Санкт-Петербургский Государственный Политехнический Университет

Институт информационных технологий и управления

Кафедра Компьютерных Систем и Программных Технологий

**Курсовой проект**

**Отчет по дисциплине «Программное обеспечение распределенных вычислительных систем»**

**По теме «Портал государственных услуг»**

**Работу выполнила студентка группы № 23541/3**

Фомина М. А. \_\_\_\_\_\_\_\_\_\_\_

**Работу принял преподаватель:**

Стручков И.В. \_\_\_\_\_\_\_\_\_\_\_

.

Санкт-Петербург

2018

**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, врач.

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

**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**

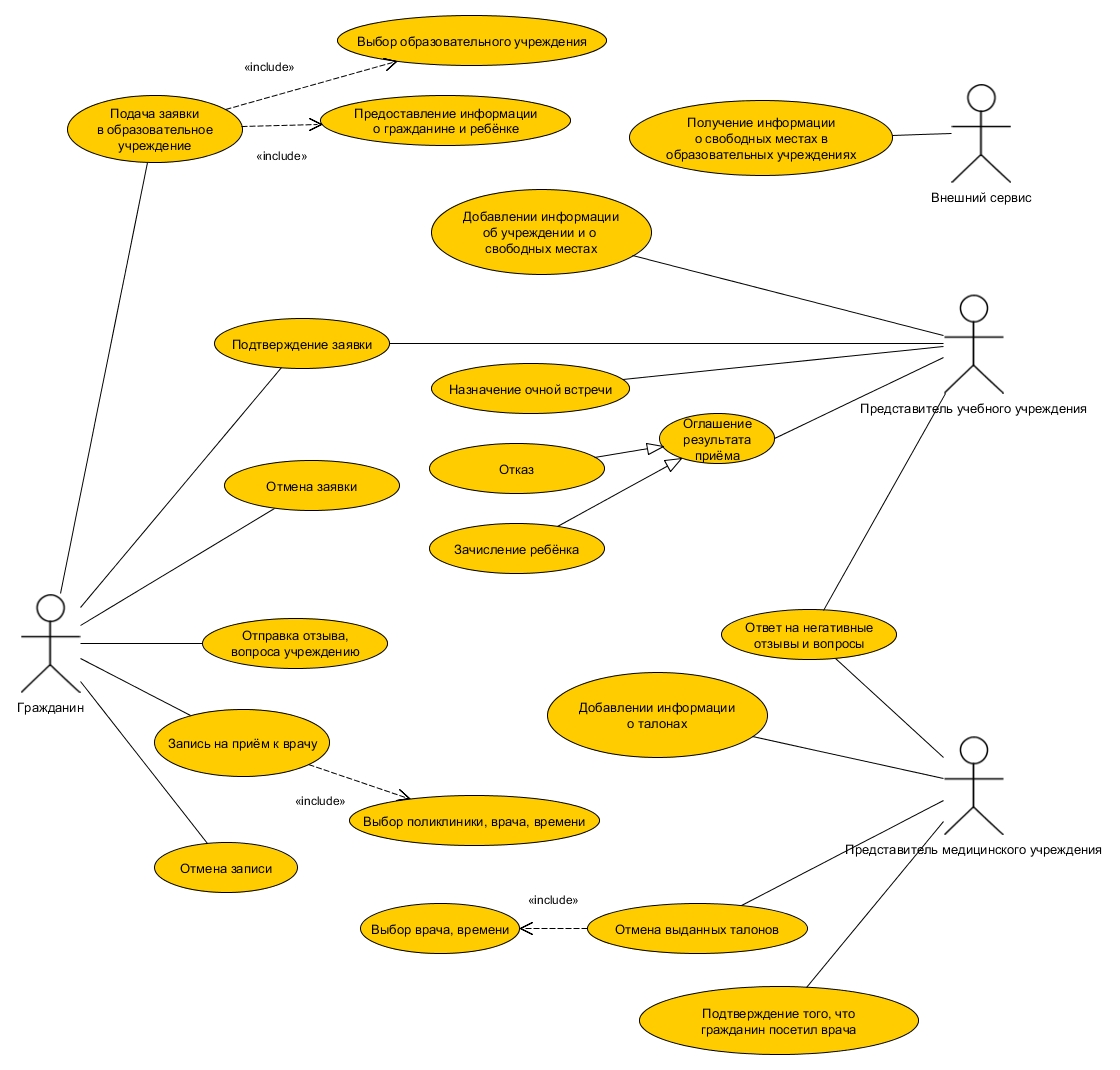


Рис. 1. Диаграмма прецедентов для всех ролей

**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**

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

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

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

**4.1. Запись ребёнка в школу**

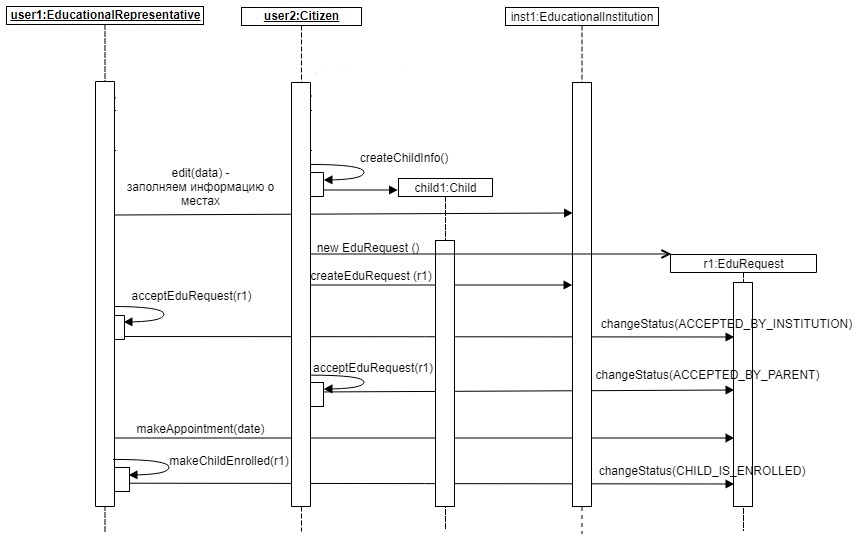


Рис. 2. Диаграмма последовательности для бизнес-процесса записи ребёнка в школу

**4.2. Запись на приём к врачу**

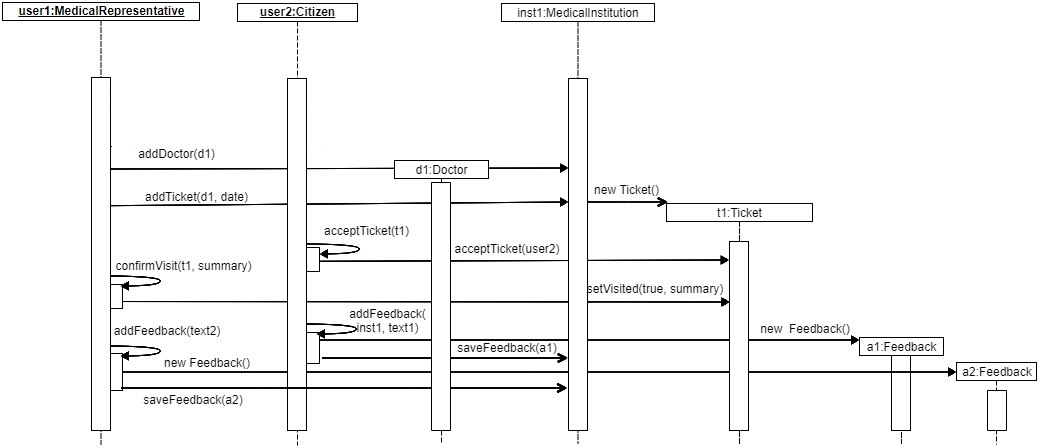


Рис. 3. Диаграмма последовательности для бизнес-процесса записи на приём к врачу

