Санкт-Петербургский Государственный Политехнический Университет

Институт информационных технологий и управления

Кафедра Компьютерных Систем и Программных Технологий

**Курсовой проект**

**Отчет по дисциплине «Проектирование архитектур ПО»**

**По теме «Портал государственных услуг»**

**Работу выполнила студентка группы № 13541/3**

Фомина М. А. \_\_\_\_\_\_\_\_\_\_\_

**Работу принял преподаватель:**

Зозуля А.В. \_\_\_\_\_\_\_\_\_\_\_

.

Санкт-Петербург

2017

## Этапы курсового проекта

1. Постановка задачи
   * Описание назначения проектируемой системы
   * Функциональные требования (текстовое описание Участников и их Интересов)
   * Описание бизнес-процессов (этапы, Участники, вовлеченные Сущности)
2. Разработка вариантов использования
   * Диаграммы прецедентов для всех ролей
   * Подробное текстовое описание всех вариантов использования с альтернативами
3. Разработка статической объектной модели предметной области
   * Моделирование предметной области при помощи диаграммы классов
4. Разработка динамической объектной модели предметной области
   * Моделирование предметной области при помощи диаграмм последовательностей
5. Проектирование слоя бизнес-логики
   * Сравнительный анализ и выбор архитектурного шаблона уровня бизнес-логики
6. Реализация слоя бизнес-логики (Java, NetBeans)
   * Применение паттернов проектирования
   * Реализация unit-тестирования всех вариантов использования (JUnit)
   * Использование вместо слоя хранения шаблона "Репозиторий"
7. Проектирование слоя источников данных
   * Выбор архитектурного шаблона уровня доступа к данным
   * Источники данных: реляционная БД, внешний сервис (REST API, xml/json/yaml файл)
8. Реализация слоя источников данных
   * Корректировка unit-тестов
9. Проектирование сервисного слоя и слоя представления GUI (Swing)
   * Клиенты бизнес-логики: GUI, внешний сервис (xml/json/yaml по HTTP)
10. Реализация слоев представления, сервисного слоя
    * Реализация внешнего API при помощи [Simple high-level embedded HTTP server](https://docs.oracle.com/javase/8/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/package-summary.html)
11. Комплексное тестирование системы
12. Написание пояснительной записки (включает все разделы, указанные выше, а также выводы)

**Содержание пояснительной записки**

1. Постановка задачи
2. Варианты использования
3. Моделирование
4. Слой бизнес-логики
5. Слой источников данных
6. Сервисный слой
7. Слой представления
8. Интеграция
9. Тестирование
10. Выводы

**1. Постановка задачи**

**1.1. Описание назначения проектируемой системы**

Система, проектируемая в ходе выполнения курсового проекта, является системой автоматизации для оказания государственных услуг, а именно, процессов, связанных с записью ребёнка в школу; записью на приём к врачу; отправкой отзывов об образовательных и медицинских учреждениях, а также о конкретных врачах.

**1.2. Функциональные требования (текстовое описание участников и их интересов)**

**Гражданин.**

1. Гражданин может заполнить информацию о своих детях и о себе.
2. Гражданин может отправлять заявки на запись ребёнка в образовательное учреждение.
3. Гражданин может отменить заявку на запись ребёнка в образовательное учреждение.
4. Гражданин может записываться на приём к врачу.
5. Гражданин может отменить свою запись на приём к врачу.
6. Гражданин может оставить вопрос, отзыв или жалобу об образовательных и медицинских учреждениях.

**Представитель учебного учреждения.**

1. Добавляет информацию об учебном учреждении.
2. Добавляет информацию о количестве свободных мест для приёма школьников.
3. Одобряет заявки по приёму школьников.
4. Назначает время для прихода в школу гражданина для сдачи документов и окончательного оформления школьника.
5. После оформления документов окончательно подтверждает, что ребёнок будет учиться в школе. Если же документы не были своевременно оформлены, то имеет право отклонить заявку.
6. Отвечает на вопросы и жалобы граждан.

**Представитель медицинского учреждения.**

1. Добавляет информацию о медицинском учреждении и врачах.
2. Добавляет информацию о свободных талонах.
3. Отменяет талоны, выданные на приём к врачу, если врач заболел.
4. Отвечает на вопросы и жалобы граждан.

**1.3. Описание бизнес-процессов (этапы, участники, вовлеченные сущности)**

**Участники:** гражданин, представитель учебного учреждения, представитель медицинского учреждения.

**Сущности:** талон, отзыв, заявка на запись в учебное учреждение, учреждение, ребёнок, врач.

**Бизнес-процессы:**

**1.3.1. Запись ребёнка в школу**

1. Представители учебных учреждений добавляют информацию об учреждениях и о свободных местах.
2. Гражданин выбирает образовательное учреждение, в которое хочет отправить ребёнка, и подаёт заявку в него. Гражданин может подать какое угодно количество заявок в разные учреждения.
3. Представитель учебного учреждения подтверждает заявку.
4. Гражданин подтверждает, что действительно выбирает это учреждение, или отклоняет свою заявку на запись.
5. Представитель учебного учреждения назначает время для прихода в школу гражданина для сдачи документов и окончательного оформления школьника.
6. Представитель учебного учреждения окончательно подтверждает, что ребёнок будет учиться в школе. При подтверждении все заявки в другие учреждения, отправленные гражданином, автоматически отменяются.
7. Гражданин добавляет отзыв о работе учебного учреждения и приёме.
8. Представитель учебного учреждения отвечает на отзыв.

