Московский Авиационный институт

(Национальный исследовательский университет)



Институт №8

«Информационные технологии и прикладная математика»

Кафедра 813

«Компьютерная математика»

Курсовой проект по дисциплине «Базы данных»

Тема: «База данных Волонтерского центра»

Студент: Ризоев Б. М.

Группа: M80-310Б-18

Преподаватель: Романенков А. М.

Оценка:

Дата:

Москва 2020

Оглавление

[Введение 4](#_Toc62342294)

[Структура базы данных 5](#_Toc62342295)

[Таблица Volunteers 5](#_Toc62342296)

[Таблица Organizations 6](#_Toc62342297)

[Таблица Events 6](#_Toc62342298)

[Промежуточная таблица Events of volunteers 6](#_Toc62342299)

[Промежуточная таблица Events requests of volunteers 6](#_Toc62342300)

[Таблица Users 7](#_Toc62342301)

[Таблица Roles 7](#_Toc62342302)

[Представление 7](#_Toc62342303)

[Хранимые процедуры 8](#_Toc62342304)

[Триггер 8](#_Toc62342305)

[Сервер 10](#_Toc62342306)

[Взаимодействие с СУБД 10](#_Toc62342307)

[Entity 10](#_Toc62342308)

[Repository 12](#_Toc62342309)

[Аутентификация 12](#_Toc62342310)

[REST API 14](#_Toc62342311)

[WPF клиент 17](#_Toc62342312)

[Авторизация/Регистрация 18](#_Toc62342313)

[Страница администратора 21](#_Toc62342314)

[Волонтеры 22](#_Toc62342315)

[Мероприятия 25](#_Toc62342316)

[Организаторы 25](#_Toc62342317)

[Пользователи 26](#_Toc62342318)

[Страница пользователя 29](#_Toc62342319)

[Новые мероприятия 29](#_Toc62342320)

[Мероприятия, на которые была подана заявка 30](#_Toc62342321)

[Информация о пользователе 31](#_Toc62342322)

[Вывод 33](#_Toc62342323)

[Список литературы. 34](#_Toc62342324)

[Код сервера 35](#_Toc62342325)

[Код клиента 61](#_Toc62342326)

# Введение

Цель курсового проекта – создание пользовательское приложение с возможностью администрирования. То есть интерфейс для волонтера и руководителя Волонтерского центра.

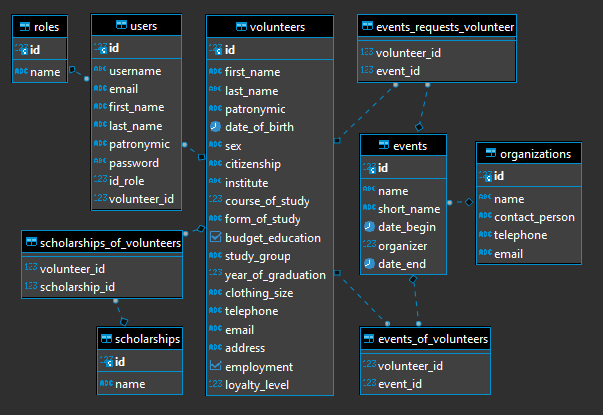
Основной особенностью проекта является интуитивно понятный интерфейс, который при этом еще и удобно связывает различные таблицы между собой.

Для работы с базой данных используется PostgreSQL – свободная объектно-реляционная система управления базами данных.

Само приложение написано на WPF – система для построения клиентских приложений Windows, графическая подсистема в составе .NET Framework, использующая язык XAML.

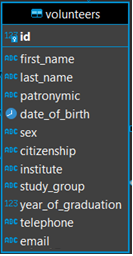
А связывает их сервер, построенный с помощью Spring Boot, который упрощает использование Spring Framework – фреймворк с открытым исходным кодом для Java-платформы.

# Структура базы данных



На первый взгляд структура кажется запутанной, ибо таблиц много и они взаимосвязаны, но на деле все просто. Разберем каждую отдельно.

## Таблица Volunteers



Разберем самую главную таблицу – таблицу волонтеров.

В ней есть поля: ID – уникальный идентификатор и первичный ключ как и в каждой последующей таблице, ФИО, дата рождения, пол, гражданство, название факультета/института, номер учебной группы, год окончания обучения, телефон и адрес электронной почты.

Такие данные о волонтере являются самыми распростараненными в нашем университете. Именно эта таблица будет связана со всеми основными таблицами.

## Таблица Organizations

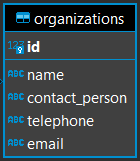
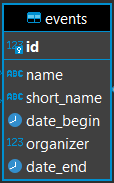
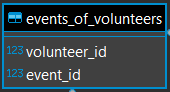


Таблица состоит из ID организации, ее имени, контактного лица, телефона и электронной почты.

## Таблица Events

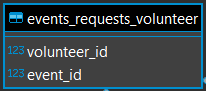
 Таблица мероприятий состоит из ID, названия, короткого названия, дат начала и конца и организатора. Поле организатор(ID) явяется внешним ключом для таблицы организаторов. Реализована связь «один к одному».

## Промежуточная таблица Events of volunteers

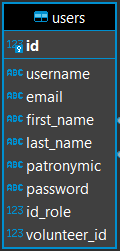


Эта таблица связывает волонтера(ID) с мероприятием(ID) с помощью внешних ключей для соответствующих таблиц. Здесь реализована связь «многие ко многим».

## Промежуточная таблица Events requests of volunteers

 У волонтеров будет возможность подавать заявки на мероприятия. Все заявки будут находиться в этой таблице. Таблица аналогична предыдущей, из нее данные будут переноситься в предыдущую.

## Таблица Users



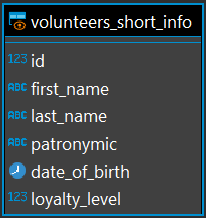
Еще одной важной таблицей является таблица пользователей.

Она состоит из полей: ID, уникальное имя пользователя, электронная почта, ФИО, зашифрованный по технологии bcrypt, пароль, внешний ключ (ID) на таблицу Roles (роль пользователя) и внешний ключ (ID) на таблицу волонтеров. К каждому пользователю прикреплен волонтер (об этом подробнее – в части про клиентского приложения).

## Таблица Roles

 Таблица ролей с ID и названием роли определяет, какая роль будет у пользователя. Всего ролей две: ROLE\_ADMIN и ROLE\_USER для администратора и пользователя соответственно. Взависимости от роли будет доступна та или иная информация. Также это влияет на само клиентское приложения: у пользователя будут одни функции, а у администратора – совершенно другие.

## Представление

 В ходе работы приложения очень часто происходит обращение к таблице волонтеров, причем не все поля нужны, поэтому было решено добавить представление «Короткая информация волонтеров», в котором есть ID, ФИО, дата рождения и уровень лояльности.

## Хранимые процедуры

Таблица волонтеров содержит большое количество полей. Некоторые из них являются выбором из нескольких вариантов, поэтому для них существуют таблицы с вариантами и промежуточные таблицы с типом связи «многие ко многим».

Вставка в эти таблицы занимало бы несколько отдельных запросов, поэтому была создана данная хранимая процедура, которая принимает в качестве аргументов: ID волонтера, название языка, ID медиа-навыка, ID степендии, ID направлений волонтерства. Сначала передаваемый язык ищется в таблице языков, в случае отсутствия вставляется в нее, а после в промежуточную таблицу вставляется ID этого языка и ID волонтера. С остальными полями происходит то же самое без вставления в таблицы, ибо выбирает пользователь из ограниченного набора вариантов.

**create** **or** **replace** **function** add\_into\_vols\_references(id\_vol int, lang varchar(80), id\_skill int, id\_scholarship int, id\_activity int) **returns** boolean **as**

$$

**declare** id\_language int;

begin

**if** **not** **exists** (**select** 1 **from** languages **where** lower(**name**) = lower(lang))

**then** **insert** **into** languages (**name**) **values** (lang) **returning** id **into** id\_language;

**else** **select** id **into** id\_language **from** languages **where** lower(**name**) = lower(lang);

end if;

**insert** **into** langs\_of\_volunteers (volunteer\_id, language\_id) **values** (id\_vol, id\_language);

**insert** **into** media\_skills\_of\_volunteers (volunteer\_id, media\_skill\_id) **values** (id\_vol, id\_skill);

**insert** **into** scholarships\_of\_volunteers (volunteer\_id, scholarship\_id) **values** (id\_vol, id\_scholarship);

**insert** **into** activities\_of\_volunteers (volunteer\_id, activity\_id) **values** (id\_vol, id\_activity);

return true;

end

$$ **language** 'plpgsql';

Другие хранимые процедура заменяются ORM в Spring Boot.

## Триггер

Пользователь может захотеть удалить свои данные. Администратор может удалить пользователя или себя. Во втором случае есть риск возникновения ситуации, когда не останется администраторов в базе. В данном случае придется напрямую ввести нового админа в базу, что нежелательно. Поэтому был создан данный тригер – при удалении пользователя, если пользователь администратор, проверяется, не является ли он единственным. В этом случае он не сможет удалить себя.

create or replace function delete\_user\_func() returns trigger as

$$

begin

if exists (select 1 from users where id = old.id and id\_role = 1)

then if (select count(1) from users where id\_role = 1) < 2

then return null;

end if;

end if;

return old;

end

$$ language 'plpgsql';

create trigger delete\_user before delete on users

for row execute procedure delete\_user\_func()

# Сервер

Как уже было выше сказано, сервер реализован на Spring Boot на языке Java.

## Взаимодействие с СУБД

Все взаимодействие происходит с помощью ORM.

ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение, или преобразование) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

В Spring Boot используется реализация ORM – JPA – технология требуется для работы с базами данных и обеспечивает объектно-реляционное отображение Java объектов, предоставляет API (Hibernate в нашем случае) для управления, сохранения и получения сущностей.

## Entity

Entity (cущности/модели) – заворачивание таблиц в виде Java-классов. Поля повторяют поля таблиц на стороне БД. Рассмотрим примеры некоторых сущностей.

@Entity

@Table(name = "users")

@Data

public class User

{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "username")

private String username;

@Column(name = "email")

private String email;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "patronymic")

private String patronymic;

@Column(name = "password")

private String password;

@ManyToOne

@JoinColumn(name = "id\_role")

private Role role;

@OneToOne

@JoinColumn(name = "volunteer\_id")

private Volunteer volunteer;

}

Как видно – поля копируют соответствующие столбцы таблицы пользователей. Виды связей указывается напрямую.

Таблица ролей:

@Entity

@Table(name = "roles")

@Data

public class Role

{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

}

Рассмотрим более интересную сущность:

@Entity

@Table(name = "events")

@Data

public class Event

{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

@Column(name = "short\_name")

private String shortName;

@Column(name = "date\_begin")

private Timestamp dateBegin;

@Column(name = "date\_end")

private Timestamp dateEnd;

@ManyToOne

@JoinColumn(name = "organizer")

private Organization org;

@ManyToMany

@JoinTable(

name = "events\_of\_volunteers",

joinColumns = {@JoinColumn(name = "event\_id")},

inverseJoinColumns = {@JoinColumn(name = "volunteer\_id")})

private List<VolunteerInfo> volunteers;

@ManyToMany

@JoinTable(

name = "events\_requests\_volunteer",

joinColumns = {@JoinColumn(name = "event\_id")},

inverseJoinColumns = {@JoinColumn(name = "volunteer\_id")})

private List<VolunteerInfo> requestedVols;

}

Так как это Java класс, то мы можем использовать списки: в данном случае у каждого мероприятия есть список волонтеров, которые участвуют, и список подавших заявку волонтеров. Причем промежуточные таблицы, к полям которых мы обращаемся, не завернуты в Java классы – ORM их самостоятельно достает из СУБД.

## Repository

Модели готовы – осталось доставать объекты из базы. И тут на помощь приходят репозитории. Для каждой сущности создаются отдельные репозитории с методами без тела, а создает эти методы во время компиляции Spring Data. Рассмотрим пример.

public interface UserRepository extends JpaRepository<User, Long>

{

User findByUsername(String name);

void deleteByUsername(String login);

}

Определены методы нахождения пользователя по username и удаление. Расширяет наш репозиторий шаблонизированный JpaRepository с нашей моделью User.

А так выглядит репозиторий для взаимодейтвия с таблицей волонтеров:

public interface VolunteerRepository extends JpaRepository<Volunteer, Long>

{

List<Volunteer> findByFirstName(String volunteerFirstName);

List<Volunteer> findByLastName(String volunteerLastName);

List<Volunteer> findBySex(String sex);

Optional<Volunteer> findByFirstNameAndLastNameAndPatronymicAndEmail(

String firstName, String lastName, String Patronymic, String email);

List<Volunteer> findByEvents(Event event);

}

Определены методы для нахождения по фамилии, имени, полу, ФИО и электронной почты (именно по связи таких полей мы однозначно опеределяем волонтера) и мероприятию. Возвращают эти методы списки, которые будут отправяться в качестве ответов на запросы.

## Аутентификация

Важной частью приложения является различный доступ к данным. Именно поэтому у каждого пользователя своя роль. Данная база данных имеет две роли – администратор и пользователь. Первый может удалять, добавлять и изменять волонтеров, мероприятия и организации. Сервер предоставляет доступ к данным взависимости от роли, при этом без авторизации доступ невозможен.

Для аутентификации пользователей используется JWT (JSON Web Token) – это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Токен создается сервером в момент обращения и подтверждения пользователя и передается клиенту, который в дальнейшем обращается к серверу с ним.

JWT состоит из трех частей: заголовок header, полезные данные payload и подпись signature. В header содержится информация о том, как должна вычисляться JWT, в payload полезные данные, которые хранятся внутри JWT, signature вычисляется на основе данных с header и payload, кодирует их, разделяя через точку, а потом на хэширует с ключевым словом.

public String createToken(String login, Role role)

{

Claims claims = Jwts.claims().setSubject(login);

claims.put("role", getRoleNames(role));

Date now = new Date();

Date validaty = new Date(now.getTime() + validityInMilliseconds);

return Jwts.builder()

.setClaims(claims)

.setIssuedAt(now)

.setExpiration(validaty)

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

**validityInMilliseconds** указывает время жизни токена. А в **claims** – заявку добавляем желаемые поля (в данном случае это имя пользователя и роль).

