = 'н/высш.' OR

education = 'высш.'),

phone\_num INTEGER NOT NULL

CHECK (phone\_num BETWEEN 80000000000 AND 89999999999),

PRIMARY KEY (id)

);

CREATE TABLE clinics (

id INTEGER PRIMARY KEY NOT NULL,

name TEXT NOT NULL UNIQUE

);

CREATE TABLE children

(

medcard\_num INTEGER NOT NULL,

surname TEXT (200) NOT NULL,

name TEXT (200) NOT NULL,

patronymic TEXT (200) NOT NULL,

kindergarten\_num INTEGER NOT NULL

REFERENCES users (kindergarten\_num) ON DELETE RESTRICT

ON UPDATE CASCADE,

birthday TEXT (10) NOT NULL,

sex TEXT (1) NOT NULL

CHECK (sex = 'М' OR

sex = 'Ж'),

group\_num INTEGER NOT NULL

CHECK (group\_num BETWEEN 1 AND 6),

address TEXT (300) NOT NULL,

clinic\_id INTEGER NOT NULL

REFERENCES clinics (id) ON DELETE RESTRICT

ON UPDATE CASCADE,

edu\_type TEXT (30) DEFAULT 'ДДУ'

NOT NULL,

entering\_date TEXT (10) NOT NULL,

father\_id INTEGER REFERENCES parents (id) ON DELETE SET NULL,

mother\_id INTEGER REFERENCES parents (id) ON DELETE SET NULL,

family\_characteristics TEXT (20) NOT NULL

CHECK (family\_characteristics = 'Полная' OR

family\_characteristics = 'Неполная'),

family\_microclimate TEXT (25) NOT NULL

CHECK (family\_microclimate = 'Благоприятный' OR

family\_microclimate = 'Неблагоприятный'),

rest\_and\_class\_opportunities TEXT (30) NOT NULL

CHECK (rest\_and\_class\_opportunities = 'Комната' OR

rest\_and\_class\_opportunities = 'Индивидуальный стол' OR

rest\_and\_class\_opportunities = 'Нет'),

case\_history TEXT,

PRIMARY KEY (medcard\_num)

);

CREATE TABLE medical\_certificates

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

disease TEXT (300) NOT NULL,

cert\_date TEXT NOT NULL,

start\_date TEXT NOT NULL,

end\_date TEXT NOT NULL,

infection\_contact INTEGER (1) NOT NULL,

sport\_exemption\_date TEXT,

vac\_exemption\_date TEXT,

doctor TEXT (250) NOT NULL,

PRIMARY KEY (

medcard\_num,

disease,

cert\_date

)

);

CREATE TABLE allergyes

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE,

allergen TEXT (200) NOT NULL,

allergy\_type TEXT (35) NOT NULL

CHECK (allergy\_type = 'Вакцинальная' OR

allergy\_type = 'Лекарственная' OR

allergy\_type = 'Аллергические заболевания'),

start\_age INTEGER NOT NULL,

reaction\_type TEXT (200) NOT NULL,

diagnosis\_date TEXT NOT NULL,

note TEXT,

PRIMARY KEY (

medcard\_num,

allergen

)

);

CREATE TABLE extra\_classes

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

classes\_type TEXT (200) NOT NULL,

age INTEGER NOT NULL,

hours\_on\_week INTEGER NOT NULL,

PRIMARY KEY (

medcard\_num,

classes\_type,

age

)

);

CREATE TABLE hospitalizations

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

start\_date TEXT (10) NOT NULL,

end\_date TEXT,

diagnosis TEXT NOT NULL,

founding TEXT (255) NOT NULL,

PRIMARY KEY (

medcard\_num,

start\_date

)

);

CREATE TABLE spa\_treatments

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

start\_date TEXT (10) NOT NULL,

end\_date TEXT (10),

diagnosis TEXT NOT NULL,

founding\_specialization TEXT (255) NOT NULL,

climatic\_zone TEXT (255) NOT NULL,

PRIMARY KEY (

medcard\_num,

start\_date));

CREATE TABLE dispensaryes

(

id INTEGER PRIMARY KEY AUTOINCREMENT

NOT NULL,

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

start\_date TEXT (10) NOT NULL,

diagnosis TEXT NOT NULL,

specialist TEXT (255),

end\_date TEXT (10),

end\_reason TEXT

);

CREATE TABLE visit\_specialist\_controls

(

dispensary\_id INTEGER NOT NULL

REFERENCES dispensaryes (id) ON DELETE CASCADE

ON UPDATE CASCADE,

assigned\_date DATE NOT NULL,

fact\_date DATE,

PRIMARY KEY (

dispensary\_id,

assigned\_date

)

);

CREATE TABLE dewormings

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

deworming\_date TEXT (10) NOT NULL,

result TEXT NOT NULL,

PRIMARY KEY (

medcard\_num,

deworming\_date

)

);

CREATE TABLE oral\_sanations

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

sanation\_date TEXT (10) NOT NULL,

dental\_result TEXT NOT NULL,

sanation\_result TEXT NOT NULL,

PRIMARY KEY (

medcard\_num,

sanation\_date

)

);

CREATE TABLE vac\_names

(

id INTEGER NOT NULL,

name TEXT (200) NOT NULL,

vac\_type TEXT (16) CHECK (vac\_type = 'Профилактическая' OR

vac\_type = 'По показаниям')

NOT NULL,

PRIMARY KEY (

id

)

);

CREATE TABLE prevaccination\_checkups

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

examination\_date TEXT (10) NOT NULL,

age INTEGER (2),

diagnosis TEXT NOT NULL,

report TEXT NOT NULL,

vac\_name\_id INTEGER NOT NULL

REFERENCES vac\_names (id) ON DELETE RESTRICT,

no\_vac\_date TEXT (10),

doctor TEXT (255) NOT NULL,

PRIMARY KEY (

medcard\_num,

examination\_date

)

);

CREATE TABLE vaccinations

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

vac\_name\_id INTEGER NOT NULL

REFERENCES vac\_names (id) ON DELETE RESTRICT,

vac\_type TEXT (30) NOT NULL

CHECK (vac\_type REGEXP '^Вакцинация ?I{1,3}$' OR

vac\_type REGEXP '^Ревакцинация ?I{1,3}?V{0,1}$'),

vac\_date TEXT (10) NOT NULL,

serial TEXT (100) NOT NULL,

dose NUMERIC (6, 3) NOT NULL,

introduction\_method TEXT (100) NOT NULL,

reaction TEXT (22) NOT NULL

CHECK (reaction REGEXP '^(Не|За)медленная$'),

doctor TEXT (250) NOT NULL,

PRIMARY KEY (

medcard\_num,

vac\_name\_id,

vac\_type

));

CREATE TABLE tuberculosis\_vaccinations

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

vac\_date TEXT (10) NOT NULL,

serial TEXT (100) NOT NULL,

dose NUMERIC (6, 3) NOT NULL,

doctor TEXT (250) NOT NULL,

PRIMARY KEY (

medcard\_num,

vac\_date

)

);

CREATE TABLE gamma\_globulin\_injections

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

vac\_date TEXT (10) NOT NULL,

reason TEXT NOT NULL,

serial TEXT (100) NOT NULL,

dose NUMERIC (6, 3) NOT NULL,

reaction TEXT (22) NOT NULL

CHECK (reaction REGEXP '^(Не|За)медленная$'),

doctor TEXT (250) NOT NULL,

PRIMARY KEY (

medcard\_num,

vac\_date

)

);

CREATE TABLE mantoux\_tests

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

check\_date TEXT (10) NOT NULL,

result TEXT NOT NULL,

PRIMARY KEY (

medcard\_num,

check\_date

)

);

CREATE TABLE medical\_examinations

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

period TEXT (80) NOT NULL

CHECK (period = 'Перед поступлением в ясли-сад, детский сад' OR

period = 'За 1 год до школы' OR

period = 'Перед школой'),

examination\_date TEXT (10) NOT NULL,

age INTEGER,

height NUMERIC (5, 2) NOT NULL,

weight NUMERIC (5, 2) NOT NULL,

complaints TEXT,

pediatrician TEXT,

orthopaedist TEXT,

ophthalmologist TEXT,

otolaryngologist TEXT,

dermatologist TEXT,

neurologist TEXT,

speech\_therapist TEXT,

denta\_surgeon TEXT,

psychologist TEXT,

other\_doctors TEXT,

blood\_test TEXT,

urine\_analysis TEXT,

feces\_analysis TEXT,

general\_diagnosis TEXT,

physical\_development TEXT,

mental\_development TEXT,

health\_group TEXT (3) CHECK (health\_group REGEXP '^(I{1,3})$|^(IV)'),

sport\_group TEXT (16) CHECK (sport\_group = 'Основная' OR

sport\_group = 'Лечебная'),

med\_and\_ped\_conclusion TEXT,

recommendations TEXT,

PRIMARY KEY (

medcard\_num,

period

)

);

CREATE TABLE ongoing\_medical\_supervisions

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

examination\_date TEXT (10) NOT NULL,

examination\_data TEXT NOT NULL,

diagnosis TEXT NOT NULL,

prescription TEXT,

doctor TEXT (250) NOT NULL,

PRIMARY KEY (

medcard\_num,

examination\_date

)

);

CREATE TABLE screenings

(

medcard\_num INTEGER NOT NULL

REFERENCES children (medcard\_num) ON DELETE CASCADE

ON UPDATE CASCADE,

examination\_date TEXT NOT NULL,

age INTEGER (2),

questionnaire\_test TEXT (20) NOT NULL

CHECK (questionnaire\_test = 'Норма' OR

questionnaire\_test = 'Отклонение'),

height NUMERIC (5, 2) NOT NULL,

weight NUMERIC (5, 2) NOT NULL,

physical\_development TEXT (30) CHECK (physical\_development = 'Нормальное' OR

physical\_development = 'Низкий рост' OR

physical\_development = 'Дефицит массы' OR

physical\_development = 'Избыток массы'),

blood\_pressures TEXT (30),

carriage TEXT (50) CHECK (carriage = 'Нормальная' OR

carriage = 'Незначительные отклонения' OR

carriage = 'Значительные нарушения'),

foot\_condition TEXT (20) CHECK (foot\_condition = 'Нормальная' OR

foot\_condition = 'Уплощена' OR

foot\_condition = 'Плоская'),

sight\_od NUMERIC (3, 2),

sight\_os NUMERIC (3, 2),

visual\_acuity TEXT (20) CHECK (visual\_acuity = 'Нормальная' OR

visual\_acuity = 'Снижена'),

malinovsky\_test TEXT (20) CHECK (malinovsky\_test = 'Нормальная' OR

malinovsky\_test = 'Предмиопия'),

binocular\_vision TEXT (20) CHECK (binocular\_vision = 'Норма' OR

binocular\_vision = 'Нарушение'),

hearing\_acuteness TEXT (20) CHECK (hearing\_acuteness = 'Норма' OR

hearing\_acuteness = 'Снижена'),

dynammetry\_left NUMERIC (3, 1),

dynammetry\_right NUMERIC (3, 1),

physical\_fitness TEXT (20) CHECK (physical\_fitness = 'Норма' OR

physical\_fitness = 'Снижена' OR

physical\_fitness = 'Повышена'),

protein\_in\_urine TEXT (25) CHECK (protein\_in\_urine = 'Норма' OR

protein\_in\_urine = 'Следы белка' OR

protein\_in\_urine = 'Белок в моче'),

glucose\_in\_urine TEXT (30) CHECK (glucose\_in\_urine = 'Норма' OR

glucose\_in\_urine = 'Глюкоза в моче'),

biological\_age TEXT (30) CHECK (biological\_age = 'Соответствует' OR

biological\_age = 'Опережает' OR

biological\_age = 'Отстает'),

speech\_defects INTEGER (1),

kern\_test INTEGER (1),

neurotic\_disorders INTEGER (1),

thinking\_and\_speech\_disorders INTEGER (1),

motor\_development\_disorders INTEGER (1),

attention\_and\_memory\_disorders INTEGER (1),

social\_contacts\_disorders INTEGER (1),

diseases\_for\_year INTEGER (3),

PRIMARY KEY (

medcard\_num,

examination\_date

)

);

**SQL-код для определения представлений в базе данных medical\_record**

Основные сведения о детях, включающие в себя: фамилию, имя, отчество, дату рождения, наименование детского сада, номер группы, а также сведения о родителях:

CREATE VIEW children\_with\_parents\_view AS

SELECT c.medcard\_num,

c.surname,

c.name,

c.patronymic,

c.kindergarten\_num,

c.birthday,

c.sex,

c.group\_num,

c.address,

c.clinic\_id,

cl.name AS clinic\_name,

c.edu\_type,

c.entering\_date,

f.surname AS father\_surname,

f.name AS father\_name,

f.patronymic AS father\_patronymic,

f.birthday\_year AS father\_birthday\_year,

f.education AS father\_education,

f.phone\_num AS father\_phone\_num,

m.surname AS mother\_surname,

m.name AS mother\_name,

m.patronymic AS mother\_patronymic,

m.birthday\_year AS mother\_birthday\_year,

m.education AS mother\_education,

m.phone\_num AS mother\_phone\_num,

c.family\_characteristics,

c.family\_microclimate,

c.rest\_and\_class\_opportunities,

c.case\_history

FROM children AS c

LEFT JOIN

parents AS f ON c.father\_id = f.id

LEFT JOIN

parents AS m ON c.mother\_id = m.id

LEFT JOIN

clinics AS cl ON c.clinic\_id = cl.id;