**1.3.2. Запись на приём к врачу**

1. Представители медицинских учреждений добавляет информацию о талонах.
2. Гражданин записывается на приём к врачу, выбирая поликлинику, врача и время приёма.
3. Представитель медицинского учреждения может отменить выданные талоны в непредвиденной ситуации, к примеру, если врач заболел. Тогда гражданам должны прийти уведомления о том, что приём отменён.
4. Гражданин может отменить свою запись к врачу. Тогда его талон будет доступен для других людей.
5. Представитель медицинского учреждения подтверждает, что гражданин посетил доктора.
6. Гражданин добавляет отзыв о медицинском учреждении.
7. Представитель медицинского учреждения отвечает на отзыв.

**2. Разработка вариантов использования**

**2.1. Диаграмма прецедентов для всех ролей**

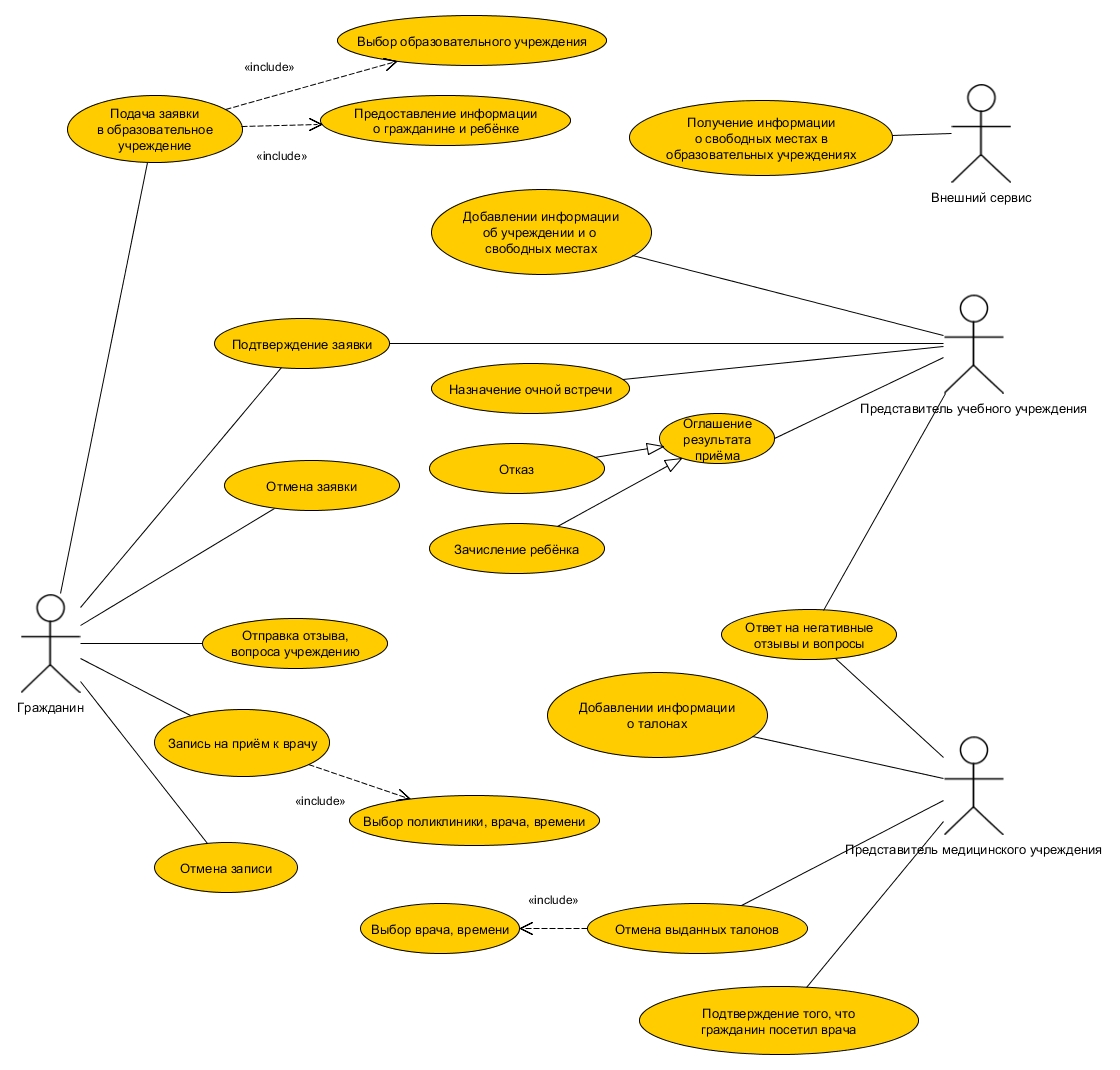


Рис. 1. Диаграмма прецедентов для всех ролей

**2.2. Подробное текстовое описание всех вариантов использования с альтернативами**

**2.2.1. Запись ребёнка в школу**

1. Представитель учебного учреждения добавляет информацию об учреждениях и о свободных местах.
2. Гражданин выбирает образовательное учреждение, в которое хочет отправить ребёнка.
3. Гражданин подаёт заявку в образовательное учреждение.
4. Представитель учебного учреждения подтверждает заявку.
5. Гражданин подтверждает, что действительно выбирает это учреждение.
6. Представитель учебного учреждения назначает время для прихода в школу гражданина для сдачи документов и окончательного оформления школьника.
7. Представитель учебного учреждения окончательно подтверждает, что ребёнок будет учиться в школе
8. Гражданин добавляет отзыв о работе учебного учреждения и приёме.
9. Представитель учебного учреждения отвечает на отзыв.

**Альтернатива.**

5а. Гражданин отклоняет свою заявку на запись. После этого переход на шаг 2.

**Альтернатива.**

6а. Гражданин отклоняет свою заявку на запись. После этого переход на шаг 2.