После получения запроса с токеном, надо проверить ее валидность. Происходит это вот так:

public boolean validateToken(String token) throws JwtAuthenticationException

{

try

{

Jws<Claims> claims = Jwts.parser().setSigningKey(secret).parseClaimsJws(token);

if (claims.getBody().getExpiration().before(new Date()))

return false;

return true;

} catch (JwtException | IllegalArgumentException e)

{

throw new JwtAuthenticationException("JWT token is expired or invalid");

}

}

Выше было сказано, что разные пользователи будут иметь различный доступ к данным. Для этого существует следующий класс:

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter

{

private final JwtTokenProvider jwtTokenProvider;

private static final String ADMIN\_ENDPOINT = "/api/admin/\*\*";

private static final String USER\_ENDPOINT = "/api/user/\*\*";

private static final String AUTH\_ENDPOINT = "/api/auth";

private static final String REG\_ENDPOINT = "/api/register";

@Override

protected void configure(HttpSecurity http) throws Exception

{

http

.httpBasic().disable()

.csrf().disable()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)

.and()

.authorizeRequests()

.antMatchers(USER\_ENDPOINT).hasRole("USER")

.antMatchers(ADMIN\_ENDPOINT).hasRole("ADMIN")

.antMatchers(AUTH\_ENDPOINT, REG\_ENDPOINT).permitAll()

.anyRequest().authenticated()

.and()

.apply(new JwtConfigurer(jwtTokenProvider));

}

}

Для пользователей взависимости от роли есть разные «конечные точки», к которым они могут достучаться, причем все запросы авторизованы. А к страницам авторизации и регистрации имеют доступ все.

## REST API

Сервер и клиент в свою очередь общаются с помощью технологии REST.

REST (от англ. Representational State Transfer – «передача состояния представления») – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

Вызов удалённой процедуры представляет собой обычный HTTP-запрос (GET или POST; такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

В проете существует 4 класса контроллеров с методами для работы с URL. Например, в контроллере аутентификации есть методы:

@PostMapping("register")

public Object register(@RequestBody RegistrationRequestDto registrationRequestDto)

{

User user = new User();

user.setUsername(registrationRequestDto.getUsername());

user.setPassword(registrationRequestDto.getPassword());

user.setEmail(registrationRequestDto.getEmail());

user.setLastName(registrationRequestDto.getLastName());

user.setFirstName(registrationRequestDto.getFirstName());

user.setPatronymic(registrationRequestDto.getPatronymic());

try

{

userService.register(user);

return "OK";

} catch (Exception e)

{

log.info("Register - Username: {} is busy", user.getUsername());

return "NO";

}

}

Принимает в качестве параметра класс с полями пользователя и пробует зарегистрировать пользователя.

@PostMapping("auth")

public ResponseEntity login(@RequestBody AuthenticationRequestDto requestDto)

{

try

{

String username = requestDto.getUsername();

authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username, requestDto.getPassword()));

User user = userService.findByUsername(username);

if (user == null)

throw new UsernameNotFoundException("User with username: " + username + " not found");

String token = jwtTokenProvider.createToken(username, user.getRole());

Map<Object, Object> response = new HashMap<>();

response.put("id", user.getId().toString());

response.put("role", user.getRole().getName());

response.put("username", username);

response.put("token", token);

return ResponseEntity.ok(response);

} catch (AuthenticationException e)

{

throw new BadCredentialsException("Invalid username or password");

}

}

Принимает в качестве параметра класс с именем пользователя и паролем. Затем сравнивает эти данные с теми, которые на базе. После чего создает ассоциативный массив с ID, ролем, именем пользователя и токеном.

Контроллер пользователя имеет следующие методы.

Получение данных пользователя:

@GetMapping("profile")

public ResponseEntity<User> getUser(HttpServletRequest request)

{

Long userId = Long.parseLong(request.getParameter("id"));

log.info("Get user (User request) by UserId = {}", userId);

Optional<User> user = userService.findById(userId);

return user.isPresent()

? new ResponseEntity<>(user.get(), HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

Получение списка мероприятий пользователя:

@GetMapping("events/my\_events")

public ResponseEntity getUserEvents(HttpServletRequest request)

{

Long userId = Long.parseLong(request.getParameter("id"));

log.info("Get all events (User request) by UserId = {}", userId);

Optional<User> user = userService.findById(userId);

if (user.isEmpty())

return ResponseEntity.badRequest().body("User hasn't found");

else if (user.get().getVolunteer() == null)

return ResponseEntity.badRequest().body("User's volunteer profile hasn't found");

Map<Object, Object> response = new HashMap<>();

response.put("events", user.get().getVolunteer().getEvents());

response.put("requestedEvents", user.get().getVolunteer().getRequestedEvents());

return ResponseEntity.ok(response);

}

Данный метод получает в качестве параметра ID, по нему ищет пользователя и, если к пользователю привязан волонтер, возвращает список мероприятий.

Подача заявки на мероприятие:

@PostMapping("events/request")

public Object requestEvent(HttpServletRequest request)

{

Long userId = Long.parseLong(request.getParameter("user\_id"));

Long eventId = Long.parseLong(request.getParameter("event\_id"));

log.info("Request event(id) = {} (User request) by user, Id = {}", eventId, userId);

Optional<User> user = userService.findById(userId);

Optional<Event> event = infoService.findEventById(eventId);

if (user.isEmpty() || event.isEmpty() || user.get().getVolunteer() == null)

return "User has no volunteer or there is no event";

user.get().getVolunteer().getRequestedEvents().add(event.get());

volunteerService.save(user.get().getVolunteer());

return "OK";

}

Получает ID пользователя и ID мероприятия, затем ищет пользователя и обновляет список его мероприятий.

Как видно, у каждого запроса есть конечная URL, который однозначно определен для этого проекта.

# WPF клиент

Клиент написан на WPF – система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык XAML – с использованием паттерна MVVM(Model-View-ViewModel) – шаблон проектирования архитектуры приложения, который позволяет отделить логику приложения от визуальной части (представления) таким образом, что модификация каждого компонента может осуществляться независимо.

* Модель (Model) предоставляет/изменяет реальные данные, содержит логику изменения этих самых данных.
* Представление (View) определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление – это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.
* Модель представления (View Model) связывает модель и представление через механизм привязки данных, содержит логику по получению данных из модели, которые потом передаются в представление. Также определяет логику по обновлению данных в модели.

Приложение написано с использованием Material Design – cтиль графического дизайна интерфейсов программного обеспечения и приложений, разработанный компанией Google.

В проекте есть важный класс – **NavigationManager**, который осуществляет переход по страницам и дает возможность вернуться на предыдущую страницу.

public static bool Navigate<TView>(BaseViewModel viewModel)

where TView : FrameworkElement, new()

{

try

{

NavigationService?.Navigate(new TView { DataContext = viewModel });

return true;

}

catch

{

return false;

}

}

Данный метод навигирует приложение на страницу, принимая в качестве шаблонизированного параметра модель, а в аргументе – модель представления. А следующий возвращает предыдущую:

public static bool GoBack()

{

try

{

if (!CanGoBack)

{

return false;

}

NavigationService?.GoBack();

return true;

}

catch

{

return false;

}

}

Также можно выделить команды (Command) – паттерн, который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. Этот объект запроса на действие и называется командой. В проекте создан класс:

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return \_canExecute == null || \_canExecute(parameter);

}

public void Execute(object parameter)

{

\_execute(parameter);

}

public event EventHandler CanExecuteChanged

{

add

{

CommandManager.RequerySuggested += value;

}

remove

{

CommandManager.RequerySuggested -= value;

}

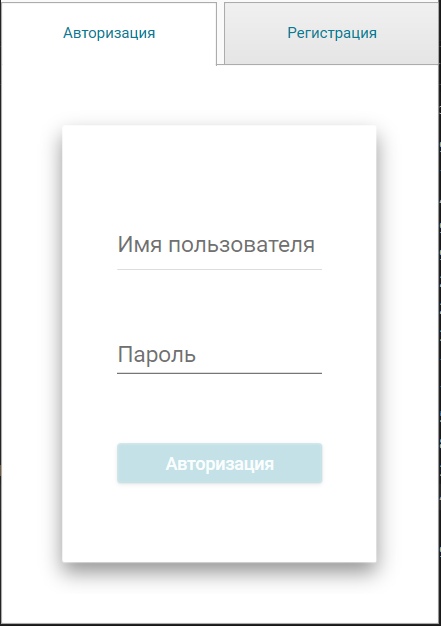
}

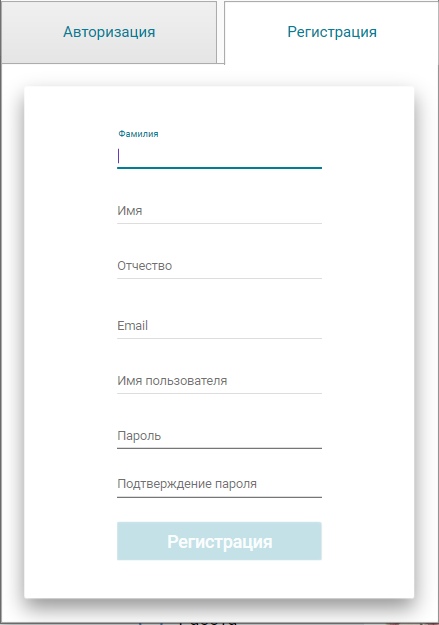
В качестве параметров конструктора принимает делегат – выполняемый метод и делегат – проверка возможности выполнения команды.

## Авторизация/Регистрация

Точкой входа в программу является страница авторизации и регистрации.

Представление выглядит следующим образом (кнопки активируются, если все поля заполнены):





XAML код:

<Grid>

<TabControl TabStripPlacement="Top" HorizontalAlignment="Center" VerticalAlignment="Center">

<TabItem Width="173" Height="50" HorizontalAlignment="Left" Margin="-2,0,0,-0.4">

<TabItem.Header>

<TextBlock Text="Авторизация" Foreground="{DynamicResource buttonBackground}"/>

</TabItem.Header>

<Grid>

<materialDesign:Card

materialDesign:ShadowAssist.ShadowDepth="Depth4"

Padding="32" Margin="46,46,46.46,46" />

<Grid>

<TextBox

materialDesign:HintAssist.Hint="Имя пользователя"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

FontSize="18"

VerticalAlignment="Center" Margin="90,109,90,280.2" Height="54"

Text="{Binding Username, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" CaretBrush="Black"/>

<PasswordBox

x:Name="FloatingPasswordBox"

FontSize="18"

materialDesign:HintAssist.Hint="Пароль"

Style="{DynamicResource MaterialDesignFloatingHintPasswordBox}" Margin="90,192,90,197.2"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

ph:PasswordHelper.Attach ="True"

ph:PasswordHelper.Password="{Binding Password, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<Button

Style="{DynamicResource MaterialDesignRaisedDarkButton}"

Background="{DynamicResource buttonBackground}"

Content="Авторизация" Margin="90,300,90,109"

Command="{Binding AuthorizeCommand, UpdateSourceTrigger=PropertyChanged}" BorderBrush="{DynamicResource buttonBackground}"/>

</Grid>

</Grid>

</TabItem>

Как и во всех представлениях все элементы хранятся в различных контейнерах. Иерархия страницы: контейнер – контейнер для вкладок – вкладка – контейнер – два текстовых поля и кнопка. Другая вклдака построена аналогично. Сами поля для ввода связаны со свойствами View-Model этого представления, а обработкой нажатия кнопки занимается соответствующая команда.

Во View-Model этой страницы есть свойства:

public string Username

{

private get =>

user.username;

set

{

user.username = value;

OnPropertyChanged(nameof(Username));

}

}

public string Email

{

private get =>

user.email;

set

{

user.email = value;

OnPropertyChanged(nameof(Email));

}

}

При изменении поля вызывается метод **OnPropertyChanged**, который обновляет данные в модели.

Также команды:

public ICommand AuthorizeCommand =>

\_authorizeCommand ?? (\_authorizeCommand = new RelayCommand(async \_ => await AuthorizeAsync(),

\_ => CanAuthorize()));

private async Task AuthorizeAsync()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/auth");

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

string json = JsonSerializer.Serialize(new

{

username = user.username,

password = user.password

});

await streamWriter.WriteAsync(json);

}

var httpResponse = (HttpWebResponse) await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

Dictionary<string, string> jsonToken = JsonSerializer.Deserialize<Dictionary<string, string>>(result);

if (jsonToken["role"] == "ROLE\_ADMIN")

NavigationManager.Navigate<AdminMainPage>(new AdminPageViewModel(jsonToken["token"]));

else

NavigationManager.Navigate<UserMainPage>(new UserMainPageViewModel(jsonToken["token"], long.Parse(jsonToken["id"])));

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private bool CanAuthorize()

{

return Username.Length > 0 && Password.Length > 0;

}

В конце приведен метод, который определяет, можно ли выполнить команду – для этого поле «имя пользователя» и «пароль» должны быть заполнены. Метод **AuthorizeAsync** является асинхронным – то есть во время обращения в сервер приложение не зависнет. В самом методе создается **HttpWebRequest**, которому указывается адрес запроса, тип содержания, метод обращения (POST). Затем в поток записи пишутся имя пользователя и пароль, а из ответа достается поток чтения и парсится JSON объект. В зависимости от роли происходит переход на страницу администратора или пользователя.

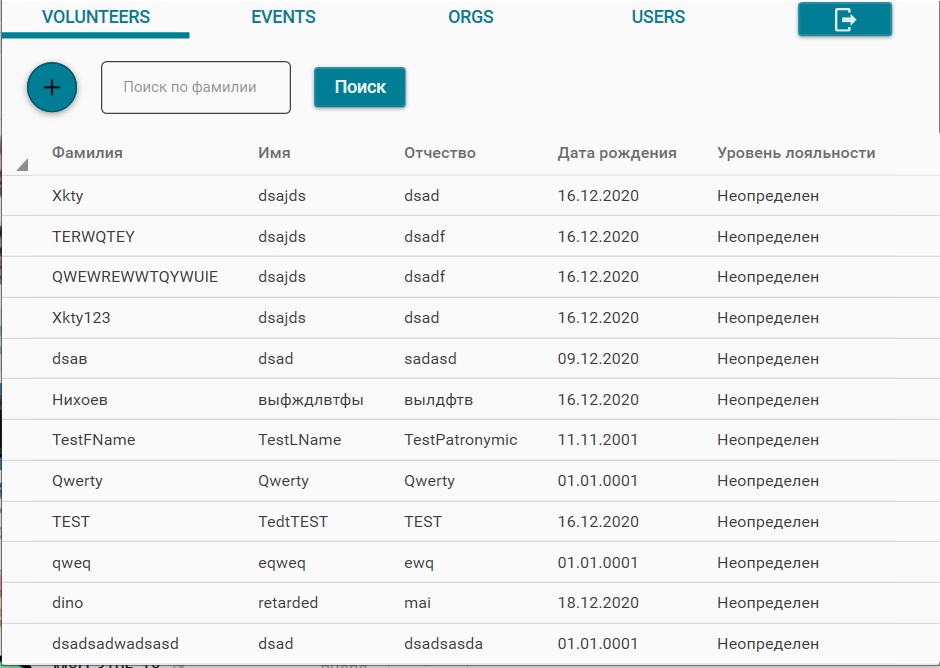
Модель ничем не отличается от раннее созданных моделей пользователя.