FROM childrens AS c

JOIN kindergartens AS k ON c.kindergarten\_num = k.number

ORDER BY 5,7,2,3,4;

Перечень всех профилактических прививок:

CREATE VIEW prof\_vaccinations AS

SELECT v.medcard\_num,

v.vac\_name\_id,

n.name AS vac\_name,

v.vac\_type,

v.vac\_date,

v.serial,

v.dose,

v.introduction\_method,

v.reaction,

v.doctor

FROM vaccinations AS v

JOIN

vac\_names AS n ON v.vac\_name\_id = n.id

WHERE n.vac\_type = 'Профилактическая';

Перечень всех прививок по эпидемиологическим показаниям:

CREATE VIEW epid\_vaccinations AS

SELECT v.medcard\_num,

v.vac\_name\_id,

n.name AS vac\_name,

v.vac\_type,

v.vac\_date,

v.serial,

v.dose,

v.introduction\_method,

v.reaction,

v.doctor

FROM vaccinations AS v

JOIN

vac\_names AS n ON v.vac\_name\_id = n.id

WHERE n.vac\_type = 'По показаниям';

Пропуск занятий по болезни:

CREATE VIEW skiping\_by\_disease AS

SELECT medcard\_num,

start\_date,

end\_date,

disease

FROM medical\_certificates

WHERE start\_date NOT NULL;

# *Приложение 9*

**SQL-код триггера для вставки возраста в поле «age»**

AFTER INSERT

ON prevaccination\_checkups

FOR EACH ROW

BEGIN

UPDATE prevaccination\_checkups

SET age = date(examination\_date, 'start of year') - date( (

SELECT c.birthday

FROM children AS c

WHERE c.medcard\_num = medcard\_num

), 'start of year');

END;

**SQL-код триггера для подсчета количества заболеваний за год**

UPDATE screenings

SET diseases\_for\_year = (

SELECT COUNT(\*)

FROM medical\_certificates AS mc

WHERE mc.medcard\_num = screenings.medcard\_num AND

date(mc.start\_date) >= date(screenings.examination\_date, '-1 year') AND

date(mc.start\_date) <= date(screenings.examination\_date)

);

# *Приложение 10*

**Код модуля импорта данных из файла**

import re

from fastapi import HTTPException

from openpyxl import load\_workbook

from openpyxl.worksheet.worksheet import Worksheet

from models.base\_table\_name import TableBaseModel

from services.catalogs.clinic import ClinicService

from services.medical\_record.dispensary import DispensaryService

from services.medical\_record.medical\_record import MedicalRecordService

from services.medical\_record.parent import ParentService

from services.medical\_record.visit\_specialist\_control import VisitSpecialistControlService

from services.medical\_record.allergy import AllergyService

from services.medical\_record.extra\_class import ExtraClassService

from services.medical\_record.past\_illness import PastIllnessService

from services.medical\_record.hospitalization import HospitalizationService

from services.medical\_record.spa\_treatment import SpaTreatmentService

from services.medical\_record.medical\_certificate import MedicalCertificateService

from services.medical\_record.deworming import DewormingService

from services.medical\_record.oral\_sanation import OralSanationService

from services.medical\_record.prevaccination\_checkup import PrevaccinationCheckupService

from services.medical\_record.vaccination import VaccinationService

from services.medical\_record.gg\_injection import GammaGlobulinInjectionService

from services.medical\_record.mantoux\_test import MantouxTestService

from services.medical\_record.tub\_vac import TuberculosisVaccinationService

from services.medical\_record.medical\_examination import MedicalExaminationService

from services.medical\_record.ongoing\_medical\_supervision import OngoingMedicalSupervisionService

from services.medical\_record.screening import ScreeningService

from services.catalogs.vac\_name import VacNameService

from models.catalogs.clinic import ClinicCreate

from models.medical\_record.child import ChildCreate

from models.medical\_record.dispensary import DispensaryCreate

from models.medical\_record.parent import ParentCreate, ParentType

from models.medical\_record.visit\_specialist\_control import VisitSpecialistControlCreate

from models.medical\_record.allergy import AllergyCreate

from models.medical\_record.deworming import DewormingCreate

from models.medical\_record.extra\_class import ExtraClassCreate

from models.medical\_record.gg\_injection import GammaGlobulinInjectionCreate

from models.medical\_record.hospitalization import HospitalizationCreate

from models.medical\_record.mantoux\_test import MantouxTestCreate

from models.medical\_record.medical\_certificate import MedicalCertificateCreate

from models.medical\_record.medical\_examination import MedicalExaminationCreate

from models.medical\_record.ongoing\_medical\_supervision import OngoingMedicalSupervisionCreate

from models.medical\_record.oral\_sanation import OralSanationCreate

from models.medical\_record.past\_illness import PastIllnessCreate

from models.medical\_record.prevaccination\_checkup import PrevaccinationCheckupCreate

from models.medical\_record.screening import ScreeningCreate

from models.medical\_record.spa\_treatment import SpaTreatmentCreate

from models.medical\_record.tub\_vac import TuberculosisVaccinationCreate

from models.catalogs.vac\_name import VacNameCreate

from models.medical\_record.vaccination import VaccinationCreate

def create\_obj\_generator\_from\_ws(ws: Worksheet, tab\_name: str):

enum\_table = {

"TableBaseModel": TableBaseModel

}

obj\_gen = ws.values

headers = next(obj\_gen)

for data in obj\_gen:

data\_as\_dict = dict(zip(headers, data))

obj = enum\_table["TableBaseModel"][tab\_name].value(\*\*data\_as\_dict)

yield obj

def camel\_to\_snake(camel\_case\_string):

parts = re.findall(r'[A-Z][a-z0-9]\*', camel\_case\_string)

return '\_'.join(part.lower() for part in parts)

def import\_from\_xlsx(filename: str,

kindergarten\_num: int,

medcard\_service: MedicalRecordService,

clinic\_service: ClinicService,

parent\_service: ParentService,

dispensary\_service: DispensaryService,

visit\_s\_control\_service: VisitSpecialistControlService,

allergy\_service: AllergyService,

extra\_class\_service: ExtraClassService,

past\_illness\_service: PastIllnessService,

hospitalization\_service: HospitalizationService,

spa\_treatment\_service: SpaTreatmentService,

medical\_certificate\_service: MedicalCertificateService,

deworming\_service: DewormingService,

oral\_sanation\_service: OralSanationService,

prevaccination\_checkup\_service: PrevaccinationCheckupService,

vaccination\_service: VaccinationService,

gg\_injection\_service: GammaGlobulinInjectionService,

mantoux\_test\_service: MantouxTestService,

tub\_vac\_service: TuberculosisVaccinationService,

medical\_examination\_service: MedicalExaminationService,

ongoing\_medical\_supervision\_service: OngoingMedicalSupervisionService,

screening\_service: ScreeningService,

vac\_name\_service: VacNameService,

):

wb = load\_workbook(filename=filename)

service\_tabs = ['Child', 'Clinic', 'Parent', 'VacName', 'Dispensary', 'VisitSpecialistControl']

try:

ws = wb["Child"]

except KeyError:

raise HTTPException(

status\_code=422,

detail="File has no sheet 'Child'"

)

child\_gen = create\_obj\_generator\_from\_ws(ws=ws, tab\_name="Child")

child = next(child\_gen)

child.kindergarten\_num = kindergarten\_num

# Clinic

try:

ws = wb["Clinic"]

clinics\_gen = create\_obj\_generator\_from\_ws(ws=ws, tab\_name="Clinic")

clinic = next(clinics\_gen)

db\_clinic = clinic\_service.get\_clinic\_by\_name(name=clinic.name)

if db\_clinic:

child.clinic\_id = db\_clinic.id

else:

clinic = clinic\_service.add\_new\_clinic(clinic\_data=ClinicCreate(name=clinic.name))

child.clinic\_id = clinic.id

except KeyError:

pass

child.medcard\_num = medcard\_service.add\_new\_medcard(child\_data=ChildCreate(\*\*child.dict())).medcard\_num

# Parent

try:

ws = wb["Parent"]

parent\_gen = create\_obj\_generator\_from\_ws(ws=ws, tab\_name="Parent")

if child.father\_id:

father = next(parent\_gen)

db\_father = parent\_service.get\_paren\_by\_full\_data(parent\_data=father)

if db\_father:

child.father\_id = parent\_service.add\_parent\_to\_medcard(medcard\_num=child.medcard\_num,

parent\_id=db\_father.id,

parent\_type=ParentType.FATHER)

else:

father = parent\_service.add\_new\_parent(medcard\_num=child.medcard\_num, parent\_data=ParentCreate(\*\*father.dict(), parent\_type='father'))

child.father\_id = father.id

if child.mother\_id:

mother = next(parent\_gen)

db\_mother = parent\_service.get\_paren\_by\_full\_data(parent\_data=mother)

if db\_mother:

child.mother\_id = parent\_service.add\_parent\_to\_medcard(medcard\_num=child.medcard\_num,

parent\_id=db\_mother.id,

parent\_type=ParentType.MOTHER)

else:

mother = parent\_service.add\_new\_parent(medcard\_num=child.medcard\_num, parent\_data=ParentCreate(\*\*mother.dict(), parent\_type='mother'))

child.mother\_id = mother.id

except KeyError:

pass

# Dispensary

try:

visit\_ws = wb["VisitSpecialistControl"]

visit\_s\_control\_gen = create\_obj\_generator\_from\_ws(ws=visit\_ws, tab\_name="VisitSpecialistControl")

visit\_s\_control\_list = list(visit\_s\_control\_gen)

except KeyError:

pass

try:

ws = wb["Dispensary"]

dispensaryes\_gen = create\_obj\_generator\_from\_ws(ws=ws, tab\_name="Dispensary")

for dispensary in dispensaryes\_gen:

old\_dispensary\_id = dispensary.id

dispensary.medcard\_num = child.medcard\_num

dispensary = dispensary\_service.add\_new\_dispensary(dispensary\_data=DispensaryCreate(\*\*dispensary.dict()))

for visit\_s\_control in visit\_s\_control\_list[:]:

if visit\_s\_control.dispensary\_id == old\_dispensary\_id:

visit\_s\_control.dispensary\_id = dispensary.id

visit\_s\_control\_service.add\_new\_visit\_specialist\_control(visit\_specialist\_control\_data=VisitSpecialistControlCreate(\*\*visit\_s\_control.dict()))

visit\_s\_control\_list.remove(visit\_s\_control)

except KeyError:

pass

services = {

"Child": medcard\_service,

"Allergy": allergy\_service,

"Clinic": clinic\_service,

"Deworming": deworming\_service,

"Dispensary": dispensary\_service,

"ExtraClass": extra\_class\_service,

"GammaGlobulinInjection": gg\_injection\_service,

"Hospitalization": hospitalization\_service,

"MantouxTest": mantoux\_test\_service,

"MedicalCertificate": medical\_certificate\_service,

"MedicalExamination": medical\_examination\_service,

"OngoingMedicalSupervision": ongoing\_medical\_supervision\_service,

"OralSanation": oral\_sanation\_service,

"Parent": parent\_service,

"PastIllness": past\_illness\_service,

"PrevaccinationCheckup": prevaccination\_checkup\_service,

"Screening": screening\_service,

"SpaTreatment": spa\_treatment\_service,

"TuberculosisVaccination": tub\_vac\_service,

"VacName": vac\_name\_service,

"Vaccination": vaccination\_service,

"VisitSpecialistControl": visit\_s\_control\_service

}

create\_model = {

"Child": ChildCreate,

"Allergy": AllergyCreate,

"Clinic": ClinicCreate,

"Deworming": DewormingCreate,

"Dispensary": DispensaryCreate,

"ExtraClass": ExtraClassCreate,

"GammaGlobulinInjection": GammaGlobulinInjectionCreate,

"Hospitalization": HospitalizationCreate,

"MantouxTest": MantouxTestCreate,

"MedicalCertificate": MedicalCertificateCreate,

"MedicalExamination": MedicalExaminationCreate,

"OngoingMedicalSupervision": OngoingMedicalSupervisionCreate,

"OralSanation": OralSanationCreate,

"Parent": ParentCreate,

"PastIllness": PastIllnessCreate,

"PrevaccinationCheckup": PrevaccinationCheckupCreate,

"Screening": ScreeningCreate,

"SpaTreatment": SpaTreatmentCreate,

"TuberculosisVaccination": TuberculosisVaccinationCreate,

"VacName": VacNameCreate,

"Vaccination": VaccinationCreate,

"VisitSpecialistControl": VisitSpecialistControlCreate

}

for tab\_name in wb.sheetnames:

if tab\_name not in service\_tabs:

ws = wb[tab\_name]

service = services[tab\_name]

method\_suffix = camel\_to\_snake(tab\_name)

add\_method\_name = f"add\_new\_{method\_suffix}"

table = create\_obj\_generator\_from\_ws(ws=ws, tab\_name=tab\_name)

if hasattr(service, add\_method\_name):

add\_method = getattr(service, add\_method\_name)

try:

for table\_row in table:

table\_row.medcard\_num = child.medcard\_num

add\_method(create\_model[tab\_name](\*\*table\_row.dict()))

except AttributeError:

print(f"Модель с именем {tab\_name} не найдена")