**Альтернатива.**

7а. Представитель учебного учреждения отклоняет заявку, по причине не прихода родителя на очную встречу и не приносу необходимых документов. После этого переход на шаг 2.

**2.2.2. Запись на приём к врачу**

1. Представитель медицинского учреждения добавляет информацию о талонах.
2. Гражданин выбирает поликлинику, врача, время.
3. Гражданин записывается на приём к врачу.
4. Представитель учебного учреждения подтверждает, что гражданин посетил врача по талону.
5. Гражданин пишет отзыв о медицинском учреждении.
6. Представитель медицинского учреждения отвечает на отзыв.

**Альтернатива.**

4а. Гражданин отменяет свою запись к врачу. Тогда переход к шагу 2. Талон доступен другим гражданам для записи.

**Альтернатива.**

4а. Представитель медицинского учреждения отменяет выданные талоны. Тогда переход к шагу 2.

**Альтернатива.**

4а. Представитель медицинского учреждения не подтверждает посещение врача по талону.

**3. Разработка статической объектной модели предметной области**

**3.1. Моделирование предметной области при помощи диаграммы классов**

**Участники:** гражданин, представитель учебного учреждения, представитель медицинского учреждения, врач.

**Сущности:** талон, отзыв, заявка на запись в учебное учреждение, учреждение, ребёнок.

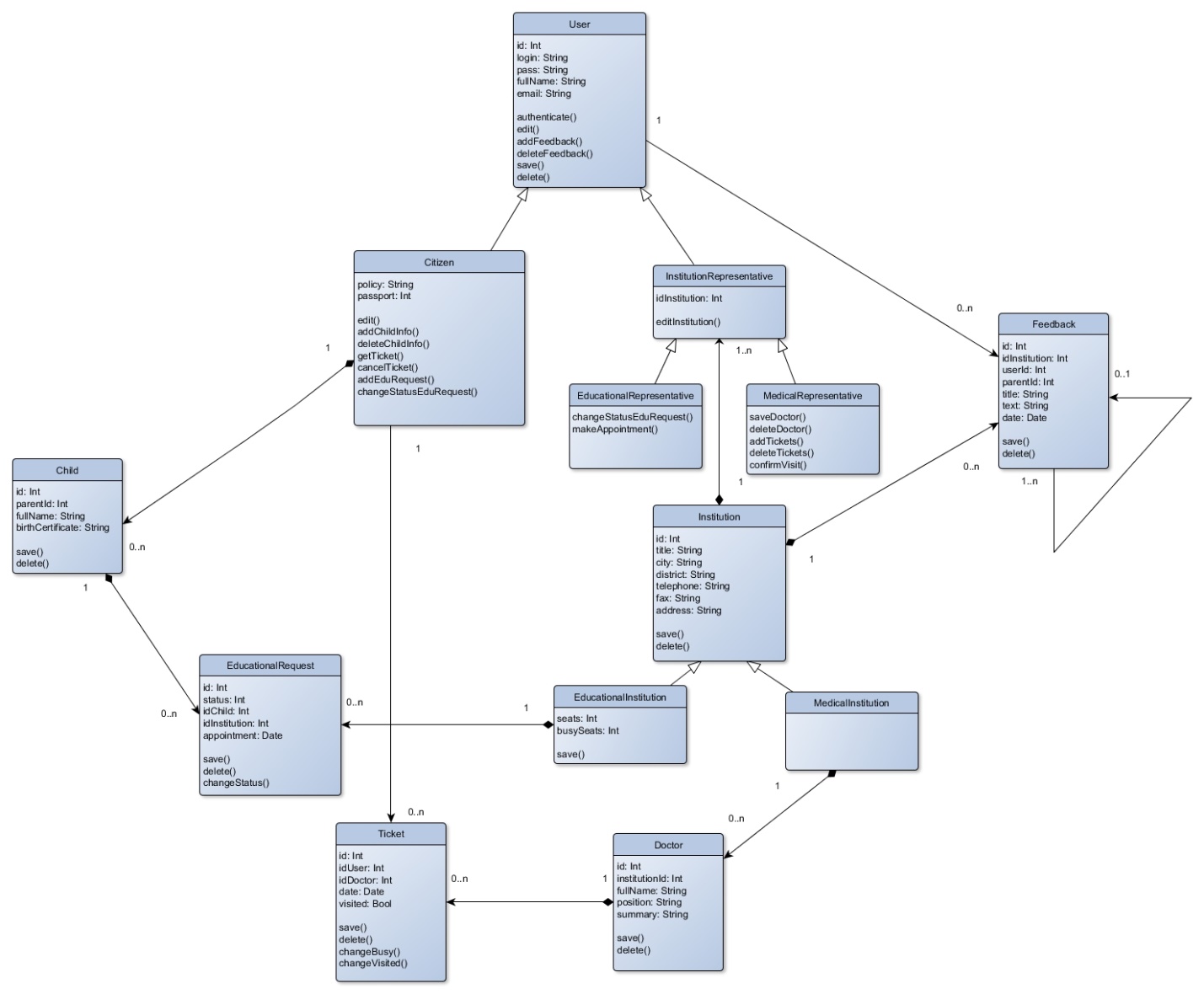
****

Рис. 2. Диаграмма классов

**4. Разработка динамической объектной модели предметной области**

**4.1. Запись ребёнка в школу**

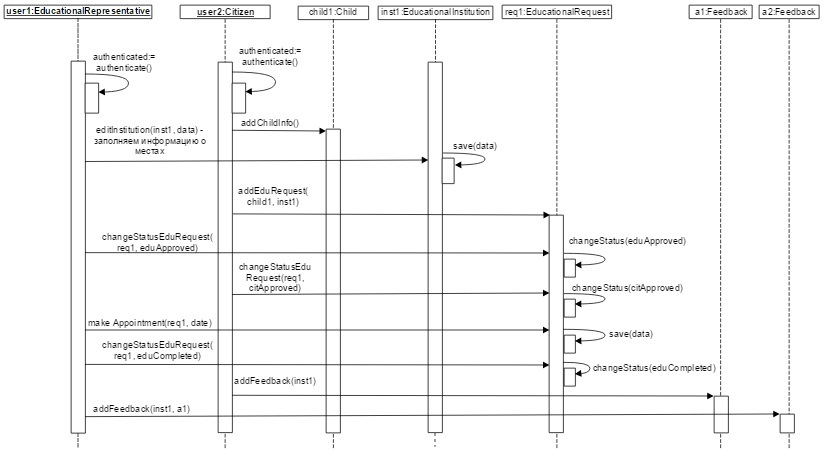


Рис. 3. Диаграмма последовательности для бизнес-процесса записи ребёнка в школу

**4.2. Запись на приём к врачу**

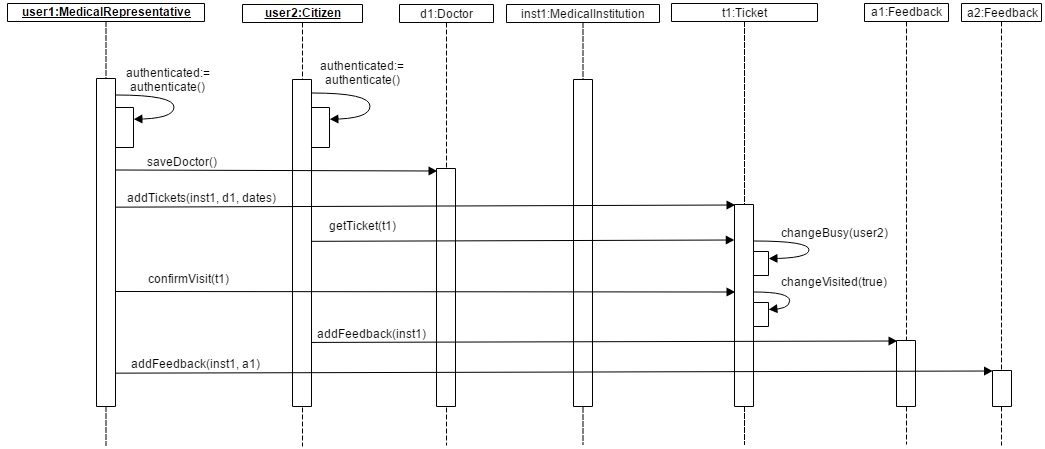


Рис. 4. Диаграмма последовательности для бизнес-процесса записи на приём к врачу

1. **Проектирование слоя бизнес-логики**

В качестве шаблона, который будет использован при создании слоя бизнес-логики была выбрана модель предметной области. Каждый объект наделяется функциями, соответствующими его природе. Образуется сеть взаимосвязанных объектов. Функции тесно сочетаются с данными.

1. **Реализация слоя бизнес-логики**

Слой бизнес логики был реализован с использованием паттерна "Модель предметной области". Все классы, реализующий слой бизнес-логики разделены на три пакета:

1. user - в данном пакете определены все классы, связанные с различными типами пользователей системы
2. entities - в данном пакете определены все классы, связанные с различными сущностями (талон, отзыв, заявка на запись в учебное учреждение, ребёнок, уведомление)
3. institutions - в данном пакете определены классы медицинского и образовательного учреждения
   1. **Пакет user**
      1. **Интерфейс UserInterface**