## Страница администратора

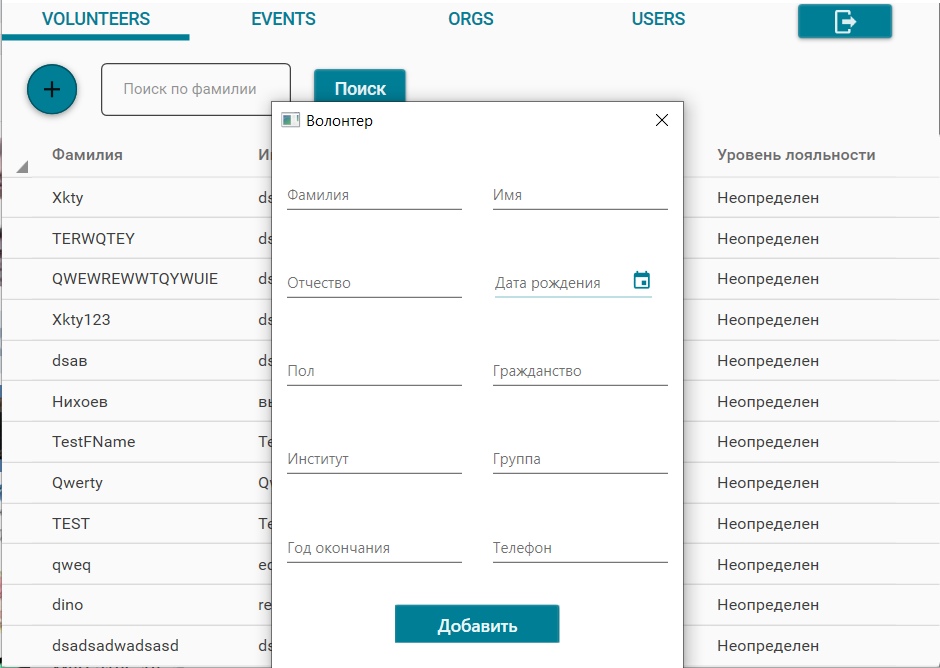
В данной странице есть 4 вкладки: волонтеры, мероприятия, организаторы и пользователи. Рассмотрим каждую отдельно.

### Волонтеры

Так выглядит представление:



Виден список волонтеров, поле с поиском по фамилии и кнопка добавление волонтера. Причем маленькая форма выскакивает также при двухкратном нажатии на любого волонтера для изменения данных:



Заглянем в поисках интересных мест в XAML код:

<DataGrid Grid.Row="1"

x:Name="volsGrid"

Visibility="Hidden"

ItemsSource="{Binding ItemsSourceVols}"

SelectedItem="{Binding SelectedVol}"

CanUserAddRows="False"

AutoGenerateColumns="False"

HeadersVisibility="All"

IsReadOnly="True">

<DataGrid.InputBindings>

<MouseBinding

MouseAction="LeftDoubleClick"

Command="{Binding SelectVolCommand}"

CommandParameter="{Binding SelectedVol}"/>

</DataGrid.InputBindings>

<DataGrid.Columns>

<DataGridTextColumn

Binding="{Binding firstName}"

Header="Фамилия"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}">

</DataGridTextColumn>

<DataGridTextColumn

Binding="{Binding lastName}"

Header="Имя"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding patronymic}"

Header="Отчество"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding DateOfBirth}"

Header="Дата рождения"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding loyalty}"

Header="Уровень лояльности"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

</DataGrid.Columns>

</DataGrid>

Здесь используется элемент управления, отображающий данные в настраиваемой сетке. Такая сетка позволяет настроить поля взависимости от наших нужд. Данные берутся из коллекции, которая есть во View-Model. Привязкой взаимодейсвий занимается **InputBindings**, который при нажатии два раза на любую строку вызывает команду с параметром – выбранная строка. Такой же способ используется при ввода в поле поиска слова и нажатии на кнопку поиска:

<TextBox

x:Name="searchVol"

Visibility="Collapsed"

Style="{DynamicResource MaterialDesignOutlinedTextFieldTextBox}"

VerticalAlignment="Top"

Height="40"

TextWrapping="NoWrap"

materialDesign1:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign1:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalScrollBarVisibility="Disabled"

materialDesign1:HintAssist.Hint="Поиск по фамилии"

Width="150" Margin="20,14,0,14"/>

<Button

x:Name="findVolButton"

Visibility="Collapsed"

Command="{Binding FindByFirstNameVolCommand}"

CommandParameter="{Binding Text, ElementName=searchVol}"

Content="Поиск" BorderBrush="{DynamicResource buttonBackground}"

Background="{DynamicResource buttonBackground}"

Style="{DynamicResource MaterialDesignRaisedDarkButton}" Margin="20,14,0,14"/>

Коллекция во View-Model выглядит так:

public ObservableCollection<Volunteer> ItemsSourceVols

{

get =>

\_volunteers;

set

{

\_volunteers = value;

OnPropertyChanged(nameof(ItemsSourceVols));

}

}

Команда выбора волонтера:

public ICommand SelectVolCommand =>

\_selectVolCommand ?? (\_selectVolCommand = new RelayCommand(async obj => await SelectVolAsync(obj)));

private async Task SelectVolAsync(Object obj)

{

VolunteerWindow volWindow = new VolunteerWindow(obj as Volunteer);

bool? result = volWindow.ShowDialog();

if (result == true)

{

await AddVolunteerAsync(volWindow.Volunteer);

}

else if (result == false && volWindow.IsDeleting)

{

await SelectedVolunteerDeleteAsync((obj as Volunteer).id);

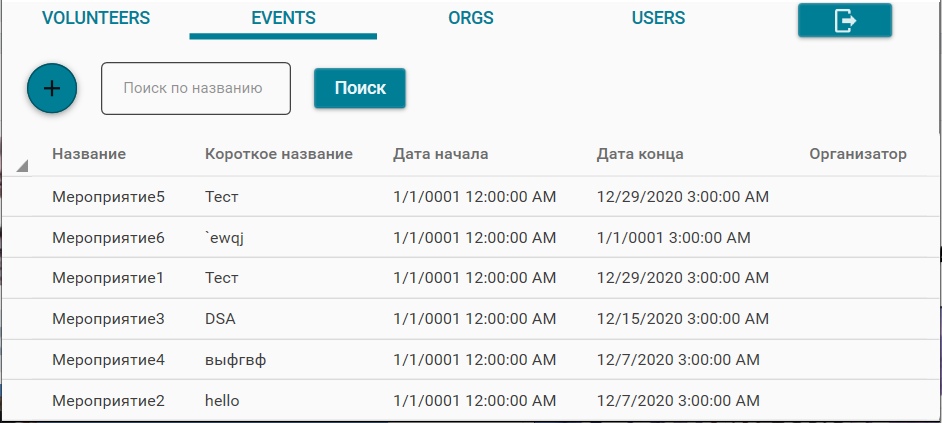
}

}

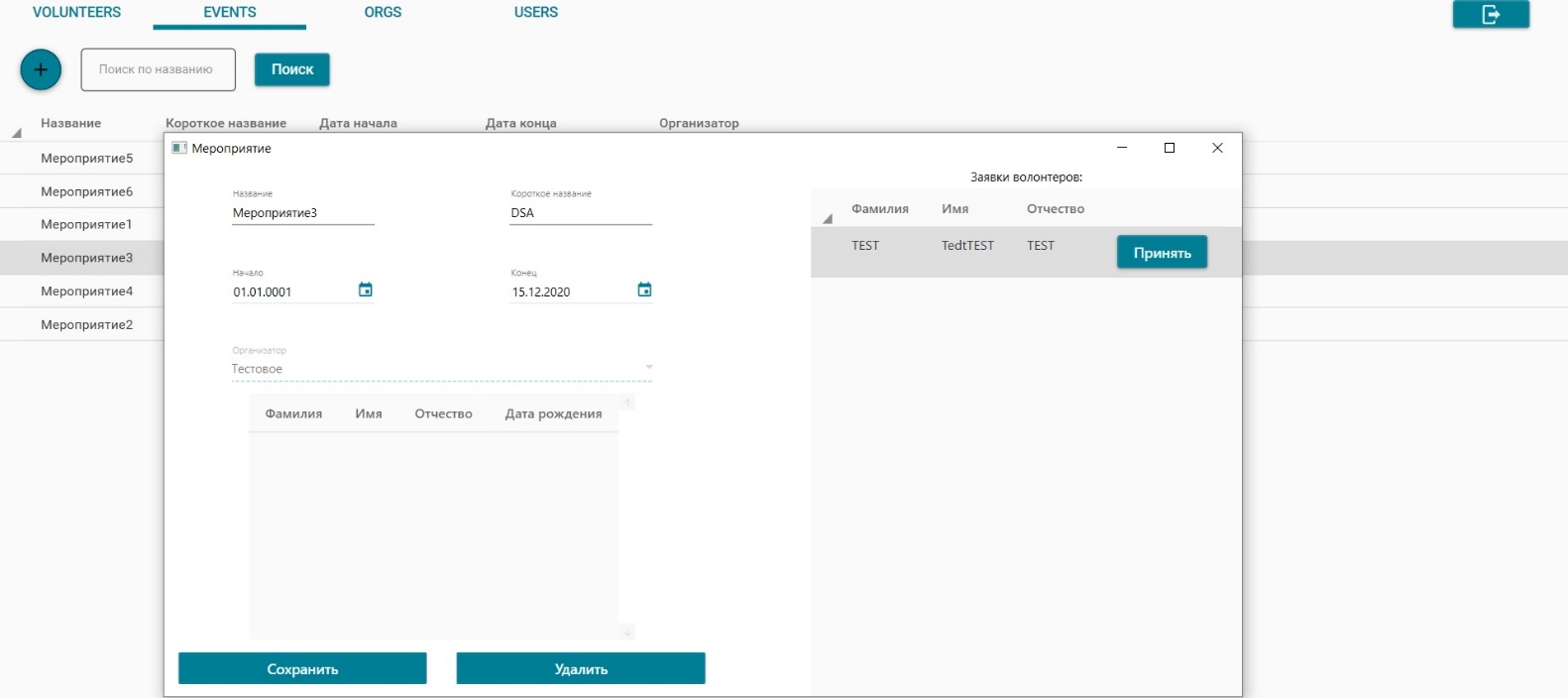
При выборе волонтера вызывается дочернее окно, при закрытии которого происходит сохранение/добавление или удаление волонтера.

### Мероприятия

Представление выглядит следующим образом:



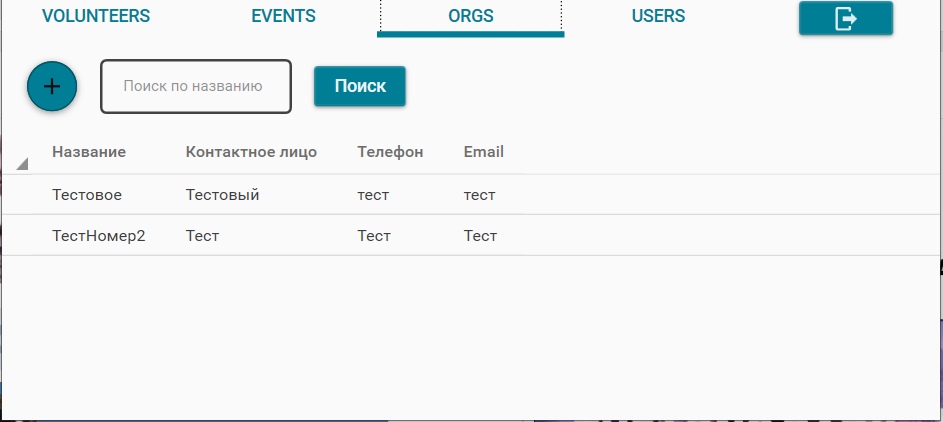
Тут также есть кнопка добавления и возможность изменения мероприятия. Причем мероприятие без организатора нельзя создать, а при выборе появляется выпадаюший список организаторов. Также показываются те волонтеры, которые участвуют в этом мероприятии, и те, которые подали заявку. Возле каждого такого волонтера есть кнопка «Принять», по нажатию которой волонтер добавляется в список принимающих участие:



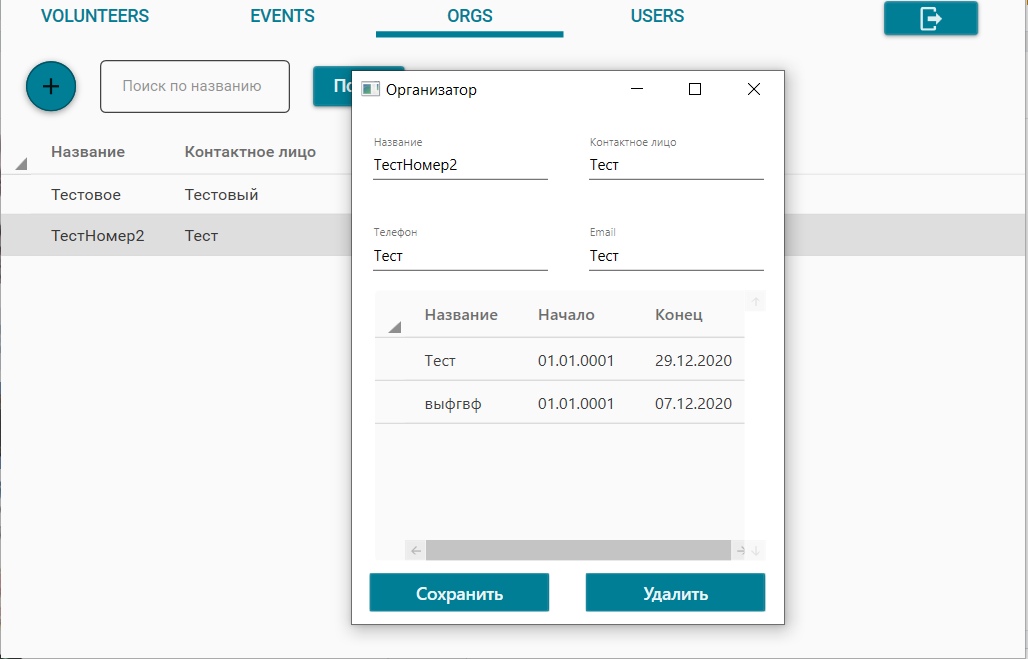
В модели представления ничего нового в отличии от предыдущей нет.