# *Приложение 11*

**Код функции формирования годового отчета**

def get\_annual\_report\_by\_age\_type(self, age\_type: AgeType, year: int):

report\_date = date(year, 12, 31)

### AgeType.FIRST

min\_age = 0

max\_age = 3

max\_diseases\_for\_year = 5

if age\_type == AgeType.SECOND:

min\_age = 4

max\_age = 5

max\_diseases\_for\_year = 4

elif age\_type == AgeType.THIRD:

min\_age = 6

max\_age = 9

max\_diseases\_for\_year = 3

report = AnnualReport(year=year, age\_type=age\_type)

children = (

self.session

.query(Child)

.filter(extract('year', report\_date) - extract('year', Child.birthday) >= min\_age)

.filter(extract('year', report\_date) - extract('year', Child.birthday) <= max\_age)

.order\_by(Child.surname)

.all()

)

if not children:

return report

for child in children:

### Diseases

medical\_certificates = (

self.session

.query(MedicalCertificate)

.filter(MedicalCertificate.medcard\_num == child.medcard\_num)

.filter(extract('year', report\_date) == extract('year', MedicalCertificate.start\_date))

.all()

)

disease\_count = len(medical\_certificates)

if disease\_count > max\_diseases\_for\_year:

report.frequently\_abs += 1

report.frequently\_list.append(child)

elif not disease\_count:

report.health\_indx\_abs += 1

report.health\_indx\_list.append(child)

report.diseases\_count\_abs += disease\_count

for medical\_certificate in medical\_certificates:

if medical\_certificate.start\_date and medical\_certificate.end\_date:

report.missed\_days\_abs = (medical\_certificate.end\_date - medical\_certificate.start\_date).days

### Health groups

medical\_examination = (

self.session

.query(MedicalExamination)

.filter(MedicalExamination.medcard\_num == child.medcard\_num)

.filter(extract('year', report\_date) <= extract('year', MedicalExamination.examination\_date))

.order\_by(desc(MedicalExamination.examination\_date))

.first()

)

if medical\_examination:

if medical\_examination.health\_group == 'I':

report.first\_health\_group += 1

report.first\_health\_group\_list.append(child)

elif medical\_examination.health\_group == 'II':

report.second\_health\_group += 1

report.second\_health\_group\_list.append(child)

elif medical\_examination.health\_group == 'III':

report.third\_health\_group += 1

report.third\_health\_group\_list.append(child)

### Tub positive

isPositiveMantouxTest = (

self.session

.query(MantouxTest)

.filter(MantouxTest.medcard\_num == child.medcard\_num)

.filter(MantouxTest.result == 'Положительно')

.filter(extract('year', report\_date) == extract('year', MantouxTest.check\_date))

.first()

)

if isPositiveMantouxTest:

report.tub\_positive += 1

report.tub\_positive\_list.append(child)

### Disorders

screening = (

self.session

.query(Screening)

.filter(Screening.medcard\_num == child.medcard\_num)

.filter(extract('year', report\_date) == extract('year', Screening.examination\_date))

.order\_by(desc(Screening.examination\_date))

.first()

)

if screening:

if screening.speech\_defects:

report.speech\_defects += 1

report.speech\_defects\_list.append(child)

if screening.carriage == 'Незначительные отклонения':

report.poor\_posture += 1

report.poor\_posture\_list.append(child)

if screening.carriage == 'Значительные отклонения':

report.scoliosis += 1

report.scoliosis\_list.append(child)

if screening.physical\_fitness == 'Снижена':

report.physical\_development\_disorders += 1

report.physical\_development\_disorders\_list.append(child)

if screening.neurotic\_disorders:

report.neurotic\_disorders += 1

report.neurotic\_disorders\_list.append(child)

if screening.thinking\_and\_speech\_disorders:

report.thinking\_and\_speech\_disorders += 1

report.thinking\_and\_speech\_disorders\_list.append(child)

if screening.motor\_development\_disorders:

report.motor\_development\_disorders += 1

report.motor\_development\_disorders\_list.append(child)

if screening.attention\_and\_memory\_disorders:

report.attention\_and\_memory\_disorders += 1

report.attention\_and\_memory\_disorders\_list.append(child)

if screening.social\_contacts\_disorders:

report.social\_contacts\_disorders += 1

report.social\_contacts\_disorders\_list.append(child)

### Allergyes

allergy = (

self.session

.query(Allergy)

.filter(Allergy.medcard\_num == child.medcard\_num)

.filter(Allergy.diagnosis\_date <= report\_date)

.first()

)

if allergy:

report.allergyes += 1

report.allergyes\_list.append(child)

### END FOR

### Calc

report.children\_count = len(children)

report.frequently\_per = round(report.frequently\_abs / report.children\_count \* 100, 2)

report.health\_indx\_per = round(report.health\_indx\_abs / report.children\_count \* 100, 2)

report.diseases\_count\_on\_thousand = round(report.diseases\_count\_abs / report.children\_count \* 1000, 2)

report.missed\_days\_on\_one = round(report.missed\_days\_abs / report.children\_count)

return report

# *Приложение 12*

**Код функций для формирования различных отчетов**

def get\_children\_by\_mantoux\_test\_result(self, mantoux\_test\_result: str, start\_date: date, end\_date: date) -> list[Child]:

return (

self.session

.query(Child)

.join(MantouxTest, Child.medcard\_num == MantouxTest.medcard\_num)

.filter(MantouxTest.result == mantoux\_test\_result)

.filter(MantouxTest.check\_date.between(start\_date, end\_date))

.all()

)

def get\_children\_by\_deworming\_result(self, deworming\_result: str, start\_date: date, end\_date: date) -> list[Child]:

return (

self.session

.query(Child)

.join(Deworming, Child.medcard\_num == Deworming.medcard\_num)

.filter(Deworming.result == deworming\_result)

.filter(Deworming.deworming\_date.between(start\_date, end\_date))

.all()

)

def get\_children\_by\_vaccine(self, vac: VacReport) -> list[Child]:

return (

self.session

.query(Child)

.join(Vaccination, Child.medcard\_num == Vaccination.medcard\_num)

.filter(Vaccination.vac\_name\_id == vac.vac\_name\_id)

.filter(Vaccination.vac\_type == vac.vac\_type)

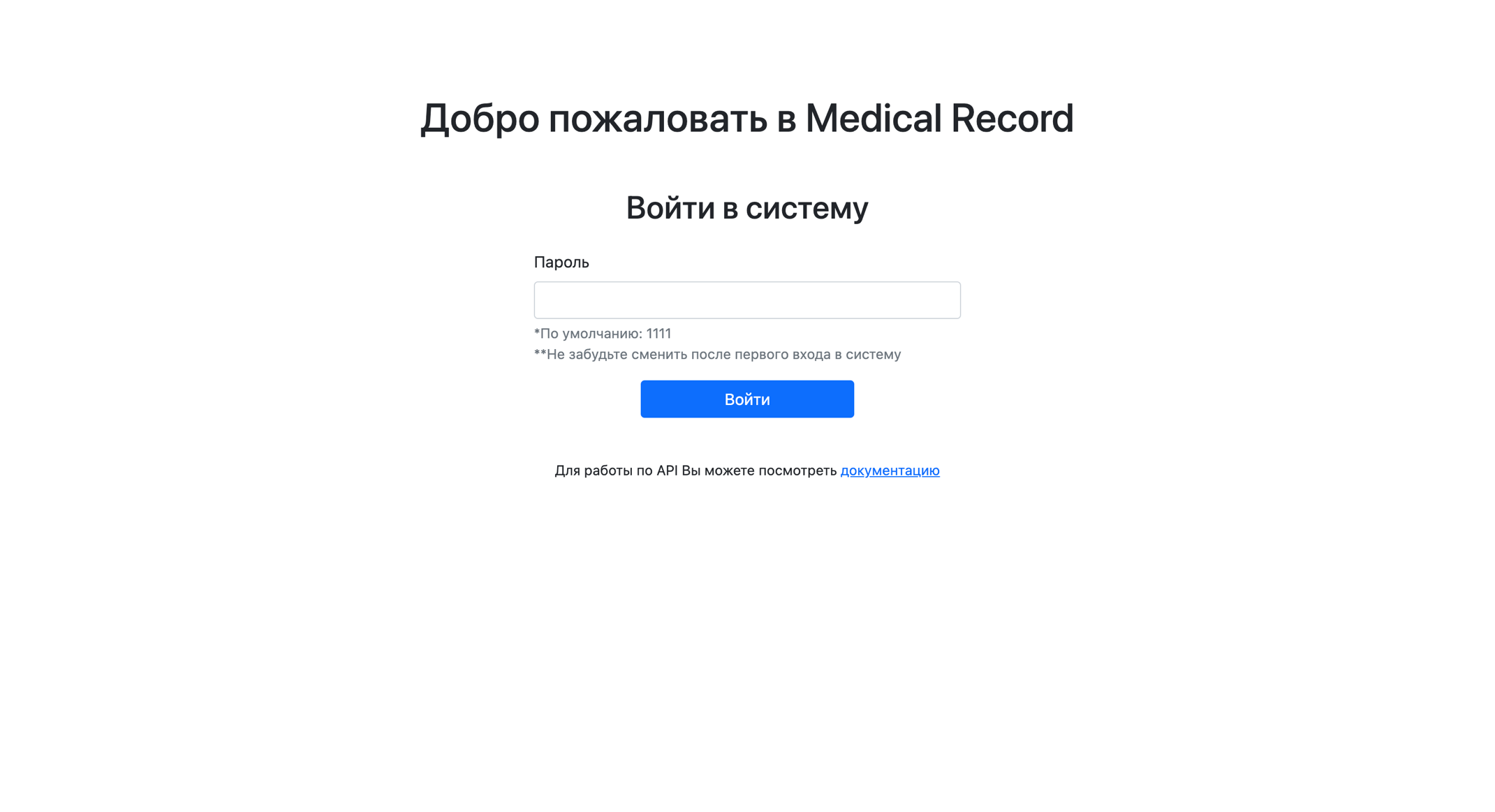
.distinct()

.all()

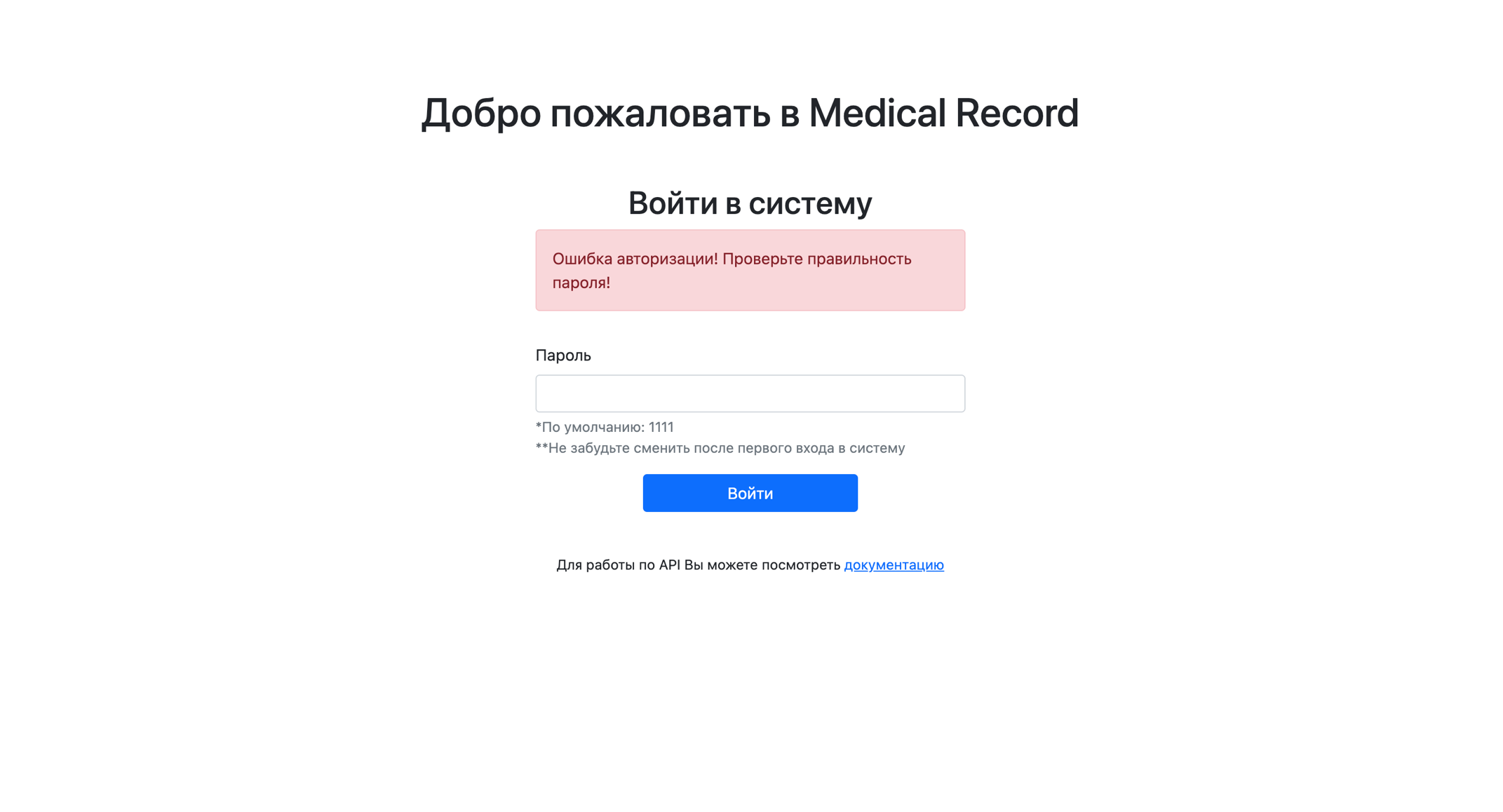
)