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

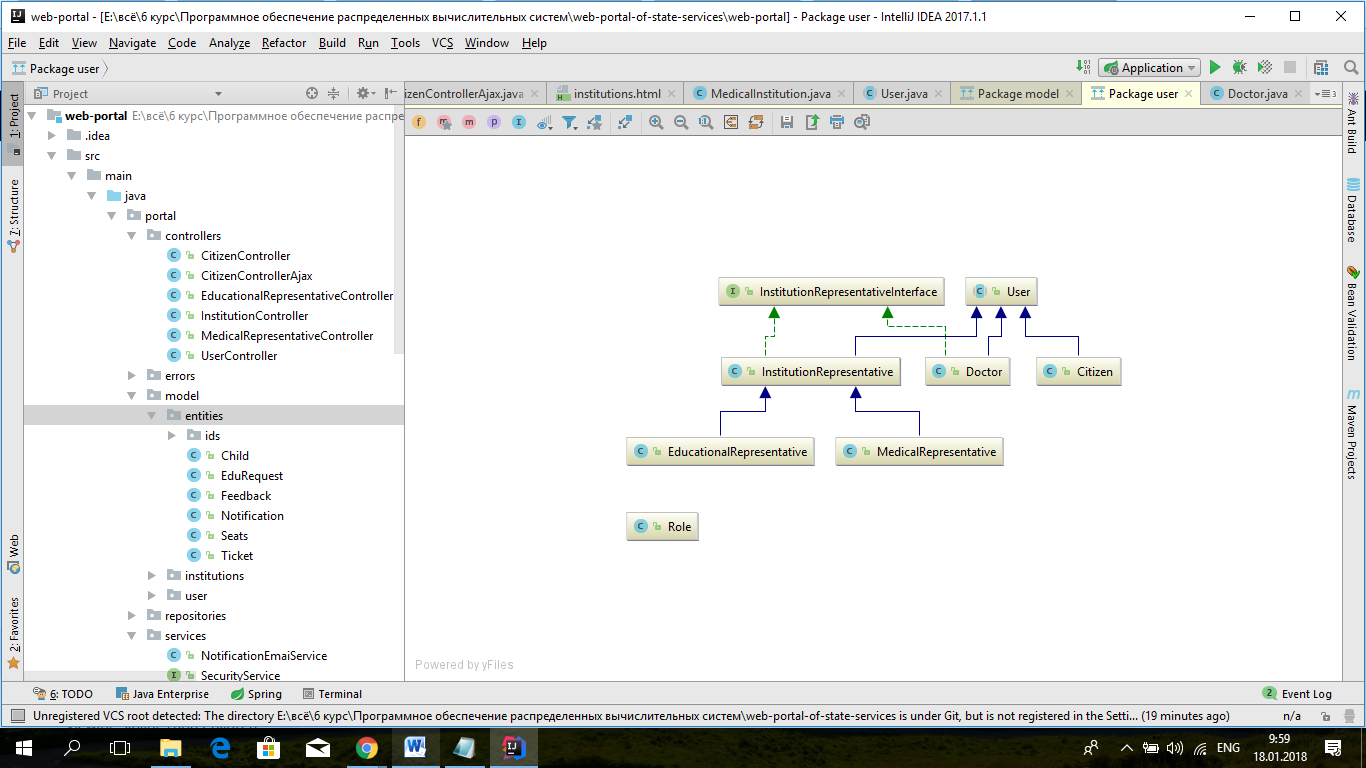


Рис. 4. Пакет user

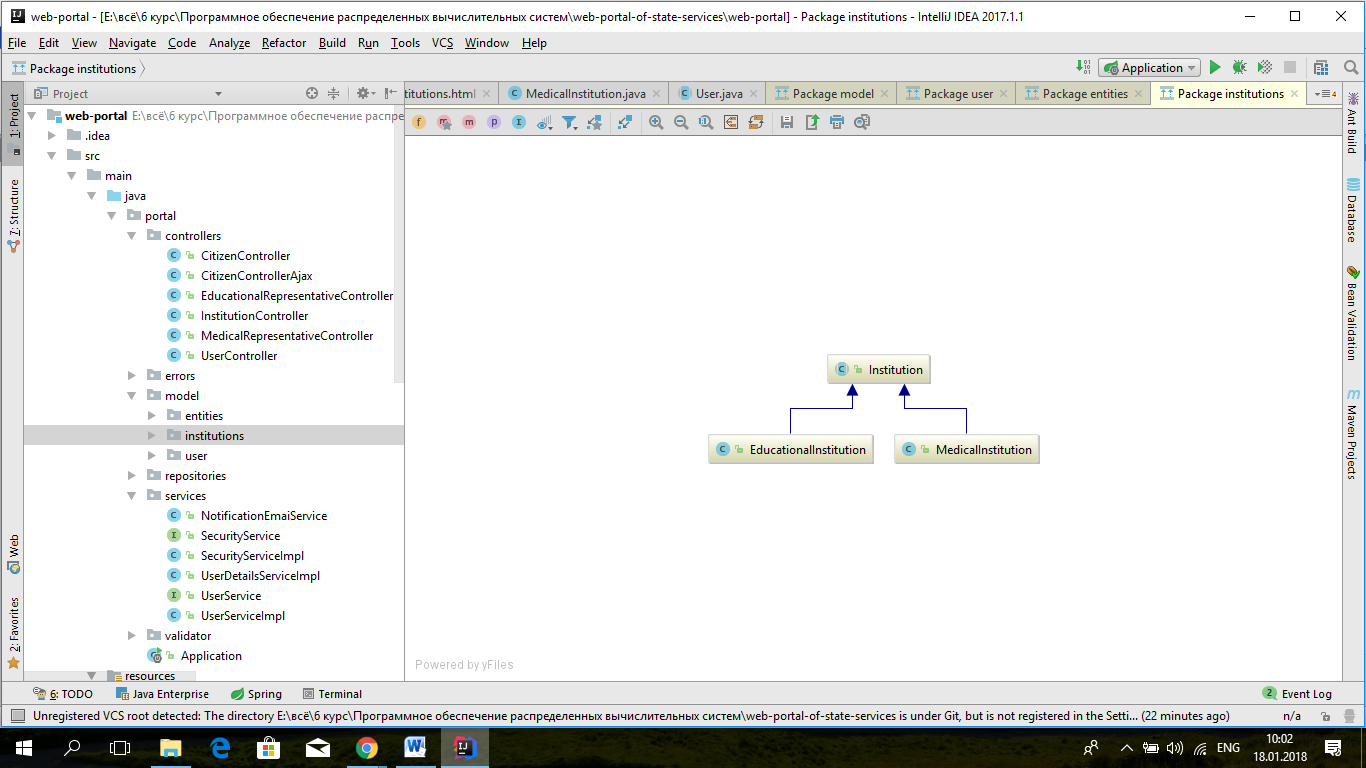


Рис. 5. Пакет institutions

* 1. **Пакет user**
     1. **Класс User**

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

1. public boolean isDoctor() – метод для проверки, является ли пользователь доктором
2. public boolean isAdministrator() – метод для проверки, является ли пользователь администратором
3. public boolean isCitizen() – метод для проверки, является ли пользователь гражданином
4. public boolean isEducationalRepresentative() – метод для проверки, является ли пользователь представителем образовательного учреждения
5. public boolean isMedicalRepresentative() – метод для проверки, является ли пользователь представителем медицинского учреждения
6. void addNotification(String notification) – метод для добавления нового уведомления
7. boolean addFeedback(Institution institution, String text) throws NoRightsException – для добавления отзыва или вопроса учреждению
8. boolean addFeedbackTo(Institution institution, String text, User userTo) throws NoRightsException – метод для добавления отзыва или вопроса учреждению, адресованного конкретному пользователю
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
    * 1. **Интерфейс InstitutionRepresentative**

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

1. public void approve(User user) throws NoRightsException – метод для подтверждения аккаунта администратором
2. public boolean addFeedback(String text) throws NoRightsException – метод для добавления отзыва или ответа к отзывам учреждения
3. public boolean addFeedbackTo(String text, User userTo) throws NoRightsException – метод для добавления отзыва или ответа, адресованного определенному пользователю, к отзывам учреждения

**6.1.3. 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 учреждению
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

**6.1.4. 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

**6.1.5. Класс Doctor**

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

1. 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
2. public void addTicket(Date date) throws NoRightsException, InvalidTicketsDatesException – метод для создания талона с определенным временем
3. public void deleteTicket(Ticket ticket) throws NoRightsException – метод для удаления талона
4. public void confirmVisit(Ticket ticket, String summary) throws NoRightsException - метод для подтверждения того, что гражданин посетил врача

**6.1.6. Класс Citizen**

Класс Citizen – класс гражданина.