### Организаторы

Представление организатора также похоже:

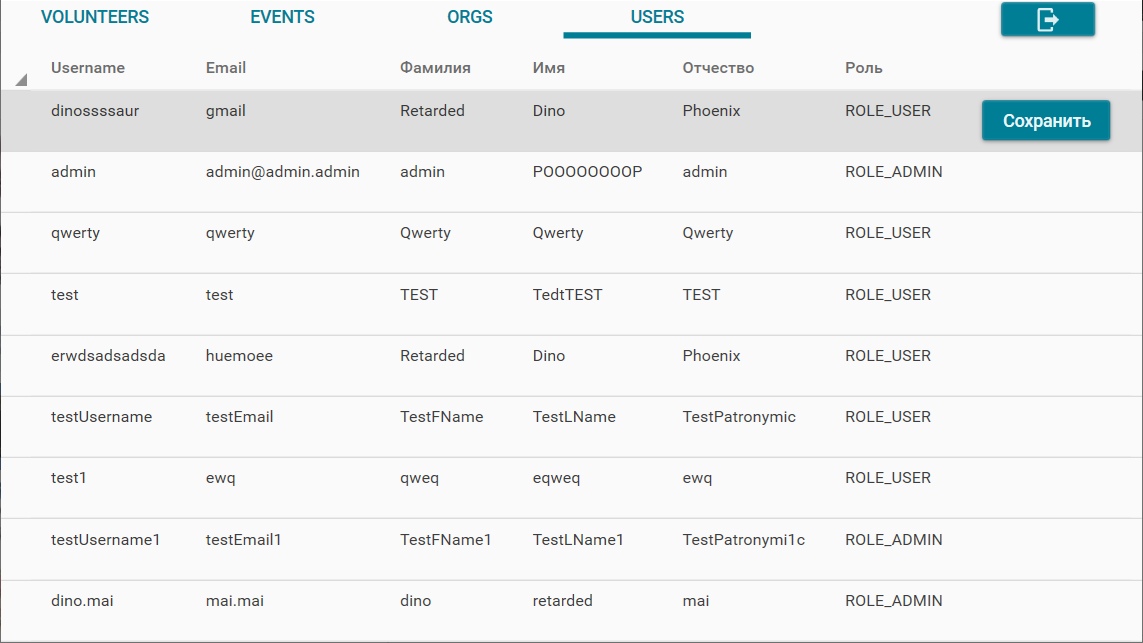


В окне показывают мероприятия, которые этот организатор проводит (-ил):



### Пользователи

Представление пользователей:



Представление ничем не отличается от других, кроме того, что тут во время наведения на роль какого-либо пользователя – появляется выпадающий список ролей. Во время выбора любой строки на ней появляется кнопка сохранить. Можно не бояться случая, что не останется ни одного администратора – это было сделано на стороне БД триггером.

В XAML коде этот момент выглядит вот так:

<DataGridTemplateColumn Header="Роль">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Role}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<ComboBox IsEditable="True"

SelectedItem="{Binding DataContext.SelectedRole,

RelativeSource={RelativeSource AncestorType=DataGrid}}"

IsSynchronizedWithCurrentItem="True">

<TextBlock>ROLE\_USER</TextBlock>

<TextBlock>ROLE\_ADMIN</TextBlock>

</ComboBox>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

Как уже говорилось выше – столбцы в DataGrid можно гибко настроить. Шаблоны позволяют задать другой элемент управления внутри этой ячейки. Ячейка отображает текстовое поле, но за ним скрыта редактируемая ячейка – выпадающий список. То есть во время нажатия на ячейку появляется список ролей. Текстовое поле достает данные из коллекции, а выбранный элемент является свойством во View-Model.

Во View-Model этот момент:

private bool \_lastUserModified;

private User \_selectedUser;

public User SelectedUser

{

get =>

\_selectedUser;

set

{

if (\_selectedUser != null && !\_lastUserModified)

{

\_selectedUser.role = new Role(\_oldUserRole.Equals("ROLE\_USER") ? 2 : 1, \_oldUserRole.ToString());

OnPropertiesChanged(nameof(SelectedUser));

}

\_lastUserModified = false;

\_selectedUser = value;

OnPropertyChanged(nameof(SelectedUser));

}

}

private string \_oldUserRole = string.Empty;

private string \_selectedRole = string.Empty;

public Object SelectedRole

{

get =>

\_selectedRole;

set

{

\_oldUserRole = SelectedUser.Role;

\_selectedRole = ((TextBlock)value).Text;

SelectedUser.role = new Role(\_selectedRole.Equals("ROLE\_USER") ? 2 : 1, \_selectedRole.ToString());

OnPropertyChanged(nameof(SelectedRole));

}

}

Во время нажатия на поле «Роль» происходит выделение целой строки – так узнается выбранный пользователь. При выборе другой роли этого пользователя меняется это еще и программно. Но при этом если не нажата кнопка «Сохранить», а произошел переход на другого пользователя – возвращается изначальная роль пользователю.

Еще одним интересным моментом является кнопка «Сохранить»:

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="saveBT" Visibility="Hidden"

Content="Сохранить" Background="{DynamicResource buttonBackground}"

BorderBrush="{DynamicResource buttonBackground}"

Command="{Binding DataContext.SelectUserSaveCommand,

RelativeSource={RelativeSource AncestorType=DataGrid}}"/>

<DataTemplate.Triggers>

<DataTrigger Binding="{Binding IsSelected,

RelativeSource={RelativeSource FindAncestor,

AncestorType={x:Type DataGridRow}}}" Value="True">

<Setter TargetName="saveBT" Property="Visibility" Value="Visible"/>

</DataTrigger>

</DataTemplate.Triggers>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

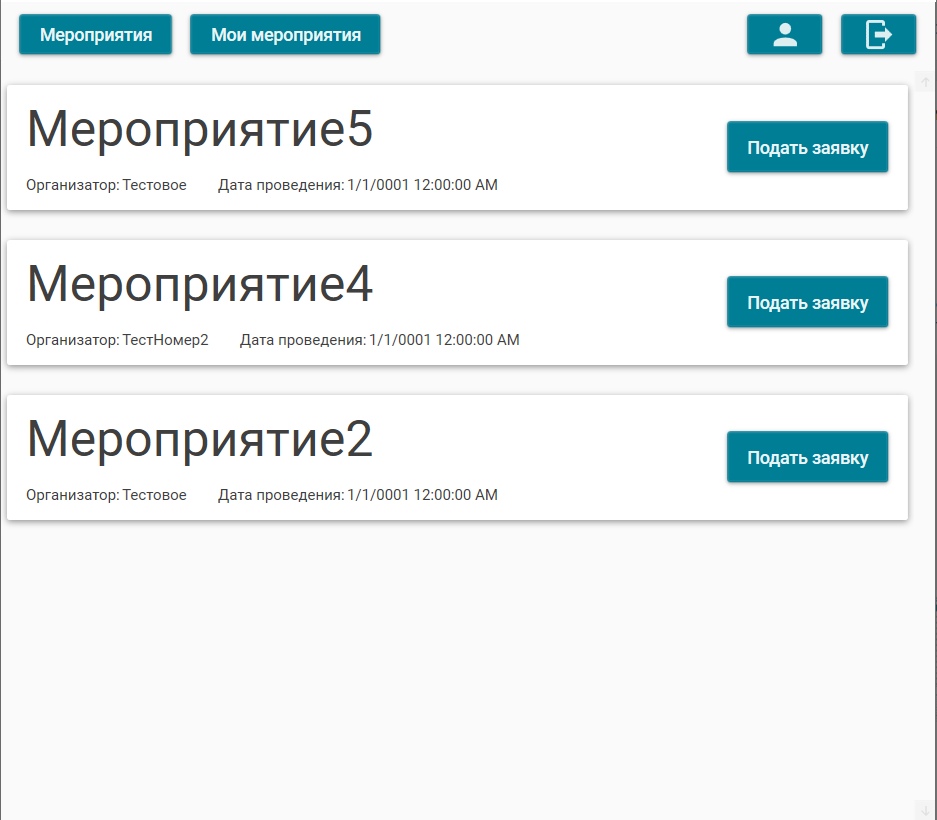
Изначально кнопка скрыта. В WPF есть триггеры элементов управления. Так, если навести на строку, то в ней кнопка становится видной.

## Страница пользователя

На этой странице есть 3 вкладки: новые мероприятия, мероприятия, на которые уже подана заявка, информация о пользователе. Рассмотрим их подробнее.

### Новые мероприятия

Так выглядит представление:



Есть новый момент в XAML коде:

<ListBox x:Name="listBox" ItemsSource="{Binding ItemsSourceEvents}">

<ListBox.ItemContainerStyle>

<Style TargetType="ListBoxItem">

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="IsSelected" Value="True"/>

</Trigger>

</Style.Triggers>

</Style>

</ListBox.ItemContainerStyle>

<ListBox.ItemTemplate>

<DataTemplate>

<materialDesign:Card Margin="0,10" Height="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="7.5\*" />

<ColumnDefinition Width="2.5\*" />

</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0"

Text="{Binding name}"

Margin="15,10,0,0.5" FontSize="40"

HorizontalAlignment="Left" Width="546"/>

<Button Grid.Column="1"

Command="{Binding DataContext.ChooseEventCommand,

RelativeSource={RelativeSource AncestorType=ListBox}}"

CommandParameter="{Binding SelectedIndex, ElementName=listBox}"

Content="Подать заявку" Height="Auto"

BorderBrush="{DynamicResource buttonBackground}"

Background="{DynamicResource buttonBackground}"

Margin="15,29,16,30" Grid.RowSpan="2"/>

<DockPanel Grid.Row="1" Grid.ColumnSpan="2">

<TextBlock

Text="Организатор:" Margin="15,5,2,5.5"/>

<TextBlock

Text="{Binding org.name}" Margin="0,5,25,5.5" />

<TextBlock

Text="Дата проведения:" Margin="0,5,2,5.5" />

<TextBlock

Text="{Binding dateStart}" Margin="0,5,25,5.5" />

</DockPanel>

</Grid>

</materialDesign:Card>

</DataTemplate>

</ListBox.ItemTemplate>

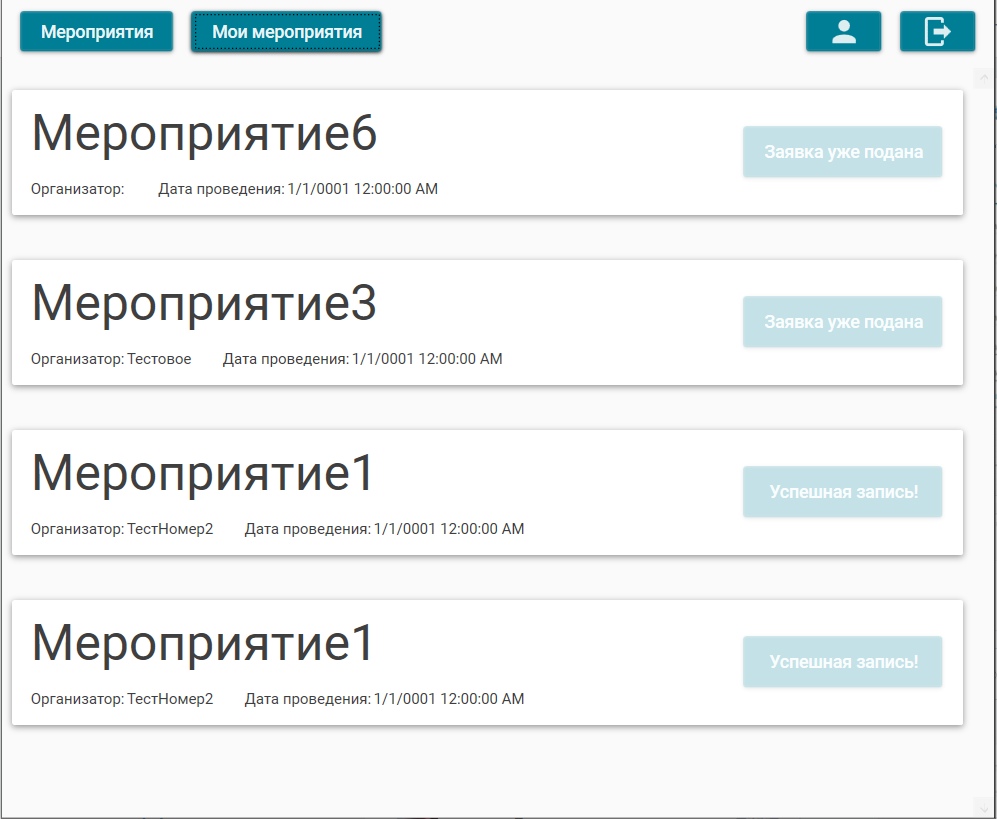
</ListBox>

В этом случае используется **ListBox**, который в отличии от **DataGrid** позволяет создавать список разных элементов управления (тут – карточки), а не только данные в таблице отображать. При наведении на карточку та выделяется. Внутри самой карточки находятся текстовые блоки и кнопка «Подать заявку».

View-Model ничем не отличается от остальных.

### Мероприятия, на которые была подана заявка

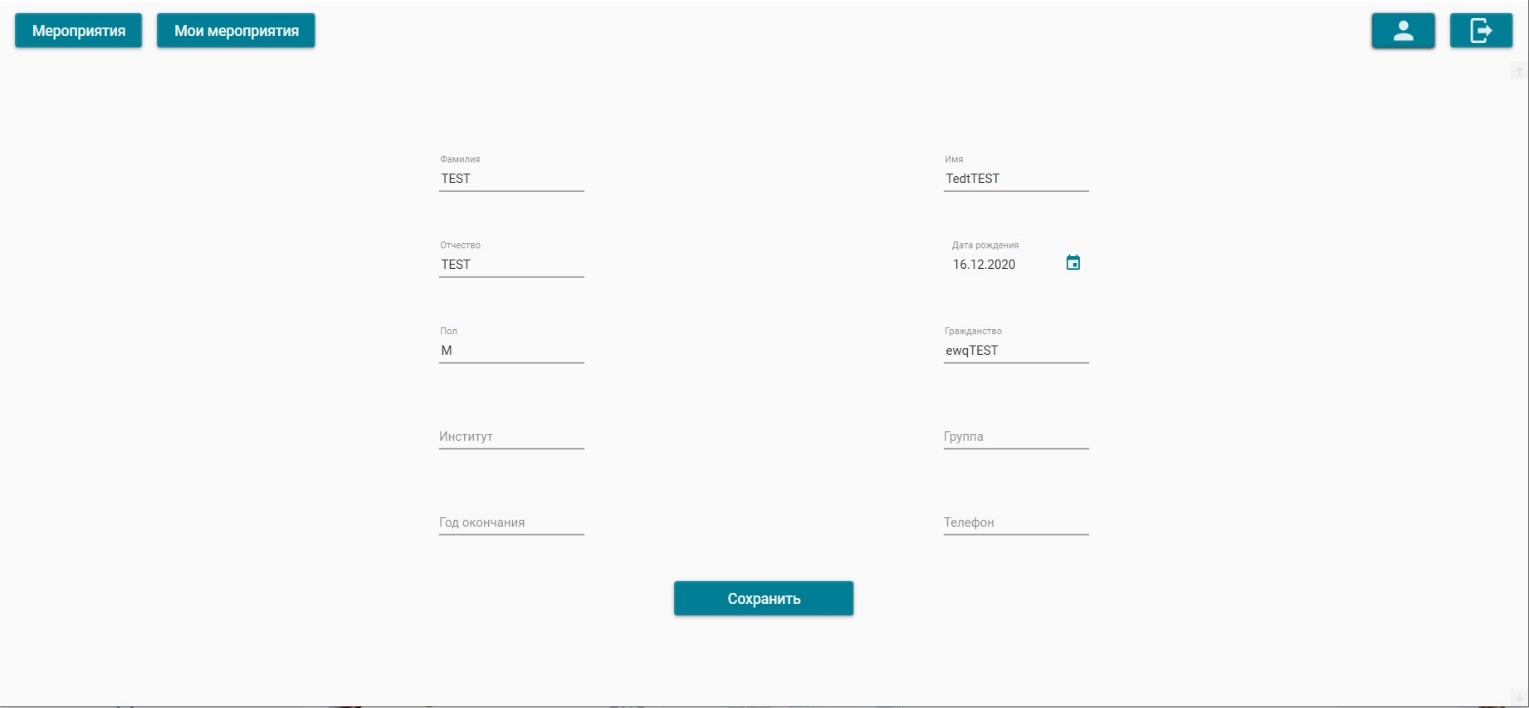
Вот представлениие:



Как видно – кнопки стали неактивны. Если заявка подана, но её еще не приняли – пишется «Заявка уже подана», а в случае приянтия – «Успешная запись».