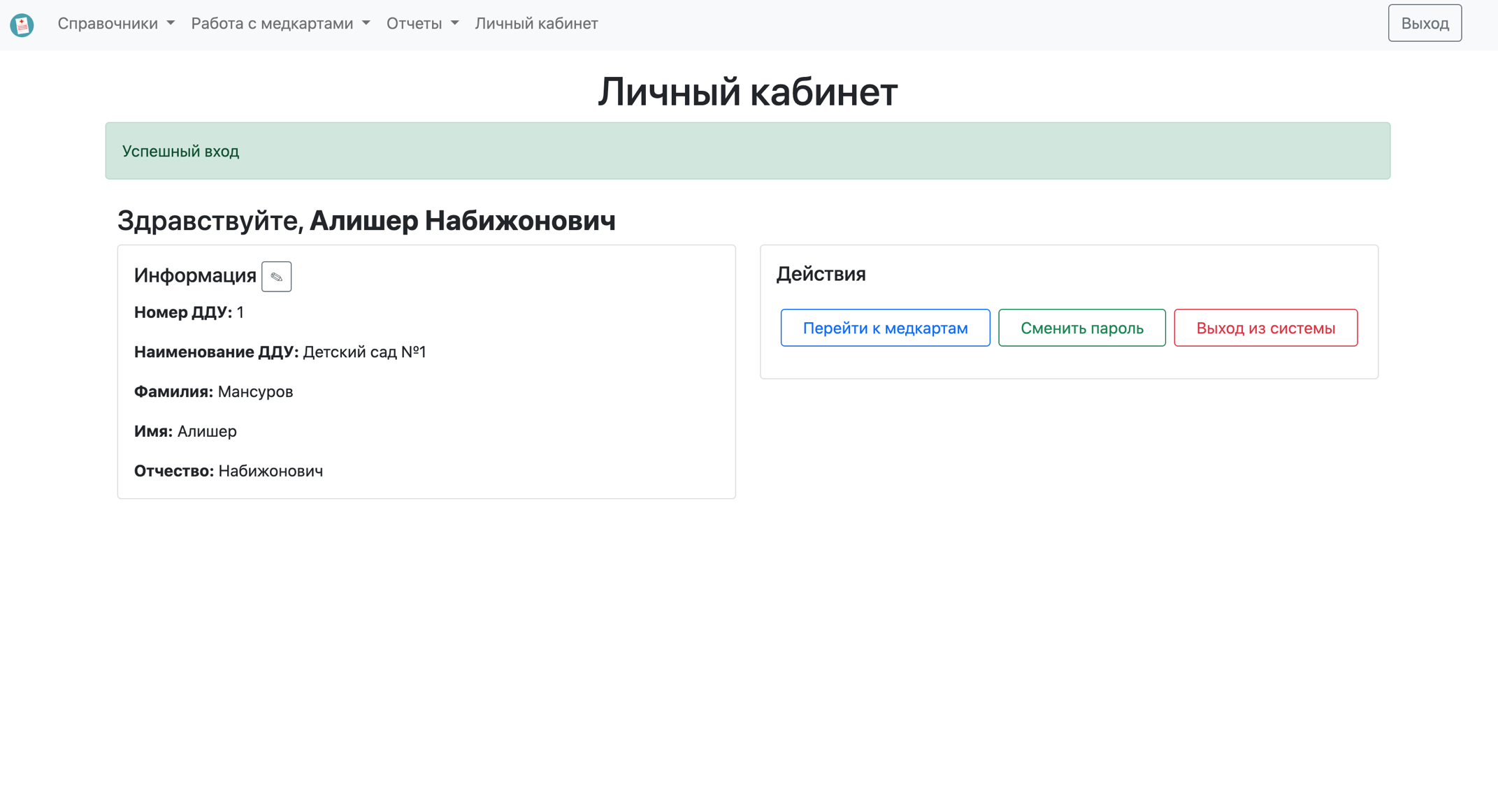
# *Приложение 13*

**Разработанный web-интерфейс**

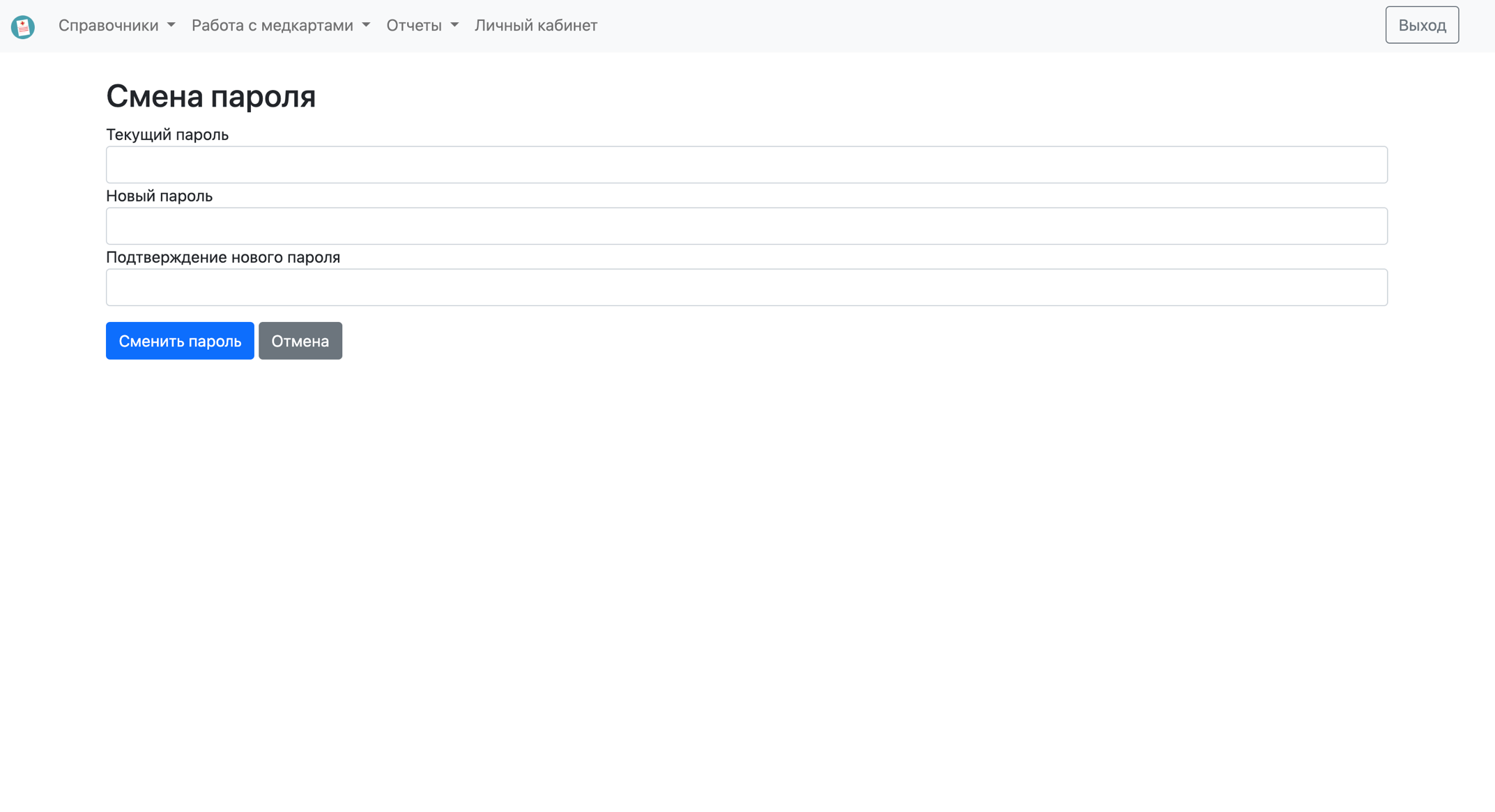
****

Окно входа в систему

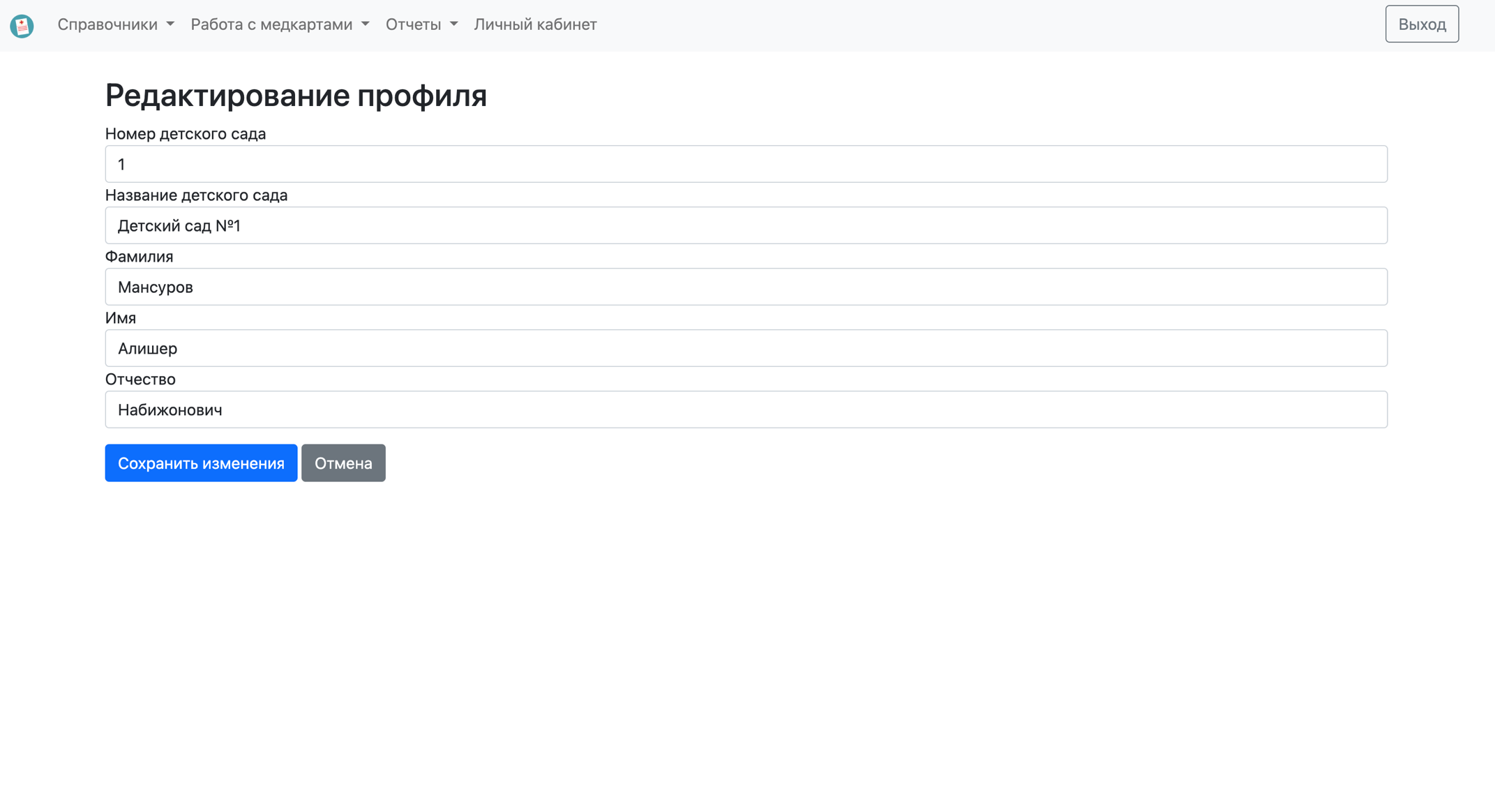
Ошибка входа в систему



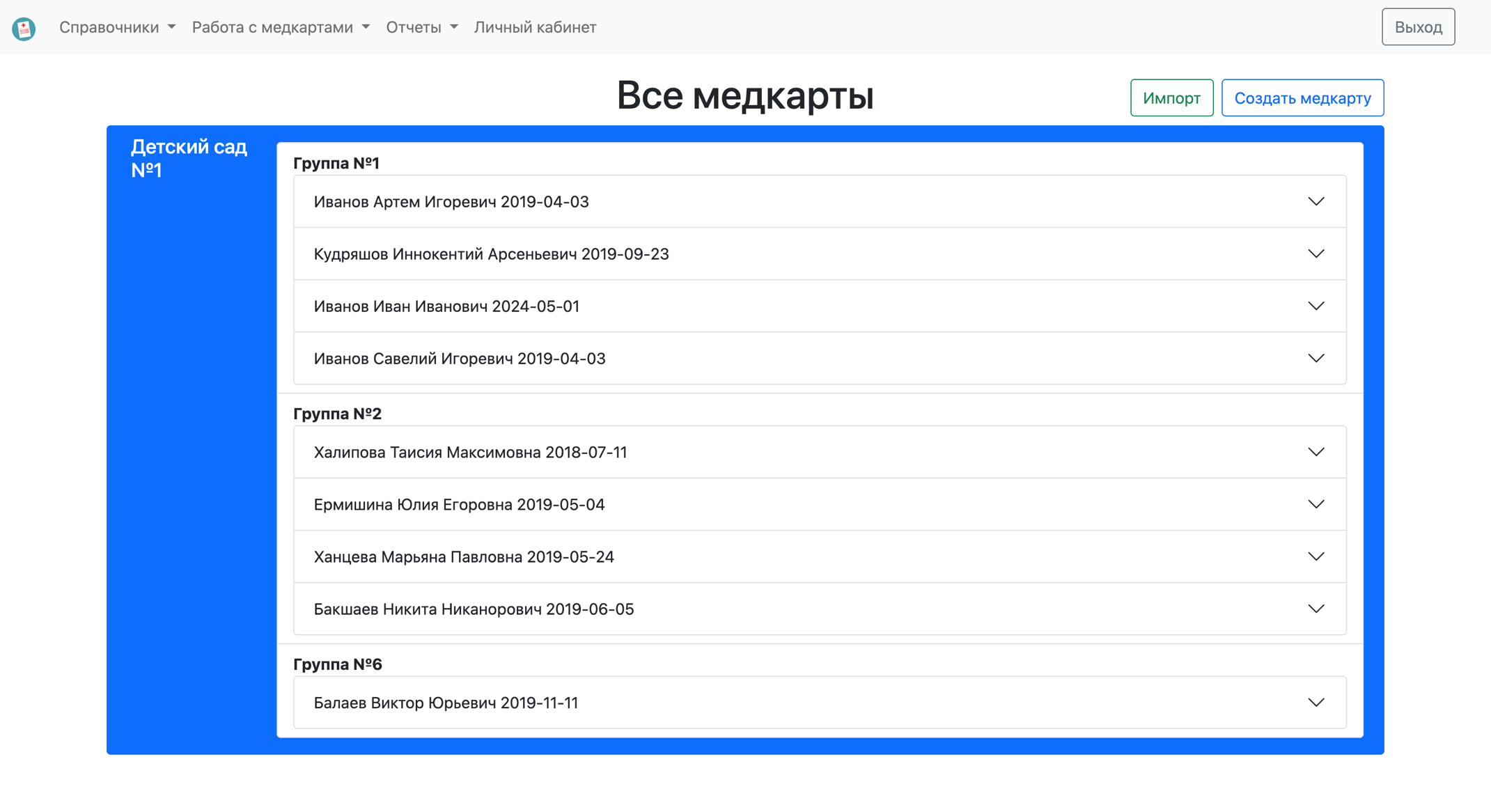
Успешный вход в систему