Интерфейс UserInterface - данный интерфейс определяет базовые методы для всех пользователей.

1. User getUser() - метод, позволяет получить объект класса User
2. void setId(int id) - метод устанавливает ID пользователя, предполагается использовать в преобразователях данных
3. int getId() – метод для получения ID пользователя, предполагается использовать в преобразователях данных
4. boolean isAuthenticated() - метод проверяет, авторизован ли пользователь
5. boolean signIn(String password) – метод для аутентификации пользователя. Возвращает true, если аутентификация успешна
6. void signOut() – метод для выхода из системы
7. void setNotifications(List<Notification> notifications) – метод для установки всех уведомлений пользователя
8. void addNotification(String notification) – метод для добавления нового уведомления
9. List<Notification> getNotifications() – метод для получения всех уведомлений пользователя
10. boolean addFeedback(Institution institution, String text) throws NoRightsException – для добавления отзыва или вопроса учреждению
11. boolean addFeedbackTo(Institution institution, String text, User userTo) throws NoRightsException – метод для добавления отзыва или вопроса учреждению, адресованного конкретному пользователю
12. User.UserType getUserType() – метод для получения конкретного типа пользователя
    * 1. **Класс User**

Класс User - базовый класс пользователя, от которого наследуются все остальные типы пользователей. Все типы пользователей наследуются от него.

1. public String getLogin() – метод для получения уникального логина пользователя
2. public String getFullName() – метод для получения полного имени пользователя
3. public getPassword() – метод для получения зашифрованного пароля пользователя
4. public boolean isDoctor() – метод для проверки, является ли пользователь доктором
5. public boolean isAdministrator() – метод для проверки, является ли пользователь администратором
6. public boolean isCitizen() – метод для проверки, является ли пользователь гражданином
7. public boolean isEducationalRepresentative() – метод для проверки, является ли пользователь представителем образовательного учреждения
8. public boolean isMedicalRepresentative() – метод для проверки, является ли пользователь представителем медицинского учреждения
9. public String toString() – метод для преобразования объекта в строку
10. public boolean equals(Object obj) – метод для проверки на равенство
11. public int hashCode() – метод для получения хэш-кода объекта
12. public enum UserType – существующие типы пользователей
    * 1. **Интерфейс InstitutionRepresentative**

Интерфейс InstitutionRepresentative - данный интерфейс определяет базовые методы для представителей учреждений.