### Информация о пользователе

Представление:



Здесь отображается все данные пользователя – дело в том, что при создании пользователя создается волонтер в базе с такими же ФИО и почтой. А при удалении удаляется такой волонтер. То есть сначала нужно создать пользователя, а затем зайти на эту вкладку и заполнить необходимую информацию.

# Вывод

В ходе работы над курсовым проектом мной были изучены подходы к решению задач в реляционных базах данных, создание REST API сервера, а также создание интерфейса на WPF.

Конечно, проект еще далек от уровня, когда его можно будет использовать на практике, потому что работы достаточно много и в течении одного семестра разработать полноценное приложение довольно сложно, но все же были реализованы самые необходимые функции. Так что, можно считать, цель была достигнута.

Мир БД огромен и интересен. Многие моменты в тех или иных базах спроектированы достаточно понятно на поверхности, но запутанно в глубине – остается только разбираться со временем.

Подход к созданию интерфейсов в WPF дает возможность очень гибко настроить все моменты, а язык C# является лаконичным и довольно приятным в плане синтаксиса.

# Список литературы.

1. К. Дж. Дэйт. Введение в системы баз данных. 8-е издание. – Москва: Вильямс, 2005г. – 1327с.
2. Ицик Бен-Ган. Microsoft SQL Server 2012. Основы T-SQL. – Москва: ЭКСМО, 2005г. – 401с.
3. The PostgreSQL Global Development Group. Документация к Postgres Pro 9.5.20.1 [Электронный ресурс]. – <https://postgrespro.ru/docs/postgrespro/9.5/index>
4. Герберт Шилдт. Java. Полное руководство. – Москва: Диалектика, 2018г. – 1489с.
5. WPF. Краткий курс. [Электронный ресурс]. – <https://metanit.com/sharp/wpf/1.php>
6. WPF. Официальная документация Microsoft. [Электронный ресурс]. – <https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netframeworkdesktop-4.8&preserve-view=true>

Приложение

## Код сервера

VolunteerCenterApplication.java

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class VolunteerCenterApplication

{

public static void main(String[] args)

{

SpringApplication.run(VolunteerCenterApplication.class, args);

}

}

SecurityConfig.java

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import dino.jwtapp.security.jwt.JwtConfigurer;

import dino.jwtapp.security.jwt.JwtTokenProvider;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter

{

private final JwtTokenProvider jwtTokenProvider;

private static final String ADMIN\_ENDPOINT = "/api/admin/\*\*";

private static final String USER\_ENDPOINT = "/api/user/\*\*";

private static final String AUTH\_ENDPOINT = "/api/auth";

private static final String REG\_ENDPOINT = "/api/register";

@Autowired

public SecurityConfig(JwtTokenProvider jwtTokenProvider)

{

this.jwtTokenProvider = jwtTokenProvider;

}

@Bean

@Override

public AuthenticationManager authenticationManager() throws Exception

{

return super.authenticationManager();

}

@Bean

public BCryptPasswordEncoder passwordEncoder()

{

return new BCryptPasswordEncoder();

}

@Override

protected void configure(HttpSecurity http) throws Exception

{

http

.httpBasic().disable()

.csrf().disable()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)

.and()

.authorizeRequests()

.antMatchers(USER\_ENDPOINT).hasRole("USER")

.antMatchers(ADMIN\_ENDPOINT).hasRole("ADMIN")

.antMatchers(AUTH\_ENDPOINT, REG\_ENDPOINT).permitAll()

.anyRequest().authenticated()

.and()

.apply(new JwtConfigurer(jwtTokenProvider));

}

}

Volunteer.java

import lombok.Data;

import javax.persistence.\*;

import java.sql.Date;

import java.util.List;

@Entity

@Table(name = "volunteers")

@Data

public class Volunteer

{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "patronymic")

private String patronymic;

@Column(name = "date\_of\_birth")

private Date dateOfBirth;

@Column(name = "sex")

private String sex;

@Column(name = "citizenship")

private String citizenship;

@Column(name = "institute")

private String institute;

@Column(name = "course\_of\_study")

private Integer courseOfStudy;

@Column(name = "form\_of\_study")

private String formOfStudy;

@Column(name = "budget\_education")

private Boolean budgetEducation;

@Column(name = "study\_group")

private String studyGroup;

@Column(name = "year\_of\_graduation")

private Integer yearOfGraduation;

@Column(name = "clothing\_size")

private String clothingSize;

@Column(name = "telephone")

private String telephone;

@Column(name = "email")

private String email;

@Column(name = "address")

private String address;

@Column(name = "employment")

private Boolean employment;

@ManyToOne

@JoinColumn(name = "loyalty\_level")

private LoyaltyCard loyaltyCard;

@ManyToMany

@JoinTable(

name = "activities\_of\_volunteers",

joinColumns = {@JoinColumn(name = "volunteer\_id")},

inverseJoinColumns = {@JoinColumn(name = "activity\_id")})

private List<Activity> activities;

@ManyToMany

@JoinTable(

name = "events\_of\_volunteers",

joinColumns = {@JoinColumn(name = "volunteer\_id")},

inverseJoinColumns = {@JoinColumn(name = "event\_id")})

private List<Event> events;

@ManyToMany

@JoinTable(

name = "events\_requests\_volunteer",

joinColumns = {@JoinColumn(name = "volunteer\_id")},

inverseJoinColumns = {@JoinColumn(name = "event\_id")})

private List<Event> requestedEvents;

@ManyToMany

@JoinTable(

name = "langs\_of\_volunteers",

joinColumns = {@JoinColumn(name = "volunteer\_id")},

inverseJoinColumns = {@JoinColumn(name = "language\_id")})

private List<Language> languages;

@ManyToMany

@JoinTable(

name = "media\_skills\_of\_volunteers",

joinColumns = {@JoinColumn(name = "volunteer\_id")},

inverseJoinColumns = {@JoinColumn(name = "media\_skill\_id")})

private List<MediaSkill> mediaSkills;

@ManyToMany

@JoinTable(

name = "scholarships\_of\_volunteers",

joinColumns = {@JoinColumn(name = "volunteer\_id")},

inverseJoinColumns = {@JoinColumn(name = "scholarship\_id")})

private List<Scholarship> scholarships;

}

User.java

import lombok.Data;

import javax.persistence.\*;

import java.util.List;

@Entity

@Table(name = "users")

@Data

public class User

{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "username")

private String username;

@Column(name = "email")

private String email;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "patronymic")

private String patronymic;

@Column(name = "password")

private String password;

@ManyToOne

@JoinColumn(name = "id\_role")

private Role role;

@OneToOne

@JoinColumn(name = "volunteer\_id")

private Volunteer volunteer;

}

Role.java

import lombok.Data;

import javax.persistence.\*;

@Entity

@Table(name = "roles")

@Data

public class Role

{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

}

Organization.java

import lombok.Data;

import javax.persistence.\*;

@Entity

@Table(name = "organizations")

@Data

public class Organization

{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

@Column(name = "contact\_person")

private String contactPerson;

@Column(name = "telephone")

private String telephone;

@Column(name = "email")

private String email;

}

Event.java

import lombok.Data;

import javax.persistence.\*;

import java.sql.Timestamp;

import java.util.List;

@Entity

@Table(name = "events")

@Data

public class Event

{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

@Column(name = "short\_name")

private String shortName;

@Column(name = "date\_begin")

private Timestamp dateBegin;

@Column(name = "date\_end")

private Timestamp dateEnd;

@ManyToOne

@JoinColumn(name = "organizer")

private Organization org;

@ManyToMany

@JoinTable(

name = "events\_of\_volunteers",

joinColumns = {@JoinColumn(name = "event\_id")},

inverseJoinColumns = {@JoinColumn(name = "volunteer\_id")})

private List<VolunteerInfo> volunteers;

@ManyToMany

@JoinTable(

name = "events\_requests\_volunteer",

joinColumns = {@JoinColumn(name = "event\_id")},

inverseJoinColumns = {@JoinColumn(name = "volunteer\_id")})

private List<VolunteerInfo> requestedVols;

}

EventsRepository.java

import dino.jwtapp.model.Event;

import dino.jwtapp.model.Organization;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface EventsRepository extends JpaRepository<Event, Long>

{

List<Event> findByName(String name);

List<Event> findByOrg(Organization org);

}

OrganizationRepository.java

import dino.jwtapp.model.Organization;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface OrganizationRepository extends JpaRepository<Organization, Long>

{

List<Organization> findByName(String name);

}

UserRepository.java

import dino.jwtapp.model.User;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long>

{

User findByUsername(String name);

void deleteByUsername(String login);

}

VolunteerRepository.java

import dino.jwtapp.model.Event;

import dino.jwtapp.model.Volunteer;

import org.springframework.data.jpa.repository.JpaRepository;

import java.sql.Date;

import java.util.List;

import java.util.Optional;

public interface VolunteerRepository extends JpaRepository<Volunteer, Long>

{

List<Volunteer> findByFirstName(String volunteerFirstName);

List<Volunteer> findByLastName(String volunteerLastName);

List<Volunteer> findBySex(String sex);

Optional<Volunteer> findByFirstNameAndLastNameAndPatronymicAndEmail(

String firstName, String lastName, String Patronymic, String email);

List<Volunteer> findByEvents(Event event);

}

AdminRestController.java

import dino.jwtapp.dto.EventRequestDto;

import dino.jwtapp.dto.OrganizationDto;

import dino.jwtapp.model.\*;

import dino.jwtapp.service.InfoService;

import dino.jwtapp.service.UserService;

import dino.jwtapp.service.VolunteerService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpServletRequest;

import java.util.List;

import java.util.Optional;

@Slf4j

@RestController

@RequestMapping(value = "/api/admin/")

public class AdminRestController

{

private final UserService userService;

private final InfoService infoService;

private final VolunteerService volunteerService;

@Autowired

public AdminRestController(UserService userService, InfoService infoService, VolunteerService volunteerService)

{

this.userService = userService;

this.infoService = infoService;

this.volunteerService = volunteerService;

}

@GetMapping("users/get")

public ResponseEntity<List<User>> getAllUsers()

{

log.info("Get all users:");

List<User> result = userService.getAll();

return result != null

? new ResponseEntity<>(result, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@RequestMapping(value = "delete/{login}")

public HttpStatus deleteByUsername(@PathVariable(name = "login") String username)

{

if (userService.deleteByUsername(username))

return HttpStatus.OK;

return HttpStatus.BAD\_REQUEST;

}

@GetMapping("events/get")

public ResponseEntity<List<Event>> getAllEvents()

{

log.info("Get all events:");

List<Event> result = infoService.getAllEvents();

return result != null

? new ResponseEntity<>(result, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@GetMapping("events/find")

public ResponseEntity<List<Event>> findEvents(HttpServletRequest request)

{

List<Event> vols = infoService.findEventsByName(request.getParameter("name"));

log.info("Get events by name: " + request.getParameter("name"));

return vols != null

? new ResponseEntity<>(vols, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("events/adding")

public Object addEvent(@RequestBody EventRequestDto eventRequestDto)

{

Event event = new Event();

event.setId(eventRequestDto.getId());

event.setName(eventRequestDto.getName());

event.setShortName(eventRequestDto.getShortName());

event.setDateBegin(eventRequestDto.getDateBegin());

event.setDateEnd(eventRequestDto.getDateEnd());

event.setOrg(eventRequestDto.getOrg());

event.setVolunteers(eventRequestDto.getVolunteers());

for (VolunteerInfo volInfo : event.getVolunteers())

{

Optional<Volunteer> vol = volunteerService.findById(volInfo.getId());

if (vol.get().getRequestedEvents() != null)

vol.get().getRequestedEvents().remove(event);

else

vol.get().getEvents().add(event);

volunteerService.save(vol.get());

}

try

{

infoService.addEvent(event);

return "OK";

} catch (Exception e)

{

log.info("Added - Event: {} is busy", event.getName());

return "NO";

}

}

@RequestMapping(value = "events/delete/{id}")

public HttpStatus deleteEventById(@PathVariable(name = "id") Long id)

{

if (infoService.deleteEventById(id))

return HttpStatus.OK;

return HttpStatus.BAD\_REQUEST;

}

@GetMapping("orgs/get")

public ResponseEntity<List<Organization>> getAllOrgs()

{

log.info("Get all events:");

List<Organization> result = infoService.getAllOrgs();

return result != null

? new ResponseEntity<>(result, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@GetMapping("orgs/find")

public ResponseEntity<List<Organization>> findOrgs(HttpServletRequest request)

{

List<Organization> vols = infoService.findOrgsByName(request.getParameter("name"));

log.info("Get orgs by name: " + request.getParameter("name"));

return vols != null

? new ResponseEntity<>(vols, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("orgs/adding")

public Object addOrg(@RequestBody OrganizationDto orgDto)

{

Organization org = new Organization();

org.setId(org.getId());

org.setName(orgDto.getName());

org.setContactPerson(orgDto.getContactPerson());

org.setEmail(orgDto.getEmail());

org.setTelephone(orgDto.getTelephone());

try

{

infoService.addOrg(org);

return "OK";

} catch (Exception e)

{

log.info("Added - Event: {} is busy", org.getName());

return "NO";

}

}

@GetMapping("orgs/get\_events")

public ResponseEntity<List<Event>> findEventById(HttpServletRequest request)

{

Long id = Long.parseLong(request.getParameter("id"));

log.info("Get events by org\_id: " + id);

Optional<Organization> org = infoService.findOrgById(id);

if (org.isEmpty())

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

List<Event> result = infoService.findEventsByOrg(org.get());

return result.size() != 0

? new ResponseEntity<>(result, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@RequestMapping(value = "orgs/delete/{id}")

public HttpStatus deleteOrgById(@PathVariable(name = "id") Long id)

{

if (infoService.deleteOrgById(id))

return HttpStatus.OK;

return HttpStatus.BAD\_REQUEST;

}

@PostMapping("users/set\_role")

public Object setUserRole(HttpServletRequest request)

{

Long id = Long.parseLong(request.getParameter("id"));

Optional<User> user = userService.findById(id);

if (user.isEmpty())

return "NO";

Role role = infoService.findRoleByName(request.getParameter("role"));

user.get().setRole(role);

log.info("Set to {} role: {}" + user.get().getUsername(), role.getName());

try

{

userService.save(user.get());

return "OK";