Окно смены пароля

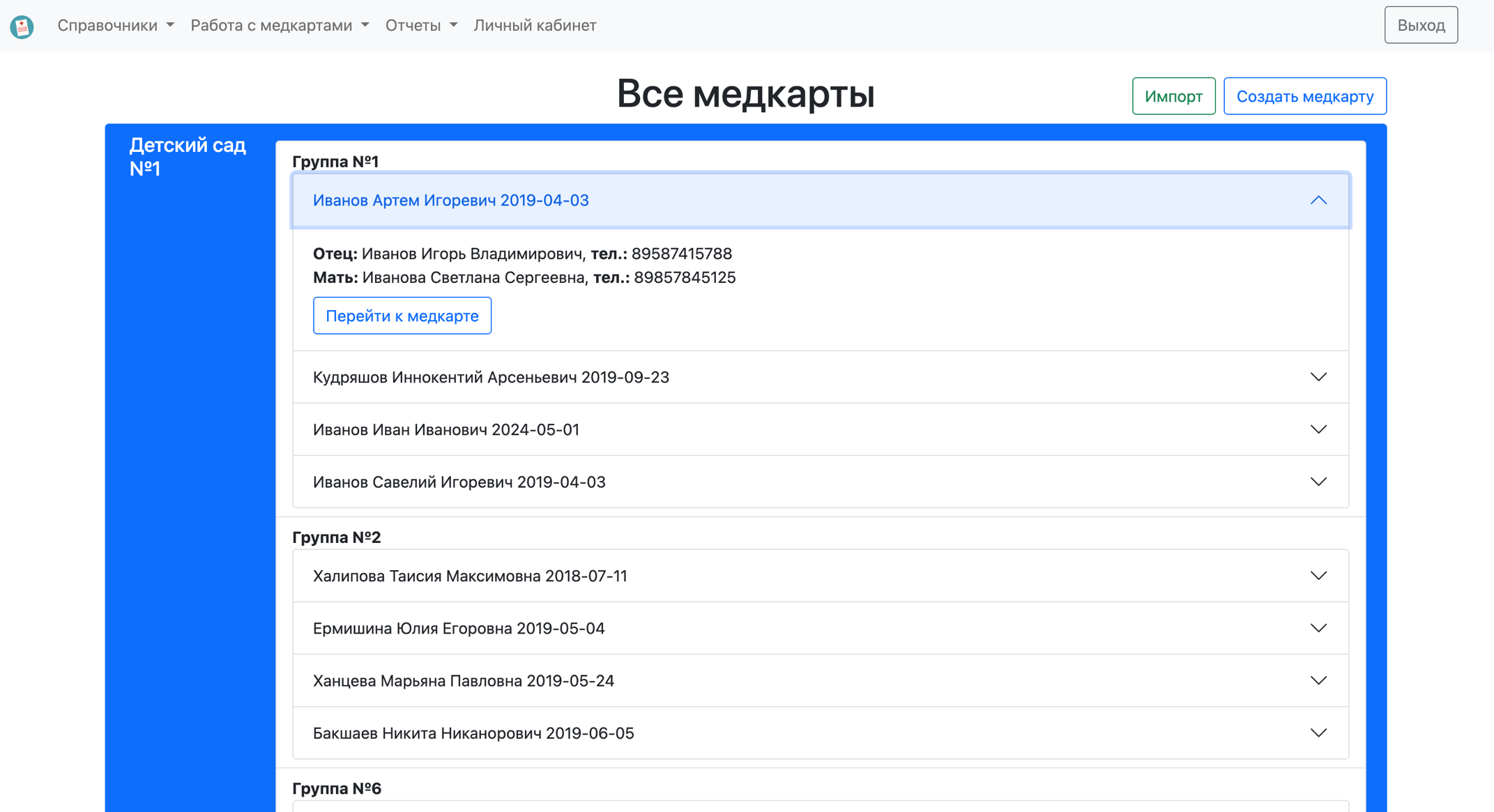


Окно редактирования данных Пользователя

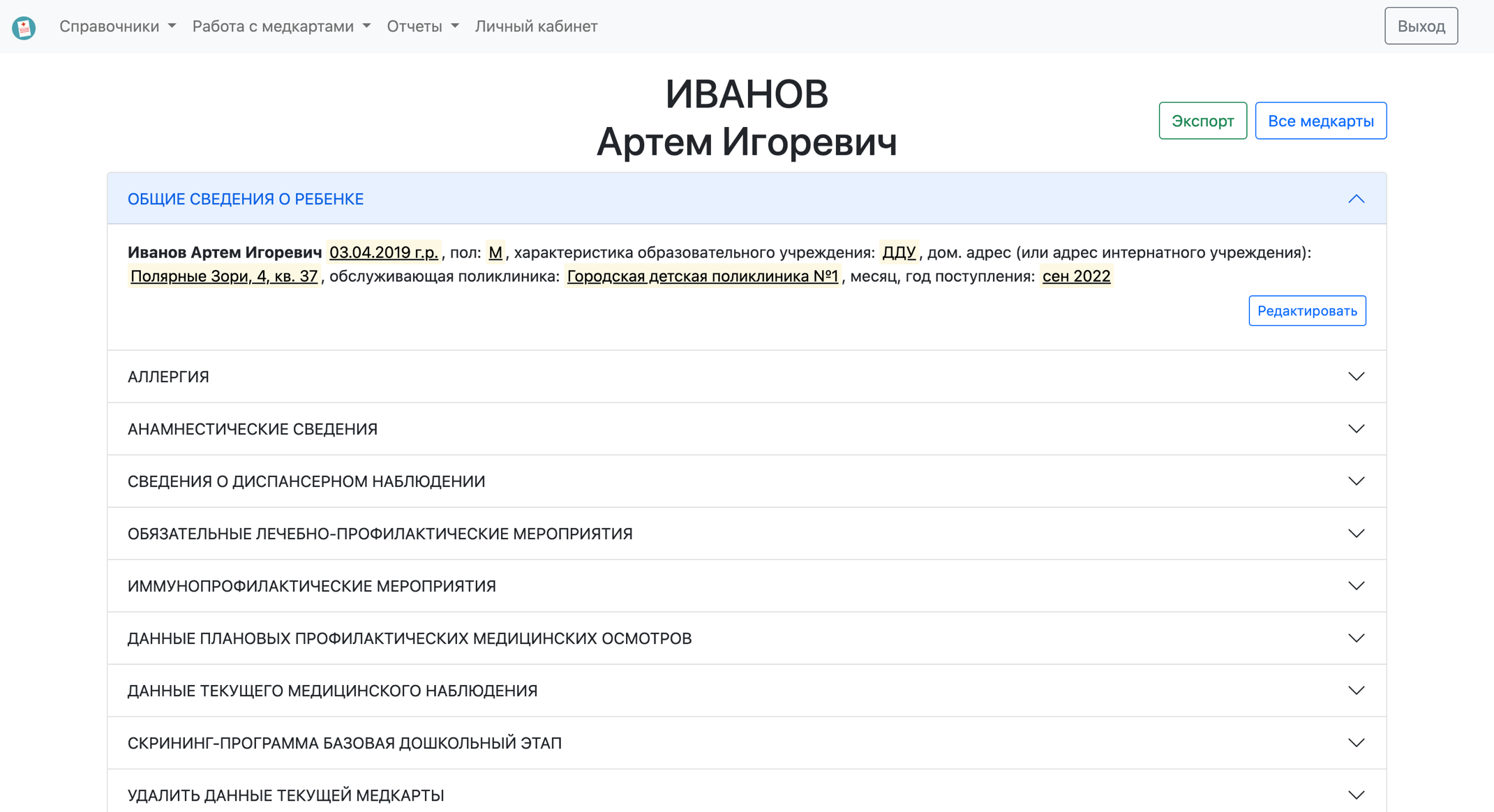


Обзор всех медкарт

В данном разделе можно выбрать любую медкарту и посмотреть краткие сведения о родителях и номера их телефонов, а также перейти к самой медкарте