1. public void addChild(Child child) – метод для добавления ребёнка в список детей
2. public void addTicket(Ticket ticket) – метод для добавления талона в список талонов
3. public void addEduRequest(EduRequest request) – метод для добавления заявки в список заявок
4. public boolean createChildInfo(String fullName, String birthCertificate, Date birthDate) – метод для добавления информации о новом ребёнке
5. public Child getChild(String birthCertificate) – метод для получения ребёнка по свидетельству о рождении
6. public boolean removeChildInfo(Child child) throws NoRightsException – метод для удаления информации о ребёнка
7. public EduRequest createEduRequest(Child child, EducationalInstitution institution, int classNumber) throws NoRightsException, NoFreeSeatsException – метод для создания новой заявки в образовательное учреждение
8. public void removeEduRequest(EduRequest request) throws NoRightsException – метод для удаления заявки в образовательное учреждение
9. public boolean acceptEduRequest(EduRequest request) throws NoRightsException – метод для подтверждения заявки гражданином
10. public boolean acceptTicket(Ticket ticket) – метод для того, чтобы пользователь забрал себе талон
11. public boolean acceptTicketForChild(Ticket ticket, Child child) throws NoRightsException - метод для того, чтобы пользователь забрал талон для своего ребёнка
12. public void cancelTicket(Ticket ticket) throws NoRightsException – метод для отказа от талона
13. public void removeTicket(Ticket ticket) throws NoRightsException – метод для удаления талона из списка талонов

**6.2. Пакет entities**

**6.2.1. Класс EduRequest**

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

1. public boolean isPassedAppointment() – метод для проверки того, прошло ли уже время личной встречи
2. public boolean isAcceptedByInstitution() – метод для проверки, подтверждена ли заявка учреждением
3. public boolean isAcceptedByParent() – метод для проверки, подтверждена ли заявка родителем ребёнка
4. public boolean isOpened() – метод для проверки, открыта ли заявка
5. public boolean isChildEnrolled() – метод для проверки, зачислен ли ребёнок в образовательное учреждение
6. public boolean equals(Object obj) – метод для проверки на равенство
7. public int hashCode() – метод для получения хэш-кода объекта

**6.2.2. Класс Ticket**

Класс Ticket – класс, представляющий талон в медицинское учреждение.

1. public boolean acceptTicket(Citizen user) – метод для получения талона гражданином
2. public boolean acceptTicket(Citizen user, Child child) – метод для получения талона гражданином для ребёнка
3. public void refuseTicket() – метод для отказа от талона
4. public void setVisited(boolean visited, String summary) – метод для установки флага, что было посещение и его результата
5. public boolean canBeRefused() – метод для проверки, можно ли отказаться от талона
6. public boolean isTicketForChild() – метод для проверки, взят ли талон для ребёнка
7. public boolean equals(Object obj) – метод для проверки на равенство
8. public int hashCode() – метод для получения хэш-кода объекта

**6.3. Пакет institutions**

**6.3.1. Класс Institution**

Класс Institution – абстрактный класс, предоставляющий методы для учреждений. Является родительским классом для всех учреждений.

1. public void edit(String title, String city, String district, String telephone, String fax, String address) – метод для редактирования информации об учреждении
2. public abstract boolean saveFeedback(Feedback feedback) throws NoRightsException – метод для сохранения нового отзыва
3. public void addFeedback(Feedback feedback) – метод для добавления отзыва в список отзывов
4. public boolean equals(Object obj) – метод для проверки на равенство
5. public int hashCode() – метод для получения хэш-кода объекта

**6.3.2. Класс MedicalInstitution**

Класс MedicalInstitution – класс медицинского учреждения.

1. public void addDoctor(Doctor doctor) throws NoRightsException – метод для добавления доктора в список докторов учреждения
2. public void removeDoctor(Doctor doctor) throws NoRightsException - метод для удаления доктора из списка докторов учреждения
3. public void addTicket(Ticket ticket) throws NoRightsException – метод для добавления талона
4. public void removeTicket(Ticket ticket) – метод для удаления талона
5. public List<Ticket> getTickets(Doctor doctor) throws NoRightsException – метод для получения талонов конкретного доктора
6. public boolean saveFeedback(Feedback feedback) throws NoRightsException – метод для сохранения нового отзыва учреждения

**6.3.3. Класс EducationalInstitution**

Класс EducationalInstitution – класс образовательного учреждения.

1. public void edit(String title, String city, String district, String telephone, String fax, String address, Map<Integer, Integer> seats, Map<Integer, Integer> busySeats) - метод для редактирования образовательного учреждения, позволяет сразу изменить полную информацию об общем и занятом количестве мест
2. public void setSeats(int classNumber, int seats, int busySeats) – метод для установки информации о местах для определенного класса
3. public void setSeats(Map<Integer, Integer> seats, Map<Integer, Integer> busySeats) – метод для установки информации о местах для всех классов
4. public void addEduRequest(EduRequest request) throws NoRightsException – метод для добавления заявки в список заявок
5. public boolean createEduRequest(EduRequest request) throws NoRightsException – метод для подачи заявки на обучение в образовательном учреждении
6. public void removeEduRequest(EduRequest request) – метод для удаления заявки на обучение
7. public boolean saveFeedback(Feedback feedback) throws NoRightsException – метод для сохранения отзыва об учреждении
8. **Слой источников данных**

В качестве СУБД используется MySQL. Для работы с базой данных был использован Java Persistance API, в частности его реализация в библиотеке Hibernate. Описание сущностей производится с помощью специальных аннотаций JPA. Пример сущностного класса Feedback приведен в листинге 1.