} catch (Exception e)

{

log.info("Register - Username: {} is busy", user.get().getUsername());

return "NO";

}

}

}

AuthenticationRestController.java

import dino.jwtapp.dto.AuthenticationRequestDto;

import dino.jwtapp.dto.RegistrationRequestDto;

import dino.jwtapp.model.User;

import dino.jwtapp.security.jwt.JwtTokenProvider;

import dino.jwtapp.service.UserService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.BadCredentialsException;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.AuthenticationException;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;

import java.util.Map;

@Slf4j

@RestController

@RequestMapping(value = "/api/")

public class AuthenticationRestController

{

private final AuthenticationManager authenticationManager;

private final JwtTokenProvider jwtTokenProvider;

private final UserService userService;

@Autowired

public AuthenticationRestController(AuthenticationManager authenticationManager, JwtTokenProvider jwtTokenProvider, UserService userService)

{

this.authenticationManager = authenticationManager;

this.jwtTokenProvider = jwtTokenProvider;

this.userService = userService;

}

@PostMapping("register")

public Object register(@RequestBody RegistrationRequestDto registrationRequestDto)

{

User user = new User();

user.setUsername(registrationRequestDto.getUsername());

user.setPassword(registrationRequestDto.getPassword());

user.setEmail(registrationRequestDto.getEmail());

user.setLastName(registrationRequestDto.getLastName());

user.setFirstName(registrationRequestDto.getFirstName());

user.setPatronymic(registrationRequestDto.getPatronymic());

try

{

userService.register(user);

return "OK";

} catch (Exception e)

{

log.info("Register - Username: {} is busy", user.getUsername());

return "NO";

}

}

@PostMapping("auth")

public ResponseEntity login(@RequestBody AuthenticationRequestDto requestDto)

{

try

{

String username = requestDto.getUsername();

authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username, requestDto.getPassword()));

User user = userService.findByUsername(username);

if (user == null)

throw new UsernameNotFoundException("User with username: " + username + " not found");

String token = jwtTokenProvider.createToken(username, user.getRole());

Map<Object, Object> response = new HashMap<>();

response.put("id", user.getId().toString());

response.put("role", user.getRole().getName());

response.put("username", username);

response.put("token", token);

return ResponseEntity.ok(response);

} catch (AuthenticationException e)

{

throw new BadCredentialsException("Invalid username or password");

}

}

}

UserRestController.java

import dino.jwtapp.dto.EventRequestDto;

import dino.jwtapp.model.Event;

import dino.jwtapp.model.Volunteer;

import dino.jwtapp.service.InfoService;

import dino.jwtapp.service.VolunteerService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import dino.jwtapp.model.User;

import dino.jwtapp.service.UserService;

import javax.servlet.http.HttpServletRequest;

import java.security.Principal;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Optional;

@Slf4j

@RestController

@RequestMapping(value = "/api/user/")

public class UserRestController

{

private final UserService userService;

private final InfoService infoService;

private final VolunteerService volunteerService;

@Autowired

public UserRestController(UserService userService, InfoService infoService, VolunteerService volunteerService)

{

this.userService = userService;

this.infoService = infoService;

this.volunteerService = volunteerService;

}

@PostMapping("volunteer/set")

public Object setVol(HttpServletRequest request)

{

Long userId = Long.parseLong(request.getParameter("id"));

log.info("Set vol into user (User request) by UserId = {}", userId);

Optional<User> foundedUser = userService.findById(userId);

if (foundedUser.isEmpty())

return new ResponseEntity(HttpStatus.NOT\_FOUND);

User user = foundedUser.get();

Optional<Volunteer> vol = volunteerService.findUsersVol(

user.getFirstName(), user.getLastName(), user.getPatronymic(), user.getEmail()

);

if (vol.isEmpty())

{

Volunteer volunteer = new Volunteer();

volunteer.setFirstName(user.getFirstName());

volunteer.setLastName(user.getLastName());

volunteer.setPatronymic(user.getPatronymic());

volunteer.setEmail(user.getEmail());

volunteerService.save(volunteer);

user.setVolunteer(volunteer);

} else

{

user.setVolunteer(vol.get());

}

userService.save(user);

return new ResponseEntity(HttpStatus.OK);

}

@GetMapping("profile")

public ResponseEntity<User> getUser(HttpServletRequest request)

{

Long userId = Long.parseLong(request.getParameter("id"));

log.info("Get user (User request) by UserId = {}", userId);

Optional<User> user = userService.findById(userId);

return user.isPresent()

? new ResponseEntity<>(user.get(), HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@PostMapping("profile/update")

public Object updateUser(@RequestBody User user)

{

log.info("Update user (User request) {}", user);

try

{

volunteerService.save(user.getVolunteer());

userService.save(user);

return "OK";

} catch (Exception e)

{

return "NO";

}

}

@GetMapping("info")

public ResponseEntity<User> getUserInfo(HttpServletRequest request)

{

Long id = Long.parseLong(request.getParameter("id"));

log.info("Get info about user, id - {}", id);

Optional<User> user = userService.findById(id);

return user.isPresent()

? new ResponseEntity<>(user.get(), HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@GetMapping("events/all")

public ResponseEntity<List<Event>> getAllEvents(HttpServletRequest request)

{

Long id = Long.parseLong(request.getParameter("id"));

Optional<User> user = userService.findById(id);

log.info("Get all events, request from User, id: {}", id);

List<Event> result = infoService.getAllEvents();

if (user.isEmpty() || user.get().getVolunteer() == null)

new ResponseEntity<>(HttpStatus.NOT\_FOUND);

result.removeIf(event -> user.get().getVolunteer().getRequestedEvents().contains(event) ||

user.get().getVolunteer().getEvents().contains(event));

return new ResponseEntity<>(result, HttpStatus.OK);

}

@GetMapping("events/my\_events")

public ResponseEntity getUserEvents(HttpServletRequest request)

{

Long userId = Long.parseLong(request.getParameter("id"));

log.info("Get all events (User request) by UserId = {}", userId);

Optional<User> user = userService.findById(userId);

if (user.isEmpty())

return ResponseEntity.badRequest().body("User hasn't found");

else if (user.get().getVolunteer() == null)

return ResponseEntity.badRequest().body("User's volunteer profile hasn't found");

Map<Object, Object> response = new HashMap<>();

response.put("events", user.get().getVolunteer().getEvents());

response.put("requestedEvents", user.get().getVolunteer().getRequestedEvents());

return ResponseEntity.ok(response);

}

@PostMapping("events/request")

public Object requestEvent(HttpServletRequest request)

{

Long userId = Long.parseLong(request.getParameter("user\_id"));

Long eventId = Long.parseLong(request.getParameter("event\_id"));

log.info("Request event(id) = {} (User request) by user, Id = {}", eventId, userId);

Optional<User> user = userService.findById(userId);

Optional<Event> event = infoService.findEventById(eventId);

if (user.isEmpty() || event.isEmpty() || user.get().getVolunteer() == null)

return "User has no volunteer or there is no event";

user.get().getVolunteer().getRequestedEvents().add(event.get());

volunteerService.save(user.get().getVolunteer());

return "OK";

}

}

VolunteerRestController.java

import dino.jwtapp.dto.AddVolunteerRequestDto;

import dino.jwtapp.dto.FindVolunteerRequestDto;

import dino.jwtapp.model.Event;

import dino.jwtapp.model.Volunteer;

import dino.jwtapp.model.VolunteerInfo;

import dino.jwtapp.service.InfoService;

import dino.jwtapp.service.VolunteerInfoService;

import dino.jwtapp.service.VolunteerService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import java.util.List;

import java.util.Optional;

@Slf4j

@RestController

@RequestMapping(value = "/api/admin/volunteers/")

public class VolunteerRestController

{

private final VolunteerService volunteerService;

private final VolunteerInfoService volunteerInfoService;

private final InfoService infoService;

@Autowired

public VolunteerRestController(VolunteerService volunteerService, VolunteerInfoService volunteerInfoService, InfoService infoService)

{

this.volunteerService = volunteerService;

this.volunteerInfoService = volunteerInfoService;

this.infoService = infoService;

}

@PostMapping("adding")

public Object add(@RequestBody AddVolunteerRequestDto addVolunteerRequestDto)

{

Volunteer volunteer = new Volunteer();

volunteer.setId(addVolunteerRequestDto.getId());

volunteer.setFirstName(addVolunteerRequestDto.getFirstName());

volunteer.setLastName(addVolunteerRequestDto.getLastName());

volunteer.setPatronymic(addVolunteerRequestDto.getPatronymic());

volunteer.setDateOfBirth(addVolunteerRequestDto.getDateOfBirth());

volunteer.setSex(addVolunteerRequestDto.getSex());

volunteer.setCitizenship(addVolunteerRequestDto.getCitizenship());

volunteer.setInstitute(addVolunteerRequestDto.getInstitute());

volunteer.setCourseOfStudy(addVolunteerRequestDto.getCourseOfStudy());

volunteer.setFormOfStudy(addVolunteerRequestDto.getFormOfStudy());

volunteer.setBudgetEducation(addVolunteerRequestDto.getBudgetEducation());

volunteer.setStudyGroup(addVolunteerRequestDto.getStudyGroup());

volunteer.setYearOfGraduation(addVolunteerRequestDto.getYearOfGraduation());

volunteer.setClothingSize(addVolunteerRequestDto.getClothingSize());

volunteer.setTelephone(addVolunteerRequestDto.getTelephone());

volunteer.setEmail(addVolunteerRequestDto.getEmail());

volunteer.setAddress(addVolunteerRequestDto.getAddress());

volunteer.setEmployment(addVolunteerRequestDto.getEmployment());

try

{

if (volunteerService.save(volunteer) == null)

return "Already existed";

return "OK";

} catch (Exception e)

{

log.info("Register - Volunteer: {} is busy", volunteer.getFirstName());

return "NO";

}

}

@PostMapping("request\_completed")

public Object requestCompleted(@RequestBody Event event)

{

try

{

infoService.addEvent(event);

return "OK";

} catch (Exception e)

{

log.info("IN requestCompleted - something went wrong");

return "NO";

}

}

@GetMapping("get\_all")

public ResponseEntity<List<Volunteer>> getAllVolunteers()

{

log.info("Get all volunteers:");

List<Volunteer> result = volunteerService.getAll();

return result != null

? new ResponseEntity<>(result, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@GetMapping("get")

public ResponseEntity<Volunteer> findById(HttpServletRequest request)

{

Long id = Long.parseLong(request.getParameter("id"));

log.info("Get volunteer by id: " + id);

Optional<Volunteer> result = volunteerService.findById(id);

return result.isPresent()

? new ResponseEntity<>(result.get(), HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@GetMapping("get\_by\_event")

public ResponseEntity<List<Volunteer>> findByEvent(HttpServletRequest request)

{

Long id = Long.parseLong(request.getParameter("event\_id"));

log.info("Get volunteer by event id: " + id);

Optional<Event> event = infoService.findEventById(id);

if (event.isEmpty())

new ResponseEntity<>(HttpStatus.NOT\_FOUND);

List<Volunteer> result = volunteerService.findByEvent(event.get());

return new ResponseEntity<>(result, HttpStatus.OK);

}

@GetMapping("get\_views")

public ResponseEntity<List<VolunteerInfo>> getAllVolunteersView()

{

log.info("Get all volunteers:");

List<VolunteerInfo> result = volunteerInfoService.getAllViews();

return result != null

? new ResponseEntity<>(result, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@RequestMapping(value = "delete/{id}")

public HttpStatus deleteVolById(@PathVariable(name = "id") String id)

{

if (volunteerService.deleteById(Long.valueOf(

id.replace("{", "")

.replace("}", ""))))

return HttpStatus.OK;

return HttpStatus.BAD\_REQUEST;

}

@GetMapping("find")

public ResponseEntity<List<Volunteer>> findVolunteers(HttpServletRequest request)

{

List<Volunteer> vols = volunteerService.findByFirstName(request.getParameter("first\_name"));

log.info("Get volunteers by first name: " + request.getParameter("first\_name"));

return vols != null

? new ResponseEntity<>(vols, HttpStatus.OK)

: new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

JwtUserDetailsService.java

import dino.jwtapp.model.User;

import dino.jwtapp.security.jwt.JwtUser;

import dino.jwtapp.security.jwt.JwtUserFactory;

import dino.jwtapp.service.UserService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;

@Service

@Slf4j

public class JwtUserDetailsService implements UserDetailsService

{

private final UserService userService;

@Autowired

public JwtUserDetailsService(UserService userService)

{

this.userService = userService;

}

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException

{

User user = userService.findByUsername(username);

if (user == null)

{

throw new UsernameNotFoundException("User with username: " + username + " not found");

}

JwtUser jwtUser = JwtUserFactory.create(user);

log.info("loadByUsername - user with username: {} successfully loaded", username);

return jwtUser;

}

}

JwtAuthenticationException.java

import org.springframework.security.core.AuthenticationException;

public class JwtAuthenticationException extends AuthenticationException

{

public JwtAuthenticationException(String msg, Throwable cause)

{

super(msg, cause);

}

public JwtAuthenticationException(String msg)

{

super(msg);

}

}

JwtConfigurer.java

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.config.annotation.SecurityConfigurerAdapter;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.DefaultSecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

public class JwtConfigurer extends SecurityConfigurerAdapter<DefaultSecurityFilterChain, HttpSecurity>

{

@Autowired

private JwtTokenProvider jwtTokenProvider;

public JwtConfigurer(JwtTokenProvider jwtTokenProvider)

{

this.jwtTokenProvider = jwtTokenProvider;

}

@Override

public void configure(HttpSecurity httpSecurity)

{

JwtTokenFilter jwtTokenFilter = new JwtTokenFilter(jwtTokenProvider);

httpSecurity.addFilterBefore(jwtTokenFilter, UsernamePasswordAuthenticationFilter.class);

}

}

JwtTokenFilter.java

import lombok.SneakyThrows;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.web.filter.GenericFilterBean;

import javax.servlet.FilterChain;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.HttpServletRequest;

public class JwtTokenFilter extends GenericFilterBean

{

private JwtTokenProvider jwtTokenProvider;

public JwtTokenFilter(JwtTokenProvider jwtTokenProvider)

{

this.jwtTokenProvider = jwtTokenProvider;

}

@SneakyThrows

@Override

public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain)

{

String token = jwtTokenProvider.resolveToken((HttpServletRequest) servletRequest);

if (token != null && jwtTokenProvider.validateToken(token))

{

UsernamePasswordAuthenticationToken authentication = jwtTokenProvider.getAuthentication(token);

if (authentication != null)

SecurityContextHolder.getContext().setAuthentication(authentication);

}

filterChain.doFilter(servletRequest, servletResponse);

}

}

JwtTokenProvider.java

import io.jsonwebtoken.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.stereotype.Component;

import dino.jwtapp.model.Role;

import dino.jwtapp.security.JwtUserDetailsService;

import javax.annotation.PostConstruct;

import javax.servlet.http.HttpServletRequest;

import java.util.Base64;

import java.util.Date;