1. public Institution getInstitution() – метод для получения учреждения представителя
2. public boolean isApproved() – метод для проверки подтверждён ли аккаунт представителя
3. public void approve(User user) throws NoRightsException – метод для подтверждения аккаунта администратором
4. public boolean addFeedback(String text) throws NoRightsException – метод для добавления отзыва или ответа к отзывам учреждения
5. public boolean addFeedbackTo(String text, User userTo) throws NoRightsException – метод для добавления отзыва или ответа, адресованного определенному пользователю, к отзывам учреждения

**6.1.4. Класс MedicalRepresentative**

Класс MedicalRepresentative – класс представителя медицинского учреждения, он реализует интерфейс InstitutionRepresentative.

1. public void addDoctor(Doctor doctor) throws NoRightsException – метод для добавления доктора к учреждения
2. public void removeDoctor(Doctor doctor) throws NoRightsException – метод для удаления доктора из учреждения
3. public void addTickets(Doctor doctor, Date start, Date end, int intervalMinutes) throws InvalidTicketsDatesException, NoRightsException – метод для создания талонов между начальной и конечной датой с заданным в минутах интервалом
4. public void addTicket(Doctor doctor, Date date) throws NoRightsException, InvalidTicketsDatesException – метод для создания талона с определенным временем
5. public void deleteTicket(Ticket ticket) throws NoRightsException – метод для удаления талона
6. public void deleteTickets(Doctor doctor, Date date) throws NoRightsException – метод для удаления талонов у доктора в заданный день
7. public void confirmVisit(Ticket ticket, String summary) throws NoRightsException – метод для подтверждения того, что гражданин посетил врача
8. public MedicalInstitution editInstitution(String title, String city, String district, String telephone, String fax, String address) – метод для изменения основной информации об учреждении

**6.1.5. Класс EducationalRepresentative**

Класс EducationalRepresentative – класс представителя образовательно учреждения, он реализует интерфейс InstitutionRepresentative.

1. public boolean acceptEduRequest(EduRequest request) throws NoRightsException – метод для подтверждения учреждением заявки на обучение
2. public void makeAppointment(EduRequest request, Date date) throws NoRightsException, InvalidAppointmentDateException – метод для назначения даты и времени личной встречи на заявку об обучении
3. public void refuseEduRequest(EduRequest request) throws NoRightsException – метод для отказа в приёме ребёнка на обучение
4. public boolean makeChildEnrolled(EduRequest request) throws NoRightsException, NoFreeSeatsException – метод для установки у заявки статуса, соответствующему тому, что ребёнок зачислен в образовательное учреждение
5. public EducationalInstitution editInstitution(String title, String city, String district, String telephone, String fax, String address, Map<Integer, Integer> seats, Map<Integer, Integer> busySeats) – метод для изменения информации об образовательном учреждении
6. public void setSeats(int classNumber, int seats, int busySeats) throws InvalidDataForSavingSeatsException – метод для установки количества свободных и занятых мест на приём в заданный класс

**6.1.6. Класс Doctor**

Класс Doctor – класс врача медицинского учреждения, он реализует интерфейс InstitutionRepresentative.

1. public String getPosition() – метод для получения позиции врача
2. public String getSummary() – метод для получения общей информации о враче
3. public void addTickets(Date start, Date end, int intervalMinutes) throws InvalidTicketsDatesException, NoRightsException – метод для создания талонов между начальной и конечной датой с заданным в минутах интервалом
4. public void addTicket(Date date) throws NoRightsException, InvalidTicketsDatesException – метод для создания талона с определенным временем
5. public void deleteTicket(Ticket ticket) throws NoRightsException – метод для удаления талона
6. public void confirmVisit(Ticket ticket, String summary) throws NoRightsException - метод для подтверждения того, что гражданин посетил врача

**6.1.7. Класс Citizen**

Класс Citizen – класс гражданина.

1. public List<EduRequest> getEduRequests() – метод для получения всех заявок в образовательные учреждения
2. public Map<String, Child> getChilds() – метод для получения всех детей гражданина
3. public List<Ticket> getTickets() – метод для получения всех талонов гражданина
4. public String getPolicy() – метод для получения страхового полиса
5. public String getPassport() – метод для получения номера паспорта
6. public Date getBirthDate() – метод для получения даты рождения гражданина
7. public void addChild(Child child) – метод для добавления ребёнка в список детей
8. public void addTicket(Ticket ticket) – метод для добавления талона в список талонов
9. public void addEduRequest(EduRequest request) – метод для добавления заявки в список заявок
10. public boolean createChildInfo(String fullName, String birthCertificate, Date birthDate) – метод для добавления информации о новом ребёнке
11. public Child getChild(String birthCertificate) – метод для получения ребёнка по свидетельству о рождении
12. public boolean removeChildInfo(Child child) throws NoRightsException – метод для удаления информации о ребёнка
13. public EduRequest createEduRequest(Child child, EducationalInstitution institution, int classNumber) throws NoRightsException, NoFreeSeatsException – метод для создания новой заявки в образовательное учреждение
14. public void removeEduRequest(EduRequest request) throws NoRightsException – метод для удаления заявки в образовательное учреждение
15. public boolean acceptEduRequest(EduRequest request) throws NoRightsException – метод для подтверждения заявки гражданином
16. public boolean acceptTicket(Ticket ticket) – метод для того, чтобы пользователь забрал себе талон
17. public boolean acceptTicketForChild(Ticket ticket, Child child) throws NoRightsException - метод для того, чтобы пользователь забрал талон для своего ребёнка
18. public void cancelTicket(Ticket ticket) throws NoRightsException – метод для отказа от талона
19. public void removeTicket(Ticket ticket) throws NoRightsException – метод для удаления талона из списка талонов

**6.1.8. Класс Administrator**

Класс Administrator – класс администратора.