Листинг 1. Описание сущности Feedback

|  |
| --- |
| @Entity @Table(name = **"feedbacks"**) @NamedQuery(name = **"Feedback.deleteFeedbacksForUser"**, query = **"delete from Feedback f where f.user.id = ?1 or f.toUser.id = ?1"**) **public class** Feedback {  @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  **private** Long **id**;  @ManyToOne  @JoinColumn(name = **"user"**)  **private** User **user**;  @Column(name = **"created"**)  **private** Date **date**;  @ManyToOne  @JoinColumn(name = **"institution\_id"**)  **private** Institution **institution**;  @Column(name = **"feedback\_text"**)  **private** String **text**;  @ManyToOne  @JoinColumn(name = **"to\_user"**)  **private** User **toUser** = **null**;   **public** Feedback() {}   **public** Feedback(Date date, User user, Institution institution, String text) {  **this**.**date** = date;  **this**.**user** = user;  **this**.**institution** = institution;  **this**.**text** = text;  }    **public** Feedback(Date date, User user, Institution institution, String text, User toUser) {  **this**.**date** = date;  **this**.**user** = user;  **this**.**institution** = institution;  **this**.**text** = text;  **this**.**toUser** = toUser;  }   **public** Long getId() {  **return id**;  }   **public void** setId(Long id) {  **this**.**id** = id;  }    **public** Date getDate() {  **return date**;  }   **public void** setDate(Date date) { **this**.**date** = date; }   **public** Institution getInstitution() { **return institution**; }   **public void** setInstitution(Institution institution) { **this**.**institution** = institution; }   **public** User getUser() {  **return user**;  }   **public void** setUser(User user) { **this**.**user** = user; }   **public** User getToUser() {  **return toUser**;  }   **public void** setToUser(User toUser) { **this**.**toUser** = toUser; }    **public** String getText() {  **return text**;  }   **public void** setText(String text) { **this**.**text** = text; }    @Override  **public** String toString() {  StringBuilder builder = **new** StringBuilder();  builder.append(**"["**);  builder.append(**date**);  builder.append(**"] "**);  **if** (**user** != **null**) {  builder.append(**user**.getFullName());  }  builder.append(**": "**);  **if** (**toUser** != **null**) {  builder.append(**toUser**).append(**", "**);  }  builder.append(**text**);  **return** builder.toString();  }    @Override  **public boolean** equals(Object obj) {  **if** ( (obj == **null**) || (obj.getClass() != **this**.getClass()) ) **return false**;  Feedback other = (Feedback) obj;  **return** (**id** == other.getId() &&   **text**.equals(other.getText()) &&   (**user** != **null** ? **user**.equals(other.getUser()) : other.getUser() == **null**) &&   (**institution** != **null** ? **institution**.equals(other.getInstitution()) : other.getInstitution() == **null**) &&  (**toUser** == **null** ? **toUser**.equals(other.getToUser()) : other.getToUser() == **null**) &&   (**date** != **null** ? **date**.equals(other.getDate()) : other.getDate() == **null**)  );  }    @Override  **public int** hashCode() {  **return** Long.*toString*(**id**).hashCode();  } } |

Для каждой сущности был реализован свой репозиторий. Репозитории были реализованы с помощью библиотеки Spring Data. Данная библиотека позволяет описывать только интерфейс, и по названиям методов этого интерфейса умеет автоматически генерировать его реализацию.

Листинг 2. Репозиторий FeedbackRepository

|  |
| --- |
| @Transactional **public interface** FeedbackRepository **extends** CrudRepository<Feedback, Long> {  Feedback findById(Long id);  @Modifying  **void** deleteFeedbacksForUser(Long userId); } |

1. **Слой представления**

Для слоя представления была выбрана технология Spring MVC, которая позволяет реализовать паттерн Model-View-Controller. В качестве модели выступает слой бизнес-логики приложения. Представление реализовано с помощью JavaScript-функций и HTML-страниц. Рассмотрим более подробно контроллеры.

**UserController**

* [/registration] – метод для получения страницы регистрации
* [/registration], метод POST – метод для регистрации нового гражданина
* [/portal] – метод для перенаправления запроса после входа в систему в зависимости от роли пользователя
* [/login] – метод для входа в систему

**CitizenController**

* [/tickets] – метод для получения страницы с талонами
* [/profile], [/profile/update] – методы для получения страницы профиля
* [/profile/update], метод POST – метод для изменения информации о пользователе
* [/requests] – метод для получения страницы с заявками в образовательное учреждение
* [/childs] – метод для получения страницы с информацией о детях гражданина

**CitizenControllerAjax**

* [/citizen/remove\_request] – метод для удаления заявки в образовательное учреждение
* [/citizen/add\_request] – метод для добавления заявки в образовательное учреждение
* [/citizen/accept\_request] – метод для подтверждения заявки в образовательное учреждение
* [/citizen/remove\_child] – метод для удаления информации о ребёнке
* [/citizen/take\_ticket] – метод для получения талона
* [/citizen/add\_child], метод POST – метод для добавления информации о ребёнке
* [/citizen/refuse\_ticket] – метод для отказа от талона

**EducationalRepresentativeController**

* [/edur/add\_seats\_info] , метод POST – метод для сохранения информации о местах
* [/edur/refuse\_request] – метод для отказа в приеме в образовательное учреждение
* [/edur/make\_appointment] – метод для назначения даты личной встречи
* [edur/accept\_request\_repr] – метод для подтверждения заявки в образовательное учреждение
* [/edur/enroll\_request] – метод для зачисления ребёнка в образовательное учреждение

**InstitutionController**

* [/get\_districts] – метод для получения списка районов
* [/institutionview] – метод для получения страницы учреждения
* [/get\_institutions] – метод для получения списка учреждений
* [/institutions] – метод для получения страницы со списком учреждений
* [/get\_cities] – метод для получения списка городов
* [/save\_institution\_info], метод POST – метод для изменения информации об учреждении
* [/add\_feedback] – метод для добавления отзыва об учреждении