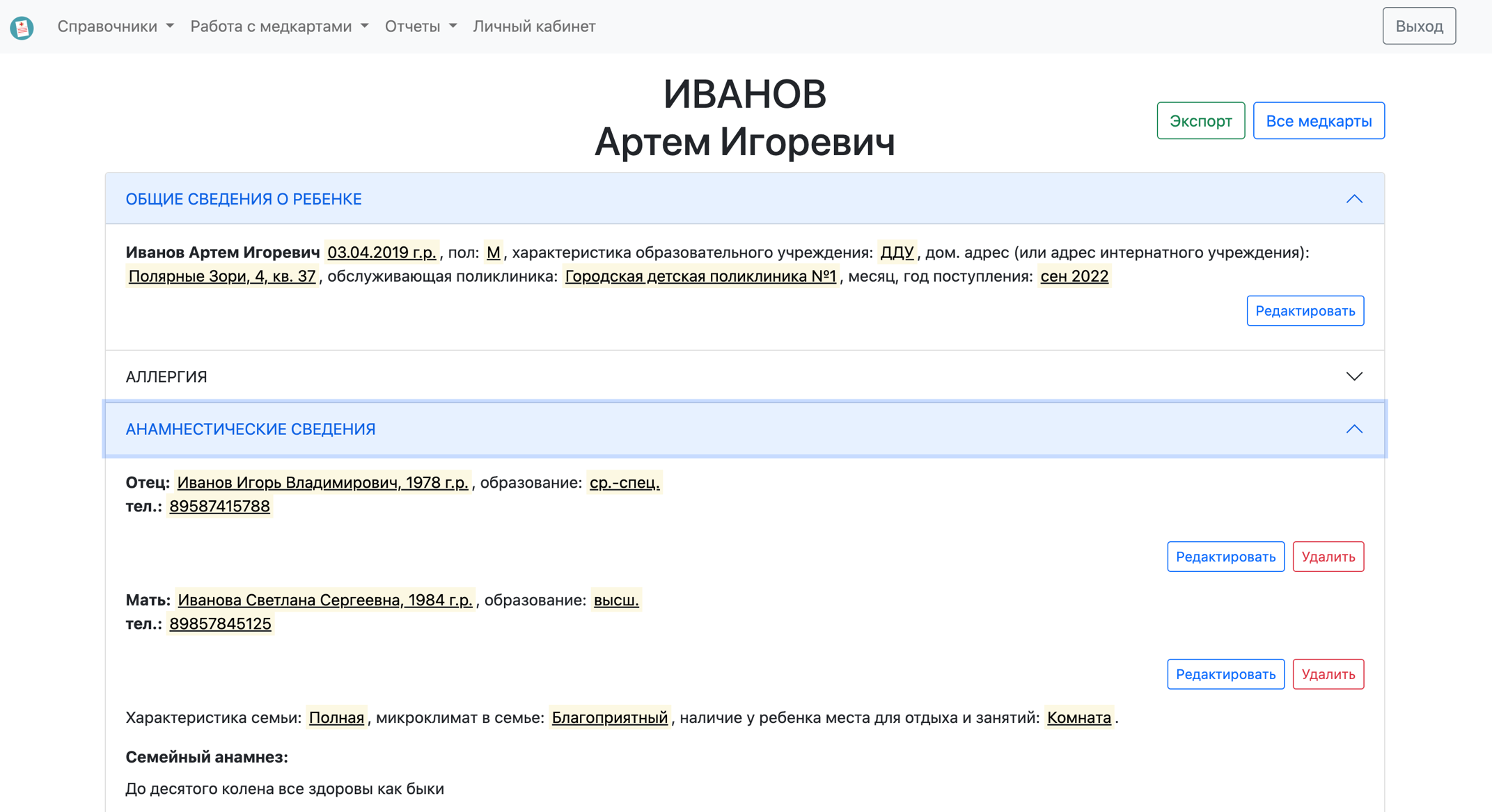


Краткий обзор при нажатии на медкарту

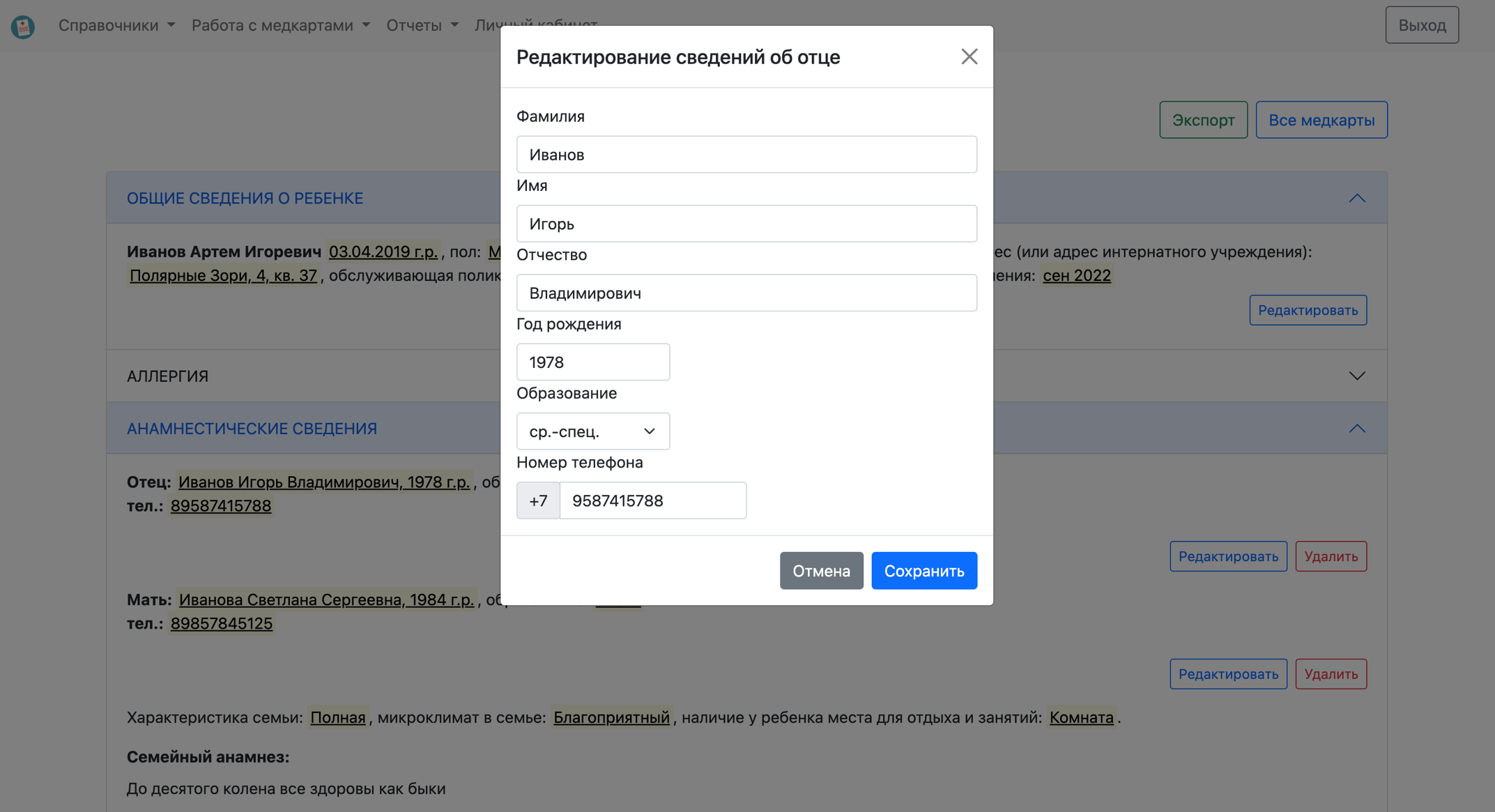


Основное окно медкарты

На данном этапе можно выбрать конкретный раздел медкарты и посмотреть, добавить, отредактировать или же удалить внесенные в него данные. Для редактирования данных используются всплывающие модальные окна.

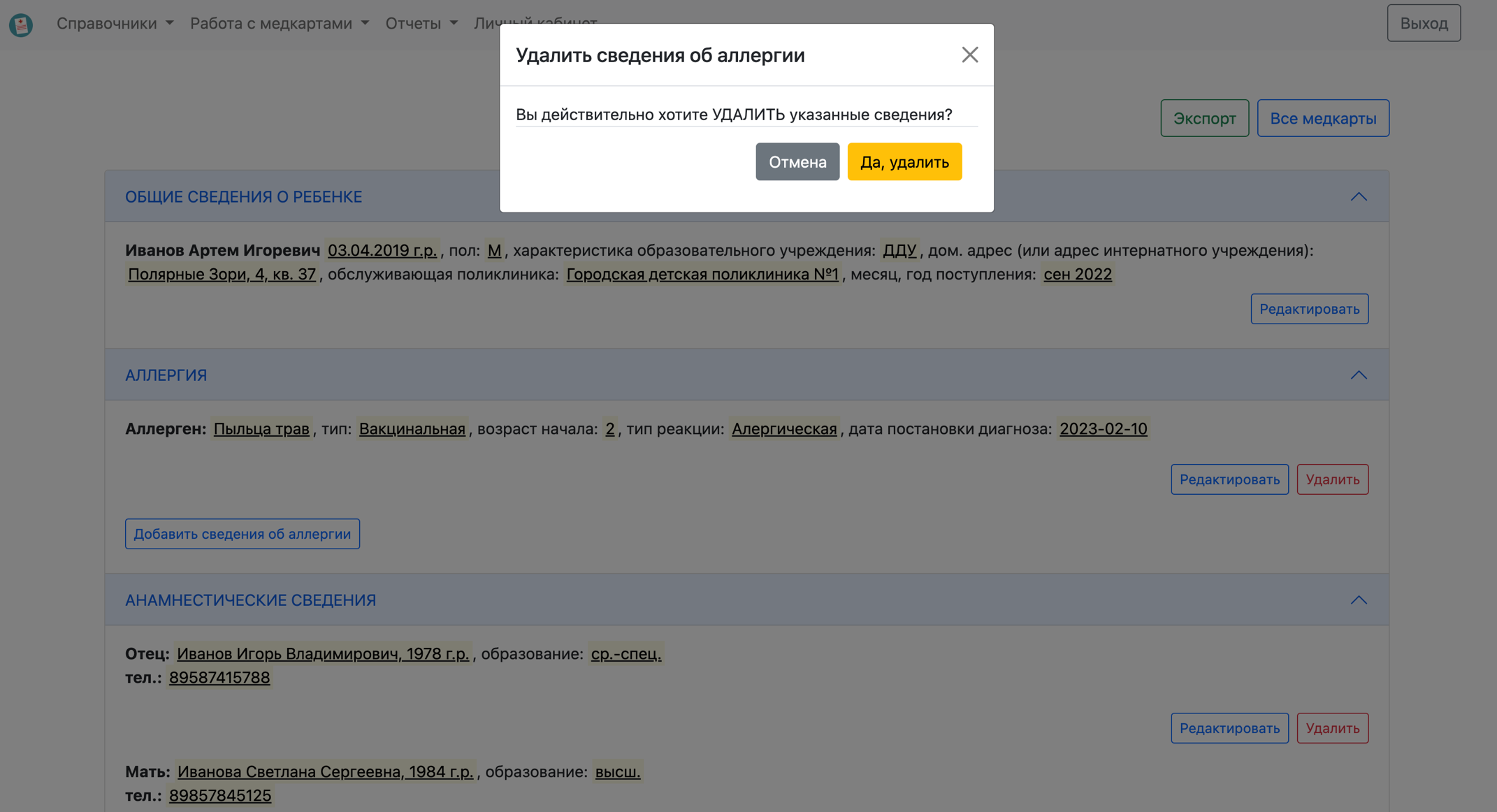


Выбор конкретного раздела



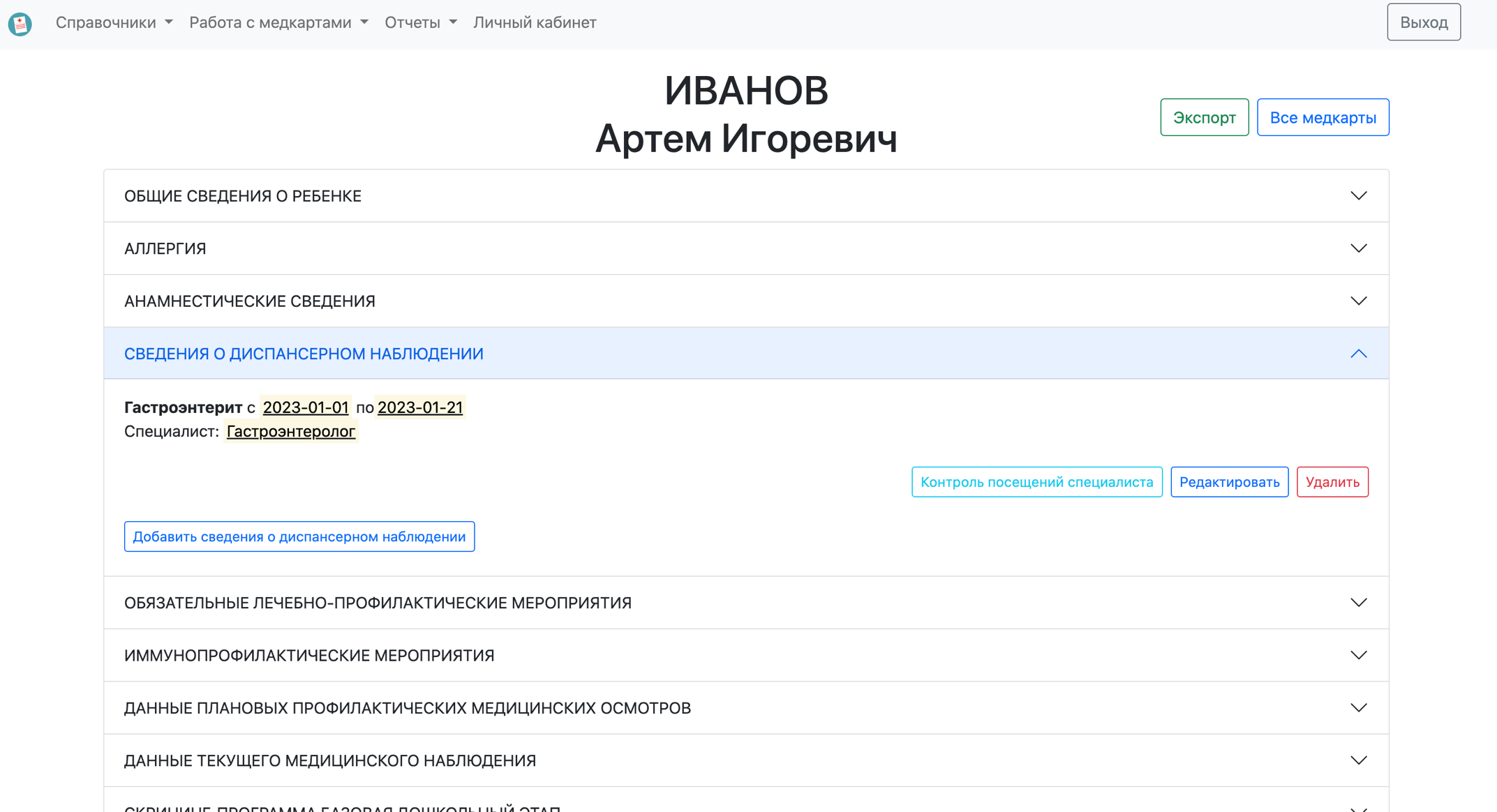
Модальное окно редактирования данных

В случае удаления данных, также предусмотрено модальное окно подтверждения опасного действия.