1. public EducationalInstitution addEducationalInstitution(String title, String city, String district, String telephone, String fax, String address, Map<Integer, Integer> seats, Map<Integer, Integer> busySeats) – метод для добавления нового образовательного учреждения
2. public EducationalInstitution addEducationalInstitution(String title, String city, String district, String telephone, String fax, String address) – метод для добавления нового образовательного учреждения
3. public MedicalInstitution addMedicalInstitution(String title, String city, String district, String telephone, String fax, String address) – метод для добавления нового медицинского учреждения
4. public void approveInstitutionRepresentative(InstitutionRepresentative representative) throws NoRightsException – метод для подтверждения аккаунта представителя учреждения

**6.2. Пакет entities**

**6.2.1. Класс Child**

Класс Child – класс с информацией о ребёнке гражданина.

1. public Citizen getParent() – метод для получения родителя ребёнка
2. public String getFullName() – метод для получения полного имени ребёнка
3. public String getBirthCertificate() – метод для получения свидетельства о рождении
4. public Date getBirthDate() – метод для получения даты рождения ребёнка
5. public void setParent(Citizen parent) – метод для установки родителя ребёнка
6. public void setId(int id) - метод устанавливает ID ребёнка
7. public int getId() - метод для получения ID ребёнка
8. public boolean equals(Object obj) – метод для проверки на равенство
9. public int hashCode() – метод для получения хэш-кода объекта

**6.2.2. Класс EduRequest**

Класс EduRequest – класс для представления заявки на обучение в образовательном учреждении.

1. public enum Status - возможные статусы заявки
2. public boolean isUpdated() – метод для проверки, была ли сейчас обновлена заявка
3. public void resetUpdated() – метод для сброса пометки об обновлении
4. public void setParent(Citizen parent) – метод для установки родителя
5. public void setInstitution(EducationalInstitution institution) 0 метод дляустановки образовательного учреждения
6. public void setId(int id) - метод устанавливает ID заявки
7. public int getId() - метод для получения ID заявки
8. public void changeStatus(Status status) – метод для изменения статуса заявки
9. public void makeAppointment(Date appointment) – метод для назначения даты и времени личной встречи
10. public boolean isPassedAppointment() – метод для проверки того, прошло ли уже время личной встречи
11. public EducationalInstitution getInstitution() – метод для получения образовательного учреждения
12. public Status getStatus() – метод для получения статуса заявки
13. public boolean isAcceptedByInstitution() – метод для проверки, подтверждена ли заявка учреждением
14. public boolean isAcceptedByParent() – метод для проверки, подтверждена ли заявка родителем ребёнка
15. public boolean isOpened() – метод для проверки, открыта ли заявка
16. public boolean isChildEnrolled() – метод для проверки, зачислен ли ребёнок в образовательное учреждение
17. public Child getChild() – метод для получения ребёнка
18. public Citizen getParent() – метод для получения родителя ребёнка
19. public Date getCreationDate() – метод для получения даты создания заявки
20. public Date getAppointment() – метод для получения времени личной встречи
21. public int getClassNumber() – метод для получения номера класса
22. public boolean equals(Object obj) – метод для проверки на равенство
23. public int hashCode() – метод для получения хэш-кода объекта

**6.2.3. Класс Feedback**

Класс Feedback – представляет отзыв, вопрос или ответ, относящийся к учреждению.

1. public void setId(int id) - метод устанавливает ID отзыва
2. public int getId() - метод для получения ID отзыва
3. public Date getDate() – метод для получения даты добавления отзыва
4. public Institution getInstitution() – метод для получения учреждения
5. public User getUser() – метод для получения пользователя, добавившего отзыв
6. public User getToUser() - метод для получения пользователя, которому адресован отзыв ( необязательно)
7. public String getText() – метод для получения текста отзыва
8. public String toString() – метод для преобразования объекта к строке
9. public boolean equals(Object obj) – метод для проверки на равенство
10. public int hashCode() – метод для получения хэш-кода объекта

**6.2.4. Класс Notification**

Класс Notification – представляет важное для пользователя уведомление об изменения.

1. public void setOwner(User owner)
2. public void setId(int id) - метод устанавливает ID уведомления
3. public int getId() - метод для получения ID уведомления
4. public User getOwner() – метод для получения владельца уведомления
5. public String getNotification() – метод для получения текста уведомления
6. public Date getDate() – метод для получения даты создания уведомления
7. public String toString() – метод для преобразования в строку объекта
8. public boolean equals(Object obj) – метод для проверки на равенство
9. public int hashCode() – метод для получения хэш-кода объекта

**6.2.5. Класс Ticket**

Класс Ticket – класс, представляющий талон в медицинское учреждение.