**MedicalRepresentativeController**

* [/medr/remove\_doctor] – метод для удаления доктора
* [/medr/add\_tickets] – метод для добавления талонов
* [/medr/add\_doctor], метод POST – метод для добавления информации о докторе
* [/medr/set\_ticket\_visited] – метод для подтверждения посещения по талону
* [/medr/remove\_ticket] – метод для удаления талона
* [/medr/add\_ticket] – метод для добавления талона
* [/medr/remove\_tickets] – метод для удаления талонов

1. **Интеграция с сервисами**

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

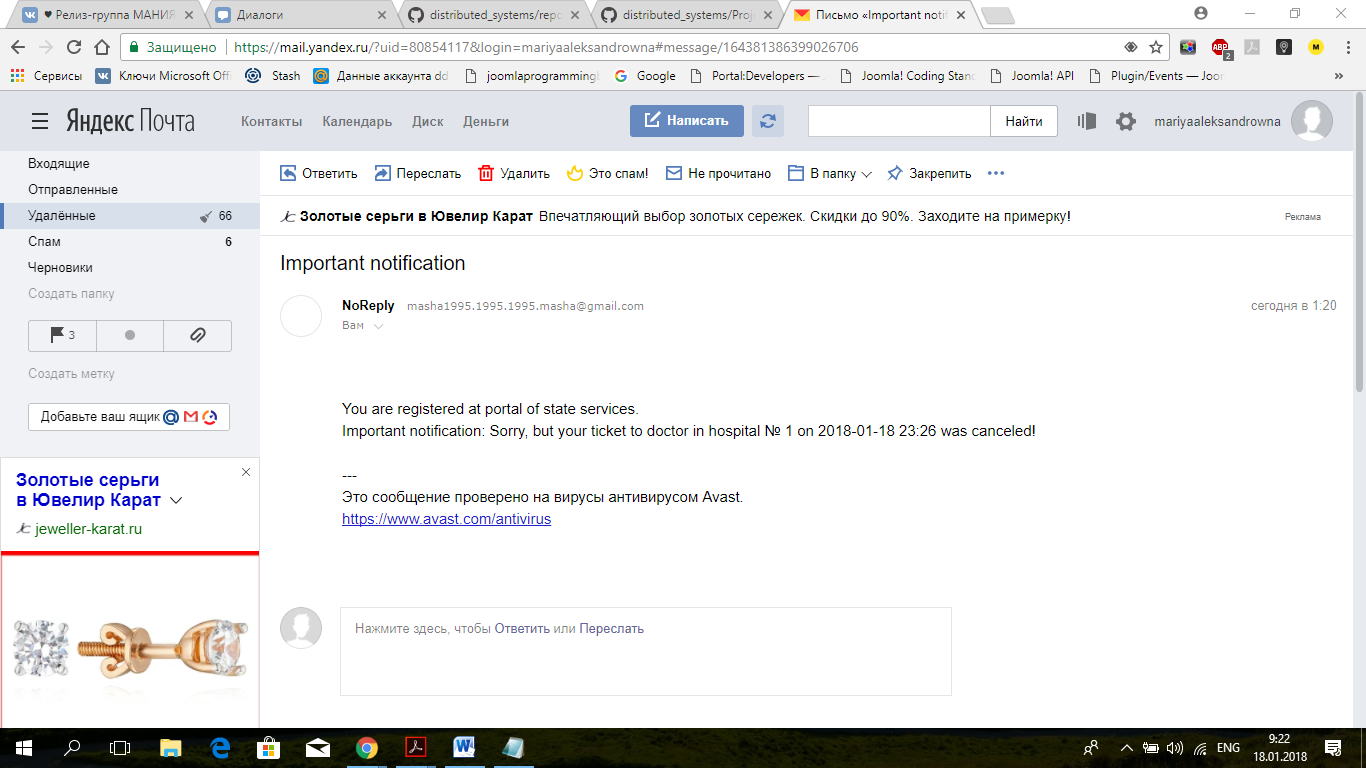


Рис. 6. Полученное уведомление на почте

1. **Тестирование**

Функциональное тестирование приложения производилось вручную через пользовательский интерфейс. Список проводимых тестов:

* Попытка добавить пользователя, который уже существует. Открываем окно регистрации нового пользователя, вводим данные пользователя. В поле **username** указываем логин уже зарегистрированного пользователя. Нажимаем кнопку **Submit**. В результате будет выведена ошибка о том, что пользователь с таким именем уже существует.

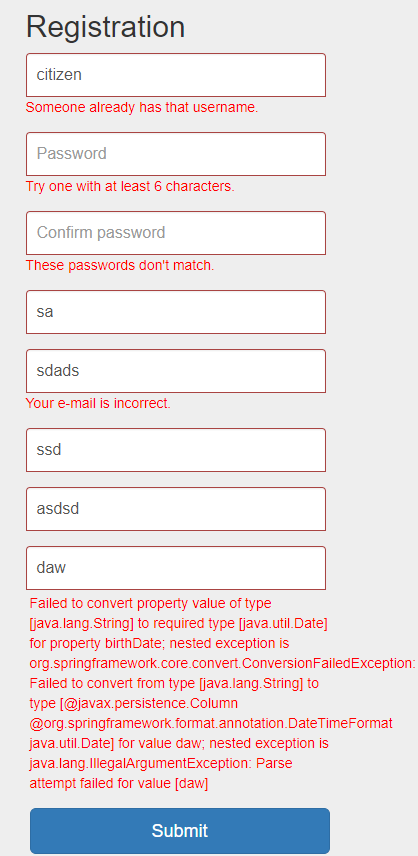


Рис. 7. Попытка регистрации пользователя с некорректными данными

* Регистрация нового пользователя. Открыть окно регистрации пользователя. Ввести уникальные данные пользователя. Нажать кнопку **Submit**. Регистрация должна пройти успешно. После этого гражданин будет перенаправлен на страницу своего профиля.
* Попытка войти в систему с неправильным логином или паролем. В окне входа в систему ввести неправильные данные пользователя (логин или пароль). Нажать кнопку **Login**. Должно появиться окно с сообщением о том, что данные некорректны.
* Вход в систему. В окне входа в систему ввести правильные данные пользователя. Нажать кнопку **Login**. Должно открыться страница для указанного пользователя.
* Проверка вывода ошибки при попытке представителем образовательного учреждения сохранить некорректную информацию о местах. Проверка того, что представитель образовательного учреждения не может добавить некорректную информацию о занятых и свободных местах:

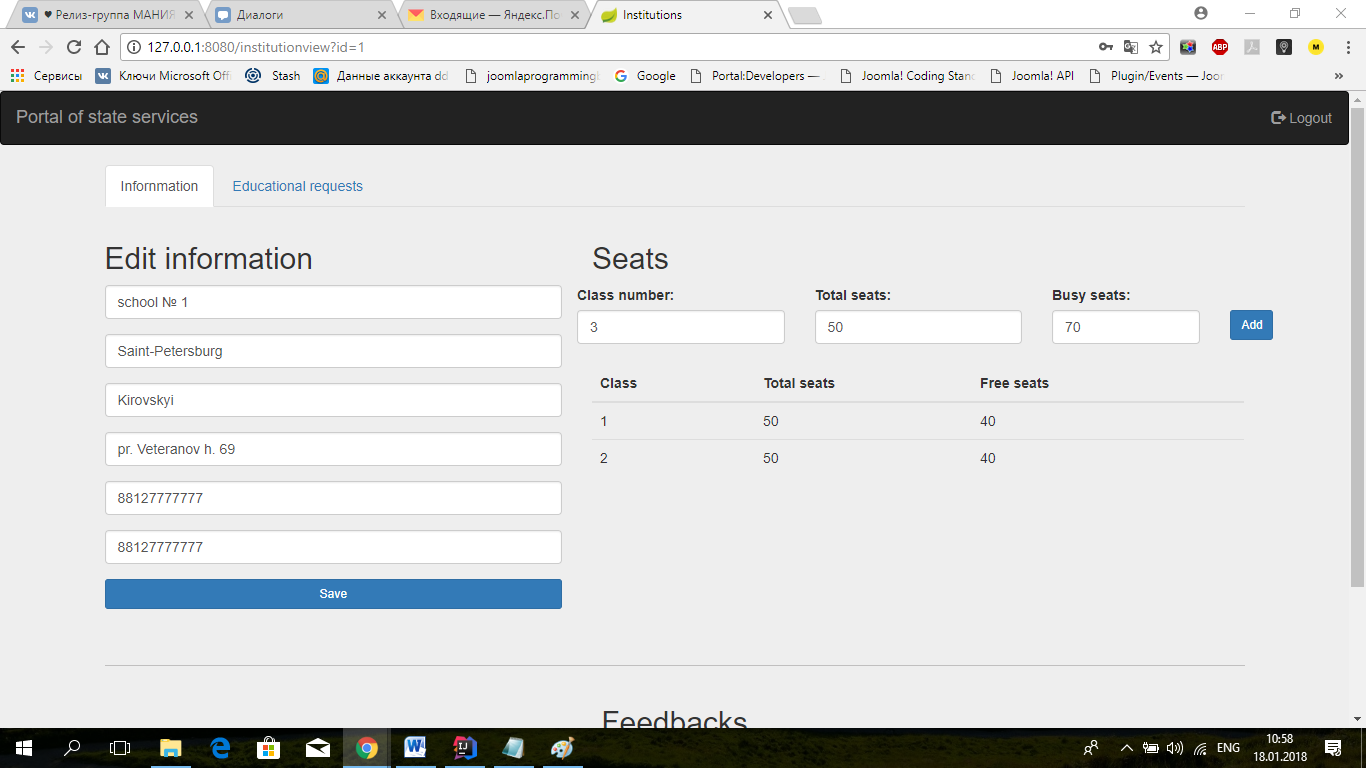


Рис. 8. Неправильно введена информация о местах, занятое количество мест не должно превышать общее количество мест