@Component

public class JwtTokenProvider

{

@Value("${jwt.token.secret}")

private String secret;

@Value("${jwt.token.expired}")

private long validityInMilliseconds;

@Autowired

private JwtUserDetailsService jwtUserDetailsService;

public String createToken(String login, Role role)

{

Claims claims = Jwts.claims().setSubject(login);

claims.put("role", getRoleNames(role));

Date now = new Date();

Date validaty = new Date(now.getTime() + validityInMilliseconds);

return Jwts.builder()

.setClaims(claims)

.setIssuedAt(now)

.setExpiration(validaty)

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

@PostConstruct

protected void init()

{

secret = Base64.getEncoder().encodeToString(secret.getBytes());

}

public UsernamePasswordAuthenticationToken getAuthentication(String token)

{

UserDetails userDetails = this.jwtUserDetailsService.loadUserByUsername(getUsername(token)); //!!!!!

return new UsernamePasswordAuthenticationToken(userDetails, "", userDetails.getAuthorities());

}

String getUsername(String token)

{

return Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody().getSubject();

}

public String resolveToken(HttpServletRequest request)

{

String bearerToken = request.getHeader("Authorization");

if (bearerToken != null && bearerToken.startsWith("Bearer\_"))

return bearerToken.substring(7, bearerToken.length());

return null;

}

public boolean validateToken(String token) throws JwtAuthenticationException

{

try

{

Jws<Claims> claims = Jwts.parser().setSigningKey(secret).parseClaimsJws(token);

if (claims.getBody().getExpiration().before(new Date()))

return false;

return true;

} catch (JwtException | IllegalArgumentException e)

{

throw new JwtAuthenticationException("JWT token is expired or invalid");

}

}

private String getRoleNames(Role userRole)

{

return userRole.getName();

}

}

JwtUser.java

import com.fasterxml.jackson.annotation.JsonIgnore;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;

public class JwtUser implements UserDetails

{

private final Long id;

private final String username;

private final String firstName;

private final String lastName;

private final String patronymic;

private final String email;

private final String password;

private final boolean enabled;

private final Collection<? extends GrantedAuthority> authorities;

public JwtUser(

Long id,

String username,

String firstName,

String lastName,

String patronymic,

String email,

String password,

boolean enabled,

Collection<? extends GrantedAuthority> authorities)

{

this.id = id;

this.username = username;

this.firstName = firstName;

this.lastName = lastName;

this.patronymic = patronymic;

this.email = email;

this.password = password;

this.enabled = enabled;

this.authorities = authorities;

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities()

{

return authorities;

}

@JsonIgnore

@Override

public String getPassword()

{

return password;

}

@Override

public String getUsername()

{

return username;

}

@JsonIgnore

@Override

public boolean isAccountNonExpired()

{

return true;

}

@JsonIgnore

@Override

public boolean isAccountNonLocked()

{

return true;

}

@JsonIgnore

@Override

public boolean isCredentialsNonExpired()

{

return true;

}

@JsonIgnore

@Override

public boolean isEnabled()

{

return enabled;

}

@JsonIgnore

public Long getId()

{

return id;

}

public String getFirstName()

{

return firstName;

}

public String getLastName()

{

return lastName;

}

public String getPatronymic()

{

return patronymic;

}

public String getEmail()

{

return email;

}

}

JwtUserFactory.java

import dino.jwtapp.model.User;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import java.util.Collections;

public class JwtUserFactory

{

public JwtUserFactory() { }

public static JwtUser create(User user)

{

return new JwtUser(

user.getId(),

user.getUsername(),

user.getFirstName(),

user.getLastName(),

user.getPatronymic(),

user.getEmail(),

user.getPassword(),

true,

Collections.singletonList(new SimpleGrantedAuthority(user.getRole().getName()))

);

}

}

InfoServiceImpl.java

import dino.jwtapp.model.Event;

import dino.jwtapp.model.Organization;

import dino.jwtapp.model.Role;

import dino.jwtapp.repository.EventsRepository;

import dino.jwtapp.repository.OrganizationRepository;

import dino.jwtapp.repository.RoleRepository;

import dino.jwtapp.service.InfoService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

@Slf4j

public class InfoServiceImpl implements InfoService

{

private final EventsRepository eventsRepository;

private final OrganizationRepository organizationRepository;

private final RoleRepository roleRepository;

@Autowired

public InfoServiceImpl(EventsRepository eventsRepository, OrganizationRepository organizationRepository, RoleRepository roleRepository)

{

this.eventsRepository = eventsRepository;

this.organizationRepository = organizationRepository;

this.roleRepository = roleRepository;

}

@Override

public Event addEvent(Event event)

{

Event addedEvent = eventsRepository.save(event);

log.info("IN addEvent - event: {} successfully added", addedEvent);

return addedEvent;

}

@Override

public List<Event> getAllEvents()

{

return eventsRepository.findAll();

}

@Override

public boolean deleteEventById(Long id)

{

if (eventsRepository.findById(id).isPresent())

return false;

eventsRepository.deleteById(id);

return true;

}

@Override

public Optional<Event> findEventById(Long id)

{

Optional<Event> founded = eventsRepository.findById(id);

log.info("IN findEventById - event: {} found", founded);

return founded;

}

@Override

public List<Event> findEventsByOrg(Organization org)

{

List<Event> events = eventsRepository.findByOrg(org);

log.info("IN findByOrg - found {} events by org\_id: {}", events.size(), org.getId());

return events;

}

@Override

public List<Event> findEventsByName(String name)

{

List<Event> events = eventsRepository.findByName(name);

log.info("IN findEventsByName - found {} events", events.size());

return events;

}

@Override

public Organization addOrg(Organization org)

{

Organization addedOrg = organizationRepository.save(org);

log.info("IN addOrg - org: {} successfully added", addedOrg);

return addedOrg;

}

@Override

public Optional<Organization> findOrgById(Long id)

{

Optional<Organization> founded = organizationRepository.findById(id);

log.info("IN findOrgById - org: {} found", founded);

return founded;

}

@Override

public List<Organization> getAllOrgs()

{

return organizationRepository.findAll();

}

@Override

public boolean deleteOrgById(Long id)

{

log.info("IN deleteOrgById deleting by id: {}", id);

if (organizationRepository.findById(id).isPresent())

return false;

organizationRepository.deleteById(id);

return true;

}

@Override

public List<Organization> findOrgsByName(String name)

{

List<Organization> orgs = organizationRepository.findByName(name);

log.info("IN findOrgsByName - {} orgs found", orgs.size());

return orgs;

}

@Override

public Role findRoleByName(String name)

{

Role founded = roleRepository.findByName(name);

log.info("IN findOrgById - org: {} found", founded);

return founded;

}

}

UserServiceImpl.java

import dino.jwtapp.model.Role;

import dino.jwtapp.model.User;

import dino.jwtapp.repository.RoleRepository;

import dino.jwtapp.repository.UserRepository;

import dino.jwtapp.service.UserService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Lazy;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.util.List;

import java.util.Optional;

@Service

@Slf4j

@Transactional

public class UserServiceImpl implements UserService

{

private final UserRepository userRepository;

private final RoleRepository roleRepository;

private final BCryptPasswordEncoder passwordEncoder;

@Autowired

public UserServiceImpl(UserRepository userRepository, RoleRepository roleRepository, @Lazy BCryptPasswordEncoder passwordEncoder)

{

this.userRepository = userRepository;

this.roleRepository = roleRepository;

this.passwordEncoder = passwordEncoder;

}

@Override

public void save(User user)

{

userRepository.save(user);

log.info("IN save - user: {} successfully saved", user.getUsername());

}

@Override

public User register(User user)

{

user.setPassword(passwordEncoder.encode(user.getPassword()));

System.out.println(user);

Role role = roleRepository.findByName("ROLE\_USER");

user.setRole(role);

User registerUser = userRepository.save(user);

log.info("IN register - user: {} successfully registered", registerUser);

return registerUser;

}

@Override

public List<User> getAll()

{

List<User> result = userRepository.findAll();

log.info("IN getAll - {} users found", result.size());

return result;

}

@Override

public User findByUsername(String username)

{

User result = userRepository.findByUsername(username);

log.info("IN findByUsername - user: {} found by username: {}", result, username);

return result;

}

@Override

public Optional<User> findById(Long id)

{

log.info("IN findById - userId - {}", id);

return userRepository.findById(id);

}

@Override

public void delete(Long id)

{

userRepository.deleteById(id);

log.info("IN delete - user with id: {} successfully deleted", id);

}

@Override

public boolean deleteByUsername(String username)

{

if (userRepository.findByUsername(username) == null)

return false;

userRepository.deleteByUsername(username);

return true;

}

}

VolunteerInfoServiceImpl.java

import dino.jwtapp.model.VolunteerInfo;

import dino.jwtapp.repository.VolunteerInfoRepository;

import dino.jwtapp.service.VolunteerInfoService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.util.List;

@Service

@Slf4j

@Transactional

public class VolunteerInfoServiceImpl implements VolunteerInfoService

{

private final VolunteerInfoRepository volunteerInfoRepository;

public VolunteerInfoServiceImpl(VolunteerInfoRepository volunteerInfoRepository)

{

this.volunteerInfoRepository = volunteerInfoRepository;

}

@Override

public List<VolunteerInfo> getAllViews()

{

List<VolunteerInfo> result = volunteerInfoRepository.findAll();

log.info("IN getAll - {} volunteers found", result.size());

return result;

}

}

VolunteerServiceImpl.java

import dino.jwtapp.model.Event;

import dino.jwtapp.model.Volunteer;

import dino.jwtapp.repository.VolunteerRepository;

import dino.jwtapp.service.VolunteerService;

import lombok.extern.slf4j.Slf4j;

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.sql.Date;

import java.util.List;

import java.util.Optional;

@Service

@Slf4j

@Transactional

public class VolunteerServiceImpl implements VolunteerService

{

private final VolunteerRepository volunteerRepository;

public VolunteerServiceImpl(VolunteerRepository volunteerRepository)

{

this.volunteerRepository = volunteerRepository;

}

@Override

public Volunteer save(Volunteer volunteer)

{

log.info("IN save - volunteer: {} saving", volunteer);

Optional<Volunteer> found = volunteerRepository.findByFirstNameAndLastNameAndPatronymicAndEmail(

volunteer.getFirstName(), volunteer.getLastName(), volunteer.getPatronymic(), volunteer.getEmail()

);

if (found.isPresent() && found.get().getId() != volunteer.getId())

return null;

else

return volunteerRepository.save(volunteer);

}

@Override

public List<Volunteer> getAll()

{

List<Volunteer> result = volunteerRepository.findAll();

log.info("IN getAll - {} volunteers found", result.size());

return result;

}

@Override

public Optional<Volunteer> findById(Long id)

{

log.info("IN findById - volunteer, id: {}", id);

return volunteerRepository.findById(id);

}

@Override

public Optional<Volunteer> findUsersVol(String firstName, String lastName, String Patronymic, String email)

{

log.info("IN findUsersVol - volunteer, info: {}", firstName + " " + lastName + " "

+ Patronymic + " " + email);

return volunteerRepository.findByFirstNameAndLastNameAndPatronymicAndEmail(

firstName, lastName, Patronymic, email);

}

@Override

public boolean deleteById(Long id)

{

log.info("DELETING {}", volunteerRepository.findById(id));

log.info("IN deleteById - volunteer, id: {}", id);

if (volunteerRepository.findById(id).isEmpty())

return false;

volunteerRepository.deleteById(id);

return true;

}

@Override

public List<Volunteer> findByEvent(Event event)

{

List<Volunteer> result = volunteerRepository.findByEvents(event);

log.info("IN findByEvent - volunteer: {} found by event: {}", result, event);

return result;

}

@Override

public List<Volunteer> findByFirstName(String volunteerFirstName)

{

List<Volunteer> result = volunteerRepository.findByFirstName(volunteerFirstName);

log.info("IN findByFirstName - volunteer: {} found by firstName: {}", result.size(), volunteerFirstName);

return result;

}

@Override

public List<Volunteer> findByLastName(String volunteerLastName)

{

List<Volunteer> result = volunteerRepository.findByLastName(volunteerLastName);

log.info("IN findByFirstName - volunteer: {} found by lastName: {}", result, volunteerLastName);

return result;

}

}

## Код клиента

App.xaml

<Application x:Class="VolunteerCenterDBClient.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Light.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Defaults.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Primary/MaterialDesignColor.DeepPurple.xaml" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/Themes/Recommended/Accent/MaterialDesignColor.Lime.xaml" />

<ResourceDictionary Source="DictionaryClientVC.xaml" />

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</Application>

App.xaml.cs

using System.Windows;

using VolunteerCenterDBClient.ViewModels;

using VolunteerCenterDBClient.Views;

namespace VolunteerCenterDBClient

{

/// <summary>

/// Interaction logic for App.xaml

/// </summary>

public partial class App : Application

{

public App()

{

(App.Current.MainWindow = new MainWindow { DataContext = new MainViewModel() }).Show();

}

}

}

MainWindow.xaml

<Window x:Class="VolunteerCenterDBClient.Views.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" SizeToContent="WidthAndHeight" WindowStartupLocation="CenterOwner"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:staticData="clr-namespace:VolunteerCenterDBClient.Models"

TextElement.Foreground="{DynamicResource MaterialDesignBody}"

TextElement.FontWeight="Regular"

TextElement.FontSize="13"

TextOptions.TextFormattingMode="Ideal"

TextOptions.TextRenderingMode="Auto"

Background="{DynamicResource MaterialDesignPaper}"

FontFamily="{DynamicResource MaterialDesignFont}">

<Grid x:Name="mainGrid">

<Grid.RowDefinitions>

<RowDefinition Height="55"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<DockPanel x:Name="\_menu" Visibility="Collapsed" Grid.Row="0" Margin="0,10" VerticalAlignment="Top">

<Button x:Name="main" Content="Мероприятия" DockPanel.Dock="Left"

Margin="15,0,0,0" Background="{DynamicResource buttonBackground}" BorderBrush="{DynamicResource buttonBackground}"/>

<Button x:Name="myEvents" Content="Мои мероприятия" DockPanel.Dock="Left"

Margin="15,0,0,0" Background="{DynamicResource buttonBackground}" BorderBrush="{DynamicResource buttonBackground}"/>

<Button x:Name="exit" Width="60" DockPanel.Dock="Right"

Margin="0,0,15,0" Background="{DynamicResource buttonBackground}" BorderBrush="{DynamicResource buttonBackground}">

<materialDesign:PackIcon Kind="Logout" HorizontalAlignment="Center" VerticalAlignment="Center" Height="28" Width="29" />

</Button>

<Button Name="profile" Width="60" DockPanel.Dock="Right" HorizontalAlignment="Right"

Margin="0,0,15.1,0" Background="{DynamicResource buttonBackground}" BorderBrush="{DynamicResource buttonBackground}">

<materialDesign:PackIcon Kind="Account" HorizontalAlignment="Center" VerticalAlignment="Center" Height="28" Width="37" BorderBrush="{DynamicResource buttonBackground}" Background="{DynamicResource buttonBackground}" />