1. public void setUser(Citizen user) – метод для установки гражданина
2. public void setId(int id) - метод устанавливает ID талона
3. public int getId() - метод для получения ID талона
4. public boolean isUpdated() – метод для проверки, был ли сейчас обновлен талон
5. public void resetUpdated() – метод для сброса пометки об обновлении
6. public boolean acceptTicket(Citizen user) – метод для получения талона гражданином
7. public boolean acceptTicket(Citizen user, Child child) – метод для получения талона гражданином для ребёнка
8. public void refuseTicket() – метод для отказа от талона
9. public void setVisited(boolean visited, String summary) – метод для установки флага, что было посещение и его результата
10. public Citizen getUser() – метод для получения гражданина
11. public Child getChild() – метод для получения ребёнка
12. public Doctor getDoctor() – метод для получения врача
13. public Date getDate() – метод для получения даты и времени талона
14. public String getSummary() – метод для получения описания результатов приёма
15. public boolean isVisited() – метод для проверки, было ли посещение врача
16. public boolean canBeRefused() – метод для проверки, можно ли отказаться от талона
17. public boolean isTicketForChild() – метод для проверки, взят ли талон для ребёнка
18. public MedicalInstitution getInstitution() – метод для получения медицинского учреждения
19. public boolean equals(Object obj) – метод для проверки на равенство
20. public int hashCode() – метод для получения хэш-кода объекта
21. **Слой источников данных**

Для хранения данных используется MySQL. Для подключения к базе данных в исходном коде используется синглтон-класс Gateway, который считывает из конфигурационного файла параметры и подключается к базе данных. Этот класс предоставляет всем остальным классам слоя источника данных доступ к БД.

Для создания слоя источников данных используется архитектурный паттерн "Преобразователь данных" (Data Mapper). Для классов слоя бизнес-логики были созданы преобразователи. Для всех типов пользователей создан один преобразователь UserMapper. Для каждого из оставшихся классов создан отдельный преобразователь. Преобразователи данных позволяют иерархически считывать и обновлять данные в БД.

Каждый класс-преобразователь данных реализует интерфейс Mapper. Интерфейс UserMapperInterface имеет новый метод T findByLogin(String login) - найти пользователя по его уникальному логину.

Каждый преобразователь имеет свой локальный кэш объектов, извлеченных из БД. Если в нём есть необходимый объект, то возвращается ссылка на него, иначе происходит извлечение данных из БД. В любой момент времени в системе существует только один экземпляр какого-либо объекта. Преобразователи могут не обновлять данные в БД, к примеру, для неизменяемой сущности.

Все классы слоя источника данных расположены в пакете storage.

Используемые структурные паттерны источников данных:

Поле идентификации (Identity Field) - каждый класс имеет поле id, которое позволяет идентифицировать экземпляры объектов этого класса. Данное поле является автогенерируемым (средствами БД).

Отображение внешних ключей (Foreign Key Mapping) - внутренние ссылки объектов друг на друга отображаются в БД в качестве внешних ключей.

Шлюзом к слою источников данных является класс StorageRepository, который реализует паттерн "Репозиторий". Он преобразует все запросы к слою источников данных в запросы к преобразователям данных, а так же позволяет обрабатывать некоторые ошибки. В классе есть методы для получения объектов, также есть метод clear() для очистки всех кэшей преобразователей данных, update() для синхронизации системы с БД, drop() для загрузки и выполнения скрипта пересоздания БД.

1. **Сервисный слой**

Использован паттерн фасад, создан класс Facade, который реализует интерфейс FacadeInterface. Данный класс скрывает за собой слои бизнес-логики и хранения данных. Фасад работает только с примитивными типами данных: Integer и String. Так же создан класс Struct, который позволяет хранить строковые пары ключ-значение. Для удобной передачи всей информации об объектах.

1. **Слой представления**

Использован паттерн Model-View-Controller. Была использована библиотека JavaFX, было использована удобное средство Scene Builder. В качестве модели выступает сервисный слой приложения. В качестве представления файлы разметки пользовательского интерфейса JavaFX с расширением .fxml. Для каждого файла .fxml создан контроллер, который отвечает за связь модели и представления. Файлы разметки находятся в папке resources/fxml. Все контроллеры находятся в пакете controllers.

1. **Интеграция с сервисами**

Классы для сервисов расположены в пакете service.

* 1. **Предоставление информации с помощью собственного сервиса**

Созданный сервис предоставляет информацию о запрошенном образовательном учреждении в виде json-ответа. Для создания сервиса был реализован простой HTTP-сервер, запускаемый на порту 8080, который обрабатывает GET-запросы. Пример URI запроса к серверу: <http://localhost:8080/get?edu_inst_id=1>. Для отправки ответа пользователю в json-формате использовалась библиотека Gson.