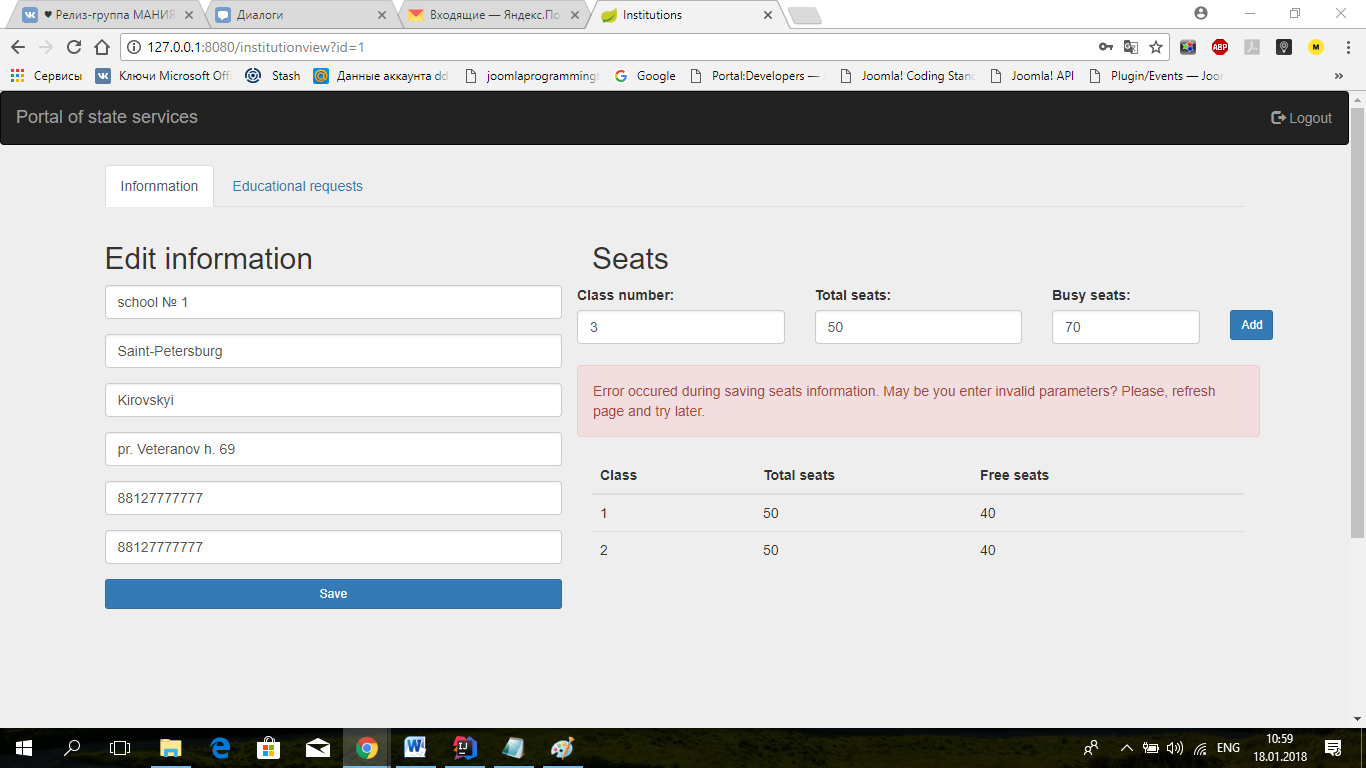


Рис. 9. Некорректные данные о местах не были сохранены, вывелась ошибка

Все описанные тесты успешно выполняются. Все полученные результаты совпадают с ожидаемыми результатами.

1. **Инструкция системного администратора**

* Скачать и распаковать сервер Apache Tomcat 7.
* Установить СУБД MySQL.
* Настроить СУБД: выполнить скрипт создания БД; создать нового пользователя и дать полный доступ к созданной базе данных.
* Скачать проект (https://github.com/MashaFomina/web-portal-of-state-services).
* Прописать параметры подключения к БД (название БД, имя пользователя и пароль) в файле **/src/main/resources/application.properties**
* Перейти в корневую папку проекта. Собрать war файл при помощи команды **mvn compilewar:war**.
* Удалить предыдущие версии проекта из папки веб приложений сервера Tomcat с помощью команды **rm -rf \$TOMCAT\\_WEBAPPS/web-portal-0.1.0-SNAPSHOT**.
* Скопировать новый war файл в папку веб-приложений сервера: **cp web-portal-0.1.0-SNAPSHOT \$TOMCAT\\_WEBAPPS**.
* Перезапустить сервер Tomcat.

1. **Инструкция пользователя**

При переходе на главную страницу сайта будет выведена форма для входа на портал. Если у пользователя нет аккаунта, он может зарегистрироваться. После успешного входа на портал, пользователь будет перенаправлен на свою главную страницу. На главной странице в верхней части страницы присутствует кнопка для выхода из системы.

**Гражданин**

В верхней части страницы находится панель навигации и кнопка для выхода из системы. Гражданину доступны страницы профиля, учреждений, информации о детях, с талонами, с заявками в образовательные учреждения.

На странице профиля гражданин может увидеть важные уведомления, а также поменять информацию профиля.

На странице учреждений гражданин может найти медицинское или образовательное учреждение по городу и району. Выбрав учреждение, он может перейти на страницу конкретного учреждения, оставить отзыв, выбрать талон, посмотреть информацию о докторах, подать заявку в образовательное учреждение.

На странице информации о детях гражданин может добавить и удалить информацию о ребёнке, просмотреть информацию о добавленных детях.

На странице с талонами гражданин может увидеть список своих талонов, а также может отказаться от талонов, если время посещения, ещё не наступило.

На странице с заявками в образовательные учреждения гражданин может увидеть список заявок, подтвердить и удалить заявку.

**Представитель образовательного учреждения**

На вкладке учреждения доступна функция редактирования информации об учреждении, о свободных местах, а также функции добавления отзыва и просмотра всех отзывов.

На странице с заявками отображается список заявок, представитель может подтвердить заявку, отказать в поступлении, подтвердить поступление ребёнка в образовательное учреждение, назначить время личной встречи.

**Представитель медицинского учреждения**

На вкладке учреждения доступна функция редактирования информации об учреждении, а также функции добавления отзыва и просмотра всех отзывов.

На странице докторов доступны функции просмотра списка докторов, добавления нового доктора, удаление информации о докторе, добавления талонов.

На странице талонов доступны функции удаления талонов, если время посещения ещё не наступило, а также удаление талонов для конкретного доктора. Представитель может подтвердить посещение медицинского учреждения, если время талона уже прошло.

**Вывод**

Был разработан портал государственных услуг. В рамках работы были изучены принципы работы с ORM Hibernate, с технологиями Spring Data и Spring MVC. ORM позволяет сделать многие вещи, связанные с хранением данных, прозрачными для программиста, а Spring Data дает возможность автоматической генерации всех необходимых методов обращения к слою хранения. Для хранения данных использовалась MySQL. Для слоя представления был использован паттерн Model-View-Controller.

Задачи, поставленные в рамках курсового проекта, были выполнены. Для дальнейшего развития системы:

* Необходимо улучшить внешний вид страниц.
* Создать страницы для доктора и администратора. Администратор должен отвечать за добавление новых учреждений в систему, а также представителей учреждений.