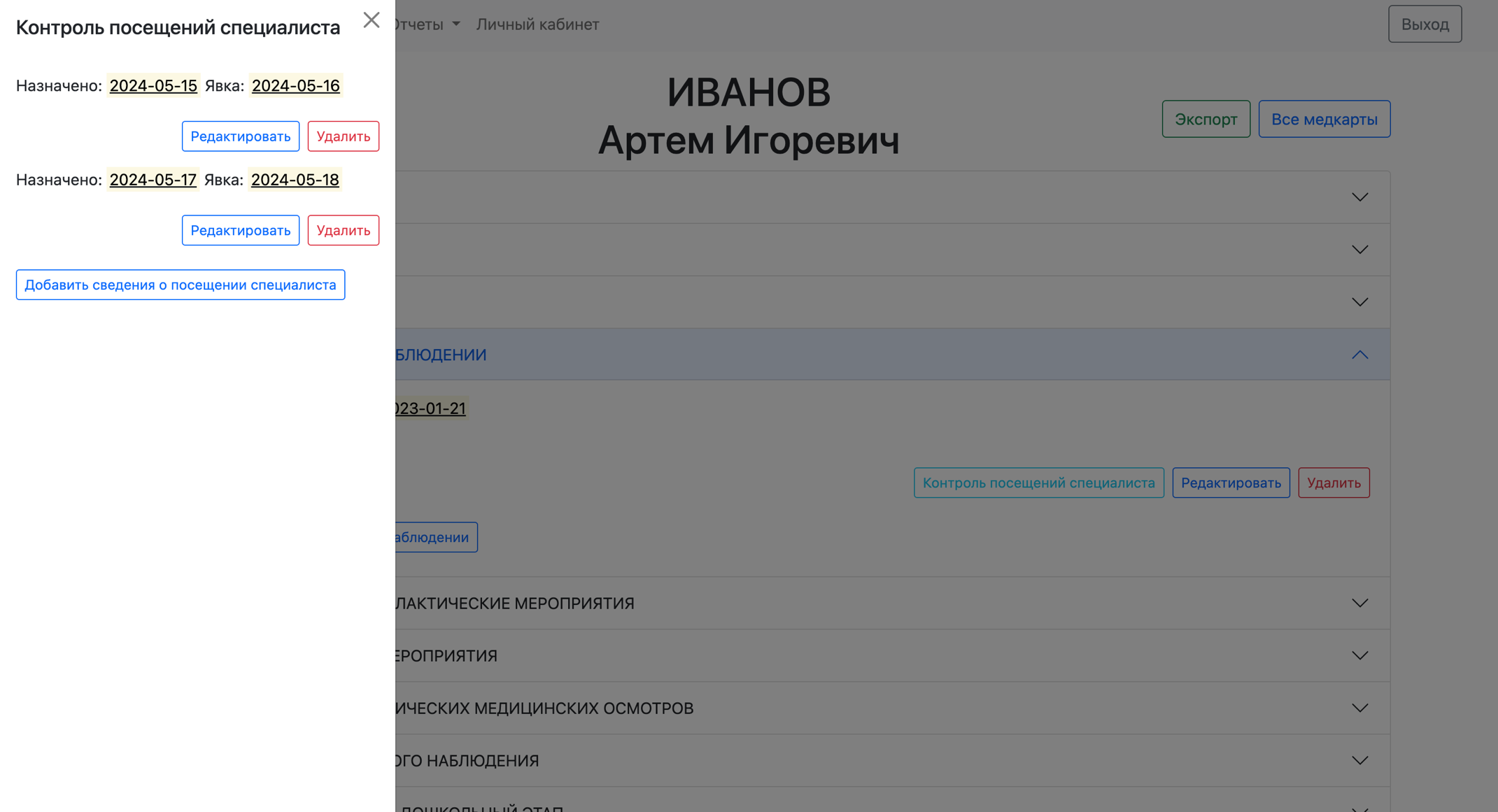


Модальное окно подтверждения удаления данных

В разделе сведений о диспансерном наблюдении предусмотрен контроль посещений специалиста, для него создана дополнительная информационная кнопка



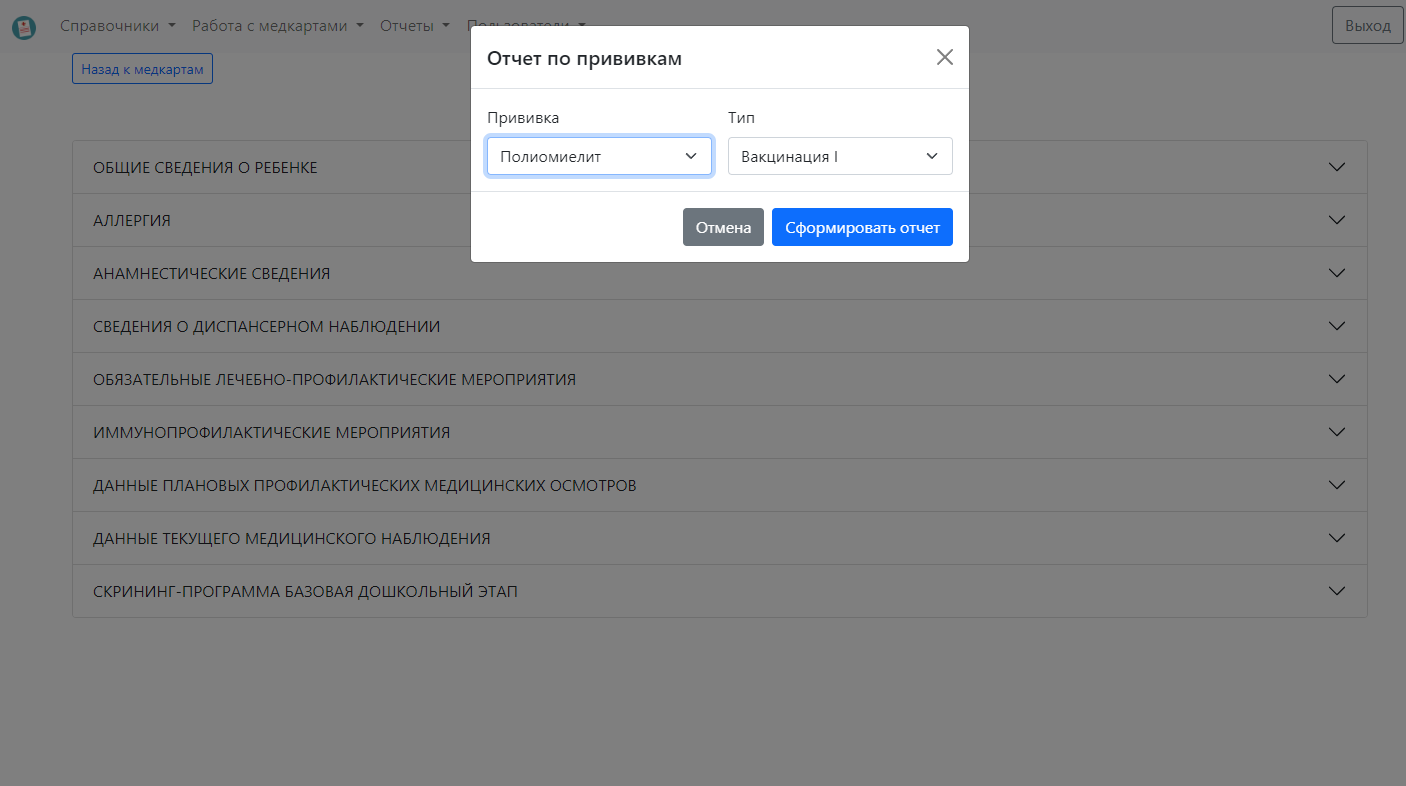
Раздел с дополнительной информационной кнопкой



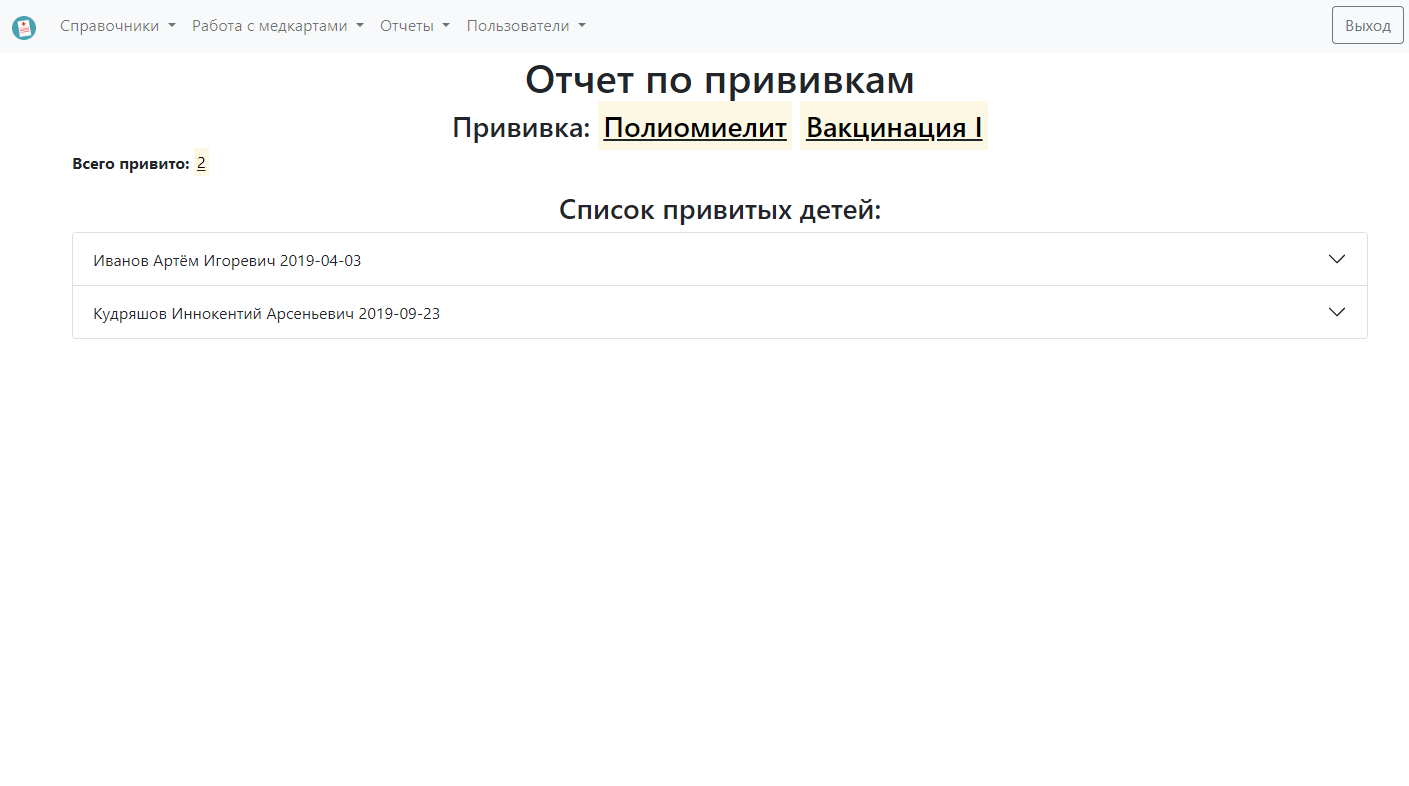
Просмотр листа контроля посещений специалиста

Также предусмотрена работа с отчетами, которая продемонстрирована на примере отчета о прививках.

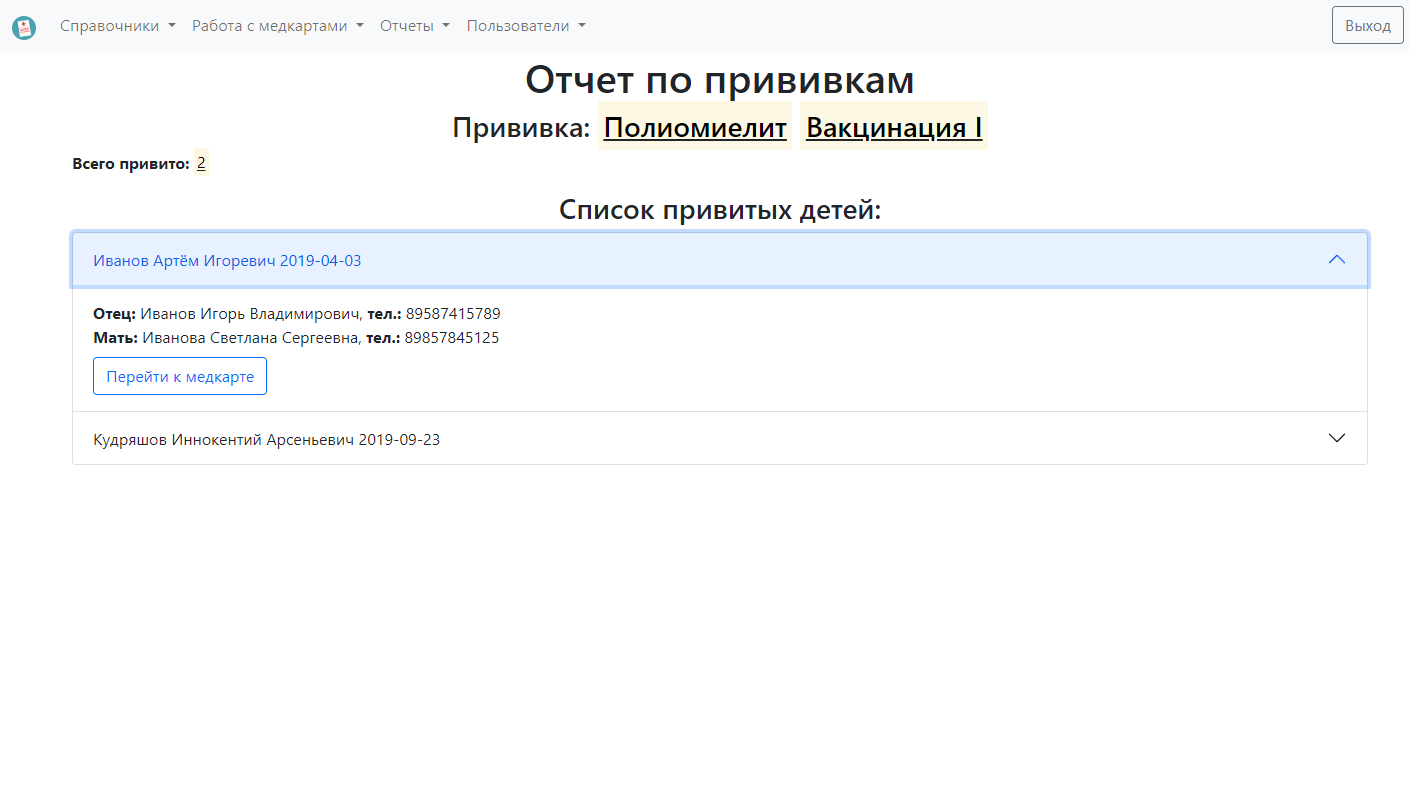
В отчете по прививкам и туберкулинодиагностике, также выводятся краткие сведения по включенным в них детям, с возможностью перехода к медкарте.