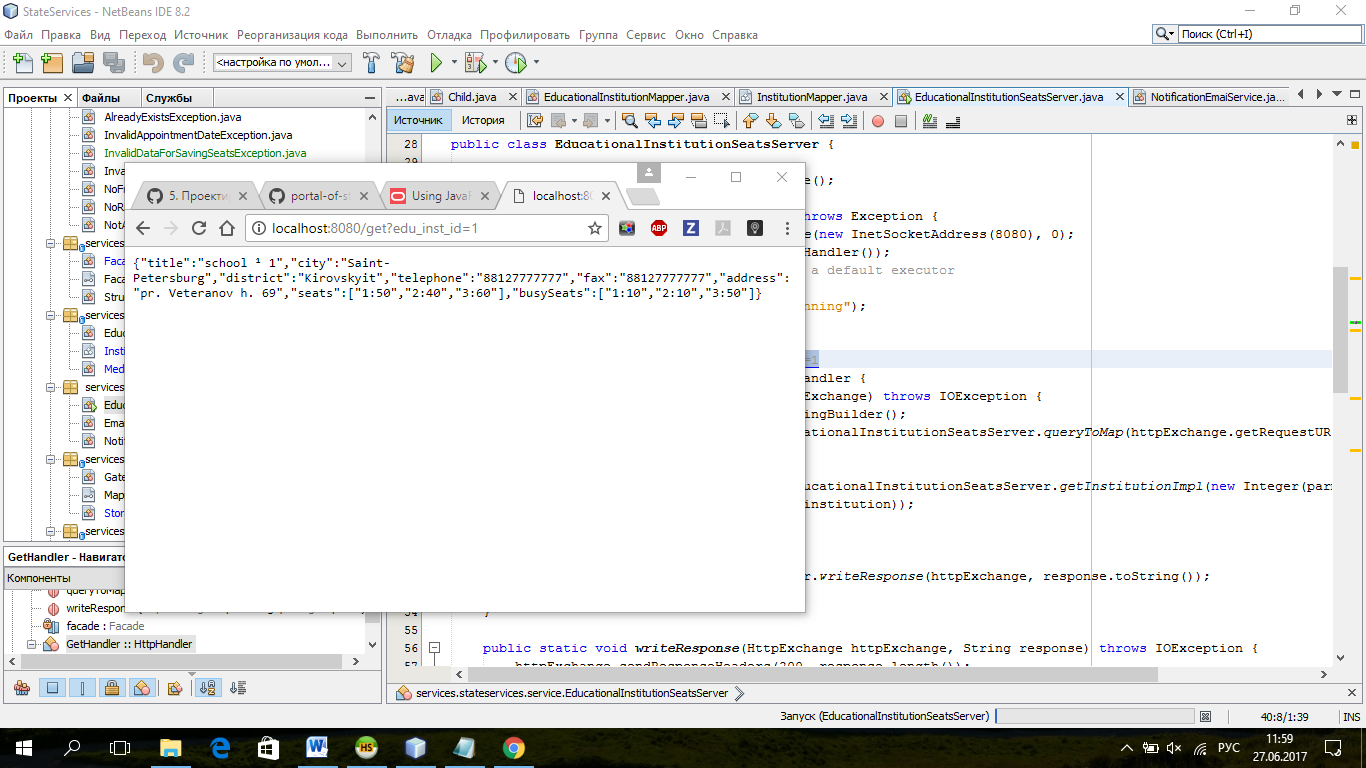


Рис.5. Работа собственного сервиса

* 1. **Обращение к внешнему сервису**

Внешний сервис будет использоваться для отправки писем с важными уведомлениями на почту пользователей. В случае успешной верификации адреса почты пользователя, ему отправляется письмо.

Рис. 6. Полученное уведомление на почте

1. **Тестирование**
   1. **Модульное тестирование**

Модульное тестирование приложения производилось автоматически с помощью библиотеки JUnit. Тесты располагаются в src/test/services/stateservices.

BPChildDoctorVisitTest – тесты записи и посещения врача ребёнком

BPDoctorVisitTest – тесты записи и посещения врача гражданином

BPEnrollChildTest – тест записи ребёнка в образовательное учреждение и его зачисление

* 1. **Функциональное тестирование**

Функциональное тестирование приложения производилось вручную через графический пользовательский интерфейс.

Рис. 7. Попытка добавить доктора с логином, который уже существует

**Вывод**

Был разработан портал государственных услуг. Для проектирования слоя бизнес-логики был использован паттерн модель предметной области. Для проектирования слоя источника данных были использованы паттерны: преобразователь данных и репозиторий. Для хранения данных использовалась MySQL. Для сервисного слоя был использован паттерн фасад. Для слоя представления был использован паттерн Model-View-Controller. Использовалась библиотека JavaFX. Для тестирования использовалась библиотека JUnit. Приложение требует доработки пользовательского интерфейса и требует объёмного тестирования.

**Список используемых источников**

* + 1. <http://www.yworks.com/products/yed/download#ReleaseNotes> – Редактор yEd
    2. <https://www.gliffy.com/> - Online-редактор gliffy (для диаграмм-последовательностей)

## <http://tiger.ftk.spbstu.ru/trac/edu-arh-2017>

## Литература

* Конспект лекций ([слайды](http://kspt.ftk.spbstu.ru/course/architecture))
* UML. Основы. 2-е издание. М. Фаулер, К. Скотт
* Предметно-ориентирование проектирование. Э. Эдванс
* [Архитектура корпоративных программных приложений. М. Фаулер](http://books.google.ru/books?id=MdMM1ZrICnsC)
* Приемы объектно-ориентированного проектирования. Э. Гамма, Р. Хелм и др.
* Шаблоны интеграции корпоративных приложений. Г. Хоп, Б. Вульф и др.

## On-line каталоги типовых решений

* <http://citforum.ru/SE/project/pattern/> (RU)
* <http://ooad.asf.ru/Patterns.aspx> (RU)
* <http://www.oodesign.com/> (EN)
* <http://design-pattern.ru/> (RU)
* <http://thierryroussel.free.fr/java/books/martinfowler/www.martinfowler.com/isa/> (EN)
* <http://martinfowler.com/eaaCatalog/> (EN)
* <http://www.enterpriseintegrationpatterns.com/toc.html> (EN)

## Проектные хостинги

* <http://bitbucket.org/>
* <http://github.com/>

## UML-редакторы

### online

* [CodeUML online](http://www.codeuml.com/)
* [yUML online](http://yuml.me/)
* [Online Diagram](https://www.draw.io/)
* [gliffy](https://www.gliffy.com/)

### desktop

* [Dia](http://live.gnome.org/Dia)
* [yEd](http://www.yworks.com/en/products_yed_about.html)
* [PlantUML](http://plantuml.sourceforge.net/)

**-Формат ввода номера свидетельства о рождении, выданного в Российской Федерации (СР):**  
Серия: 2 римские цифры (в латинском регистре), две буквы кириллицей, номер: шесть цифр.   
Без тире и пробелов.