Окно выбора отчета по прививкам

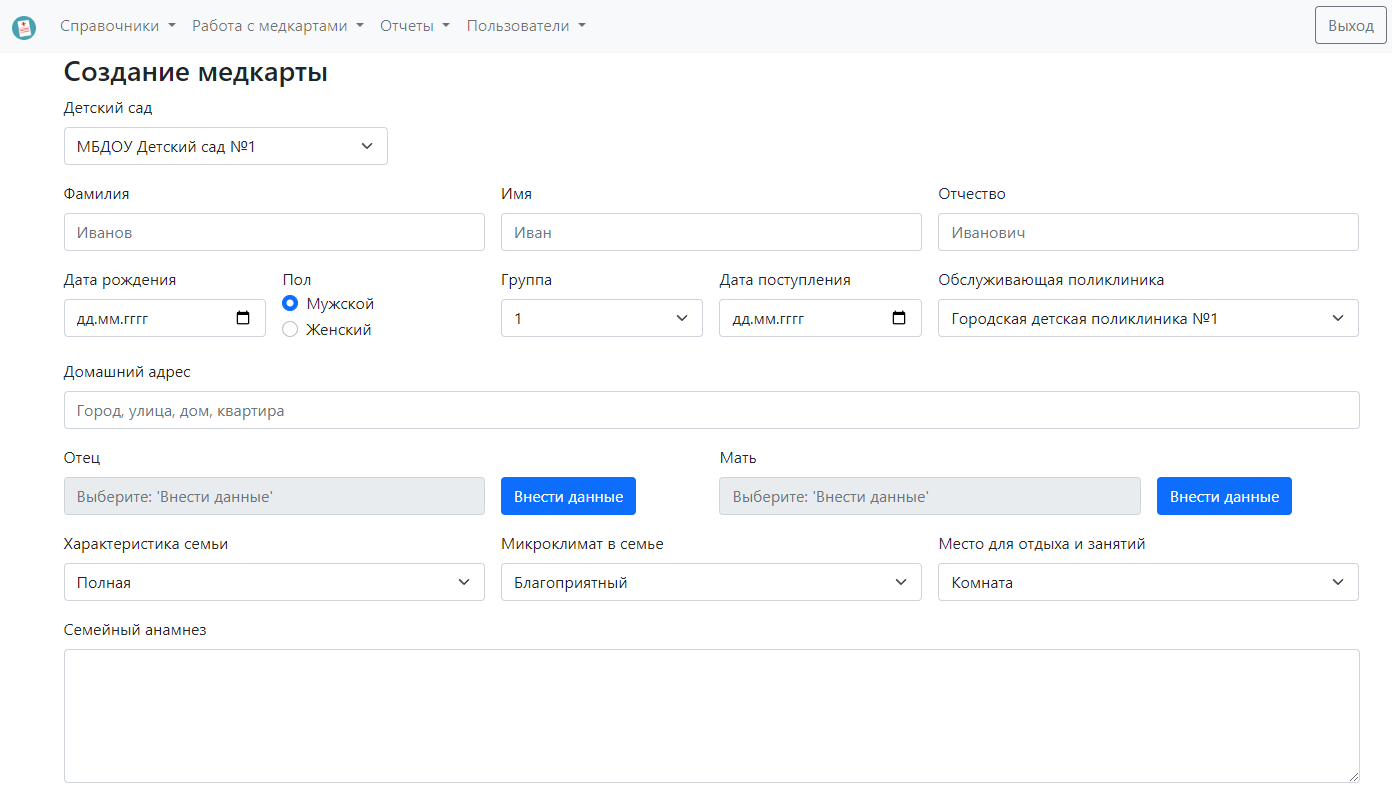


Отчет по прививкам

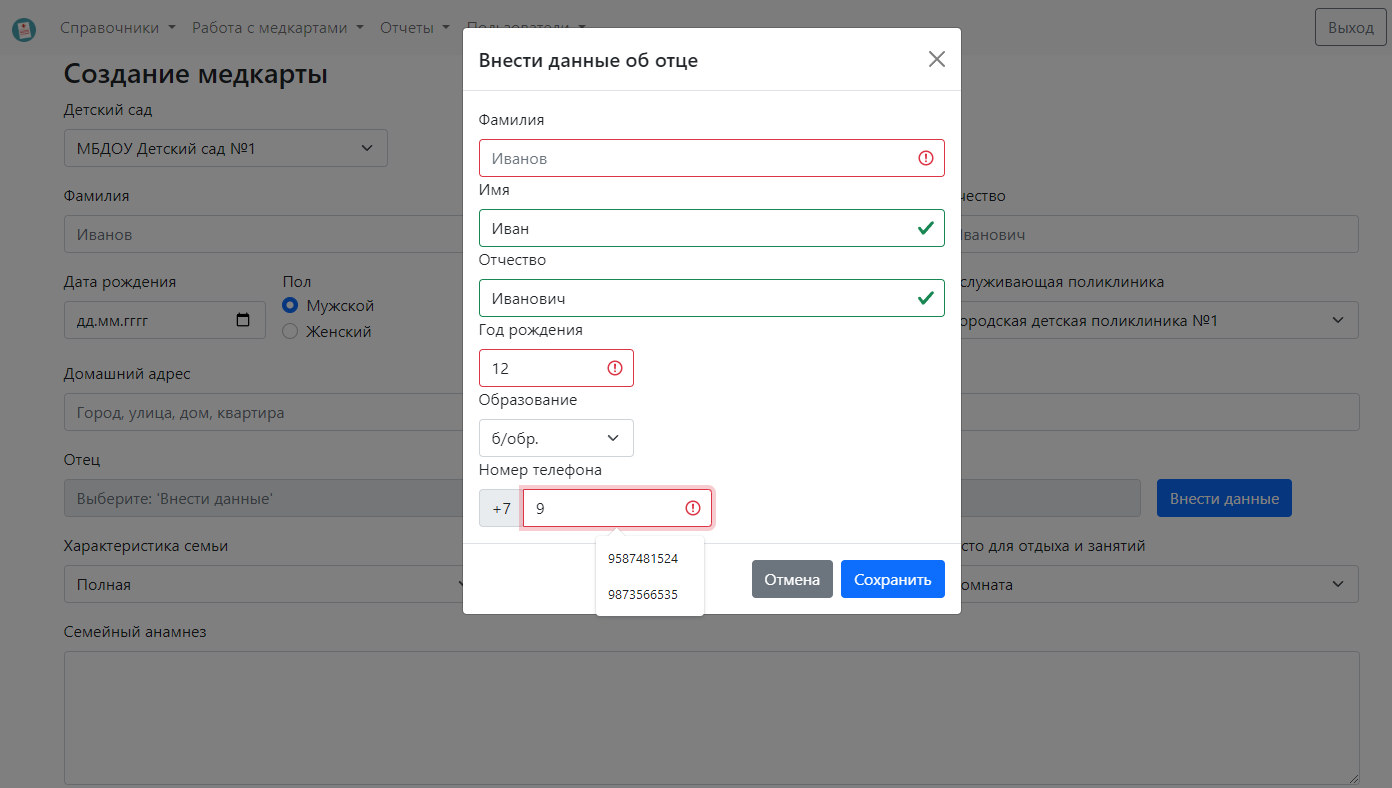


Просмотр кратких сведений и медкарт в отчете

При создании медкарты, для правильности заполнения данных поля форм для заполнения валидируются.

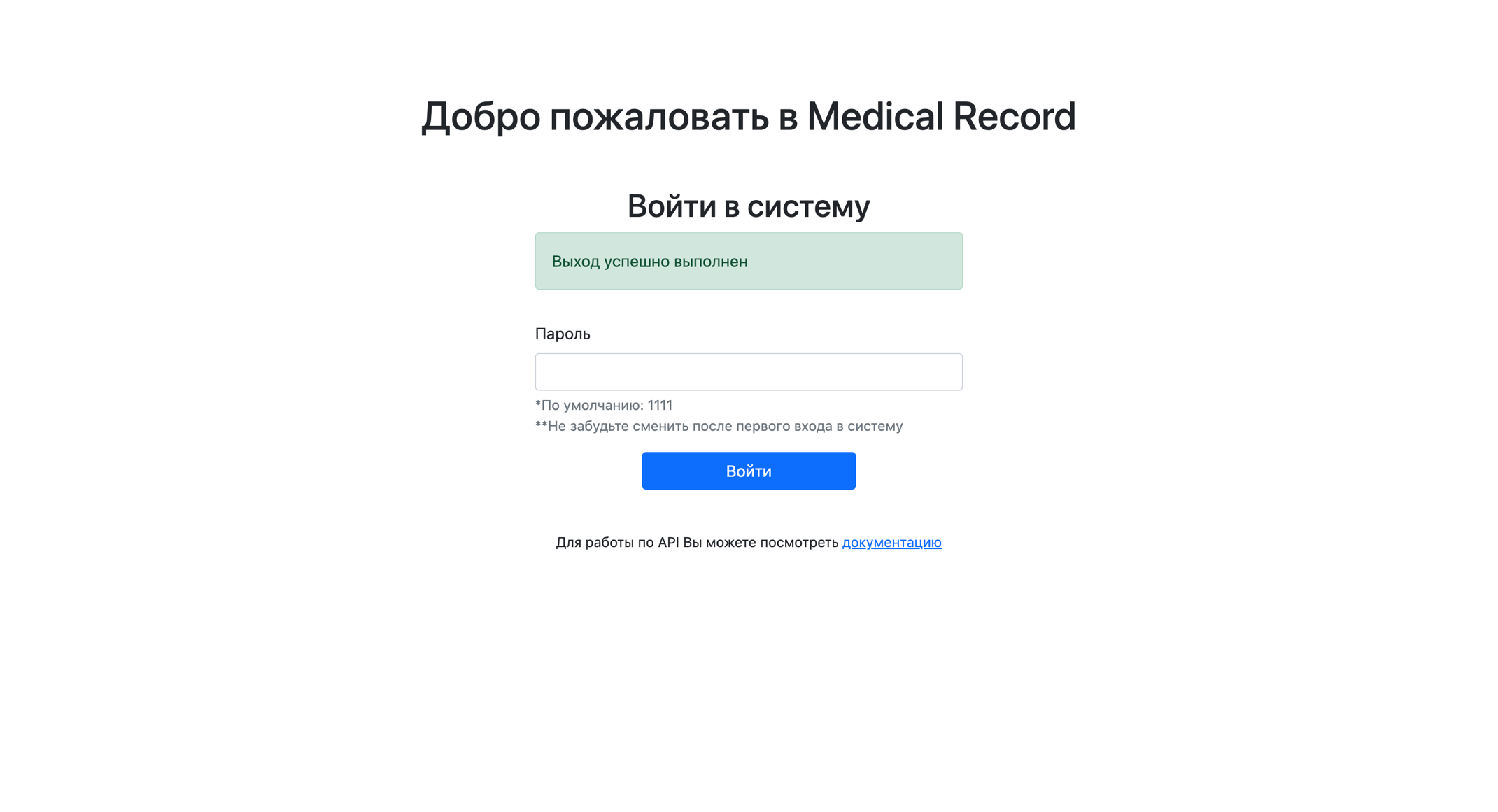


Окно создания медкарты



Пример валидации формы

По нажатию кнопки «Выход» - происходит выход из системы.



Выход из системы

1. Далее – «Медицинская карта ребенка» [↑](#footnote-ref-1)
2. Далее – «Правила» [↑](#footnote-ref-2)