</Button>

</DockPanel>

<Frame Grid.Row="0" Grid.RowSpan="2" x:Name="\_navigationFrame"

NavigationUIVisibility="Hidden" VerticalAlignment="Top" />

</Grid>

</Window>

MainWindow.xaml.cs

using System.Windows;

using System.Windows.Input;

using VolunteerCenterDBClient.Navigation;

namespace VolunteerCenterDBClient.Views

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

NavigationCommands.BrowseBack.InputGestures.Clear();

NavigationCommands.BrowseForward.InputGestures.Clear();

NavigationManager.NavigationService = \_navigationFrame.NavigationService;

}

}

}

AuthorizationPage.xaml

<Page x:Class="VolunteerCenterDBClient.Views.Pages.AuthorizationPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:ph="clr-namespace:VolunteerCenterDBClient.ViewModels"

mc:Ignorable="d"

Title="AuthorizationPage" Height="500" Width="350">

<Grid>

<TabControl TabStripPlacement="Top" HorizontalAlignment="Center" VerticalAlignment="Center">

<TabItem Width="173" Height="50" HorizontalAlignment="Left" Margin="-2,0,0,-0.4">

<TabItem.Header>

<TextBlock Text="Авторизация" Foreground="{DynamicResource buttonBackground}"/>

</TabItem.Header>

<Grid>

<materialDesign:Card

materialDesign:ShadowAssist.ShadowDepth="Depth4"

Padding="32" Margin="46,46,46.46,46" />

<Grid>

<TextBox

materialDesign:HintAssist.Hint="Имя пользователя"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

FontSize="18"

VerticalAlignment="Center" Margin="90,109,90,280.2" Height="54"

Text="{Binding Username, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" CaretBrush="Black"/>

<PasswordBox

x:Name="FloatingPasswordBox"

FontSize="18"

materialDesign:HintAssist.Hint="Пароль"

Style="{DynamicResource MaterialDesignFloatingHintPasswordBox}" Margin="90,192,90,197.2"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

ph:PasswordHelper.Attach ="True"

ph:PasswordHelper.Password="{Binding Password, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<Button

Style="{DynamicResource MaterialDesignRaisedDarkButton}"

Background="{DynamicResource buttonBackground}"

Content="Авторизация" Margin="90,300,90,109"

Command="{Binding AuthorizeCommand, UpdateSourceTrigger=PropertyChanged}" BorderBrush="{DynamicResource buttonBackground}"/>

</Grid>

</Grid>

</TabItem>

<TabItem Width="173" Height="50" HorizontalAlignment="Right" Margin="0,0,-3.4,-0.4">

<TabItem.Header>

<TextBlock Text="Регистрация" Foreground="{DynamicResource buttonBackground}"/>

</TabItem.Header>

<Grid>

<materialDesign:Card

materialDesign:ShadowAssist.ShadowDepth="Depth4"

Padding="32" Margin="16" />

<Grid>

<TextBox

materialDesign:HintAssist.Hint="Фамилия"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

FontSize="10"

VerticalAlignment="Center" Margin="89.9,44,89.9,360.2" Height="39"

Text="{Binding FirstName, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<TextBox

materialDesign:HintAssist.Hint="Имя"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

FontSize="10"

VerticalAlignment="Center" Margin="89.9,88,89.9,316.2" Height="39"

Text="{Binding LastName, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<TextBox

materialDesign:HintAssist.Hint="Отчество"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

FontSize="10"

VerticalAlignment="Center" Margin="89.9,132,89.9,272.2" Height="39"

Text="{Binding Patronymic, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<TextBox

materialDesign:HintAssist.Hint="Email"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

FontSize="10"

VerticalAlignment="Center" Margin="89.9,180,89.9,224.2" Height="39"

Text="{Binding Email, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<TextBox

materialDesign:HintAssist.Hint="Имя пользователя"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

FontSize="10"

VerticalAlignment="Center" Margin="89.9,224,89.9,180.2" Height="39"

Text="{Binding Username, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<PasswordBox

FontSize="10"

materialDesign:HintAssist.Hint="Пароль"

Style="{DynamicResource MaterialDesignFloatingHintPasswordBox}" Margin="89.9,268,89.9,136.2"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

ph:PasswordHelper.Attach ="True"

ph:PasswordHelper.Password="{Binding Password, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<PasswordBox

FontSize="10"

materialDesign:HintAssist.Hint="Подтверждение пароля"

Style="{DynamicResource MaterialDesignFloatingHintPasswordBox}" Margin="89.9,307,89.9,97.2"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

ph:PasswordHelper.Attach ="True"

ph:PasswordHelper.Password="{Binding PasswordConfirmation, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>

<Button

Style="{DynamicResource MaterialDesignRaisedDarkButton}"

Background="{DynamicResource buttonBackground}"

Content="Регистрация" Margin="89.9,364,89.9,47.2"

Command="{Binding RegisterCommand, UpdateSourceTrigger=PropertyChanged}" BorderBrush="{DynamicResource buttonBackground}"/>

</Grid>

</Grid>

</TabItem>

</TabControl>

</Grid>

</Page>

AuthorizationPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace VolunteerCenterDBClient.Views.Pages

{

/// <summary>

/// Interaction logic for AuthorizationPage.xaml

/// </summary>

public partial class AuthorizationPage : Page

{

public AuthorizationPage()

{

InitializeComponent();

}

}

}

AdminMainPage.xaml

<Page x:Class="VolunteerCenterDBClient.Views.Pages.AdminMainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VolunteerCenterDBClient.Views.Pages"

xmlns:materialDesign="http://schemas.microsoft.com/netfx/2009/xaml/presentation"

xmlns:materialDesign1="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:staticData="clr-namespace:VolunteerCenterDBClient.Models"

xmlns:app="clr-namespace:VolunteerCenterDBClient"

mc:Ignorable="d"

Title="AdminMainPage">

<Page.Resources>

<app:MultiValueToOneConverter x:Key="ToOneValueConverter"/>

</Page.Resources>

<StackPanel>

<Grid x:Name="gi" Height="30">

<Grid.RowDefinitions>

<RowDefinition Height="25"/>

<RowDefinition Height="5"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="150"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Row="0" Orientation="Horizontal" VerticalAlignment="Top" >

<Button Uid="0" x:Name="start" Width="150" Content="ВОЛОНТЕРЫ" VerticalAlignment="Center" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="{DynamicResource buttonBackground}" Click="Menu\_Click" Command="{Binding GetAllVolunteersCommand}"/>

<Button Uid="1" Width="150" Content="МЕРОПРИЯТИЯ" VerticalAlignment="Center" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="{DynamicResource buttonBackground}" Click="Menu\_Click" Command="{Binding GetAllEventsCommand}"/>

<Button Uid="2" Width="150" Content="ОРГАНИЗАТОРЫ" VerticalAlignment="Center" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="{DynamicResource buttonBackground}" Click="Menu\_Click" Command="{Binding GetAllOrgsCommand}"/>

<Button Uid="3" Width="150" Content="ПОЛЬЗОВАТЕЛИ" VerticalAlignment="Center" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="{DynamicResource buttonBackground}" Click="Menu\_Click" Command="{Binding GetAllUsersCommand}"/>

</StackPanel>

<Grid Grid.Row="1" x:Name="GridCursor" Width="150" Height="5" Background="{DynamicResource buttonBackground}" HorizontalAlignment="Left" Margin="-177,0,0,0"/>

<Button Grid.Column="1" Grid.RowSpan="2" Width="75" DockPanel.Dock="Right"

Command="{Binding GoBackCommand}"

Background="{DynamicResource buttonBackground}" BorderBrush="{DynamicResource buttonBackground}" Height="27" HorizontalAlignment="Center" VerticalAlignment="Center">

<materialDesign1:PackIcon Kind="Logout" HorizontalAlignment="Center" VerticalAlignment="Center" Height="25" Width="23" />

</Button>

</Grid>

<Grid Margin="0 5 0 0">

<Grid.RowDefinitions>

<RowDefinition Height="1\*"/>

<RowDefinition Height="14\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Orientation="Horizontal">

<Button

x:Name="addButtonVol"

Command="{Binding AddVolCommand}"

Visibility="Collapsed"

Background="{DynamicResource buttonBackground}"

Style="{DynamicResource MaterialDesignFloatingActionMiniAccentButton}"

Content="{materialDesign1:PackIcon Kind=Plus, Size=22}"

Margin="20,14,0,14" BorderBrush="#FF005464" />

<TextBox

x:Name="searchVol"

Visibility="Collapsed"

Style="{DynamicResource MaterialDesignOutlinedTextFieldTextBox}"

VerticalAlignment="Top"

Height="40"

TextWrapping="NoWrap"

materialDesign1:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign1:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalScrollBarVisibility="Disabled"

materialDesign1:HintAssist.Hint="Поиск по фамилии"

Width="150" Margin="20,14,0,14"/>

<Button

x:Name="findVolButton"

Visibility="Collapsed"

Command="{Binding FindByFirstNameVolCommand}"

CommandParameter="{Binding Text, ElementName=searchVol}"

Content="Поиск" BorderBrush="{DynamicResource buttonBackground}"

Background="{DynamicResource buttonBackground}"

Style="{DynamicResource MaterialDesignRaisedDarkButton}" Margin="20,14,0,14"/>

<Button

x:Name="addButtonEvent"

Command="{Binding AddEventCommand}"

Visibility="Collapsed"

Background="{DynamicResource buttonBackground}"

Style="{DynamicResource MaterialDesignFloatingActionMiniAccentButton}"

Content="{materialDesign1:PackIcon Kind=Plus, Size=22}"

Margin="20,14,0,14" BorderBrush="#FF005464" />

<TextBox

x:Name="searchEvent"

Visibility="Collapsed"

Style="{DynamicResource MaterialDesignOutlinedTextFieldTextBox}"

VerticalAlignment="Top"

Height="40"

TextWrapping="NoWrap"

materialDesign1:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign1:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalScrollBarVisibility="Disabled"

materialDesign1:HintAssist.Hint="Поиск по названию"

Width="150" Margin="20,14,0,14"/>

<Button

x:Name="findEventButton"

Visibility="Collapsed"

Command="{Binding FindByEventNameCommand}"

CommandParameter="{Binding Text, ElementName=searchEvent}"

Content="Поиск" BorderBrush="{DynamicResource buttonBackground}"

Background="{DynamicResource buttonBackground}"

Style="{DynamicResource MaterialDesignRaisedDarkButton}" Margin="20,14,0,14"/>

<Button

x:Name="addButtonOrg"

Command="{Binding AddOrgCommand}"

Visibility="Collapsed"

Background="{DynamicResource buttonBackground}"

Style="{DynamicResource MaterialDesignFloatingActionMiniAccentButton}"

Content="{materialDesign1:PackIcon Kind=Plus, Size=22}"

Margin="20,14,0,14" BorderBrush="#FF005464" />

<TextBox

x:Name="searchOrg"

Visibility="Collapsed"

Style="{DynamicResource MaterialDesignOutlinedTextFieldTextBox}"

VerticalAlignment="Top"

Height="40"

TextWrapping="NoWrap"

materialDesign1:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign1:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalScrollBarVisibility="Disabled"

materialDesign1:HintAssist.Hint="Поиск по названию"

Width="150" Margin="20,14,0,14"/>

<Button

x:Name="findOrgButton"

Visibility="Collapsed"

Command="{Binding FindByOrgNameCommand}"

CommandParameter="{Binding Text, ElementName=searchOrg}"

Content="Поиск" BorderBrush="{DynamicResource buttonBackground}"

Background="{DynamicResource buttonBackground}"

Style="{DynamicResource MaterialDesignRaisedDarkButton}" Margin="20,14,0,14"/>

</StackPanel>

<DataGrid Grid.Row="1"

x:Name="volsGrid"

Visibility="Hidden"

ItemsSource="{Binding ItemsSourceVols}"

SelectedItem="{Binding SelectedVol}"

CanUserAddRows="False"

AutoGenerateColumns="False"

HeadersVisibility="All"

IsReadOnly="True">

<DataGrid.InputBindings>

<MouseBinding

MouseAction="LeftDoubleClick"

Command="{Binding SelectVolCommand}"

CommandParameter="{Binding SelectedVol}"/>

</DataGrid.InputBindings>

<DataGrid.Columns>

<DataGridTextColumn

Binding="{Binding firstName}"

Header="Фамилия"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}">

</DataGridTextColumn>

<DataGridTextColumn

Binding="{Binding lastName}"

Header="Имя"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding patronymic}"

Header="Отчество"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding DateOfBirth}"

Header="Дата рождения"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding loyalty}"

Header="Уровень лояльности"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

</DataGrid.Columns>

</DataGrid>

<DataGrid Grid.Row="1"

x:Name="eventsGrid"

ItemsSource="{Binding ItemsSourceEvents}"

SelectedItem="{Binding SelectedEvent}"

CanUserAddRows="False"

AutoGenerateColumns="False"

HeadersVisibility="All"

IsReadOnly="True"

Visibility="Hidden">

<DataGrid.InputBindings>

<MouseBinding

MouseAction="LeftDoubleClick"

Command="{Binding SelectEventCommand}"

CommandParameter="{Binding SelectedEvent}"/>

</DataGrid.InputBindings>

<DataGrid.Columns>

<DataGridTextColumn

Binding="{Binding name}"

Header="Название"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding shortName}"

Header="Короткое название"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding dateStart}"

Header="Дата начала"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding dateEnd}"

Header="Дата конца"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding orgName}"

Header="Организатор"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

</DataGrid.Columns>

</DataGrid>

<DataGrid Grid.Row="1"

x:Name="orgsGrid"

ItemsSource="{Binding ItemsSourceOrgs}"

SelectedItem="{Binding SelectedOrg}"

CanUserAddRows="False"

AutoGenerateColumns="False"

HeadersVisibility="All"

IsReadOnly="True"

Visibility="Hidden">

<DataGrid.InputBindings>

<MouseBinding

MouseAction="LeftDoubleClick"

Command="{Binding SelectOrgCommand}"

CommandParameter="{Binding SelectedOrg}"/>

</DataGrid.InputBindings>

<DataGrid.Columns>

<DataGridTextColumn

Binding="{Binding name}"

Header="Название"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding contactPerson}"

Header="Контактное лицо"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding telephone}"

Header="Телефон"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

Binding="{Binding email}"

Header="Email"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

</DataGrid.Columns>

</DataGrid>

<DataGrid Grid.RowSpan="2"

x:Name="usersGrid" IsEnabled="True"

ItemsSource="{Binding ItemsSourceUsers}"

SelectedItem="{Binding SelectedUser}"

IsSynchronizedWithCurrentItem="True"

CanUserAddRows="False"

AutoGenerateColumns="False"

HeadersVisibility="All"

Visibility="Collapsed">

<DataGrid.Columns>

<DataGridTextColumn

IsReadOnly="True"

Binding="{Binding username}"

Header="Username"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

IsReadOnly="True"

Binding="{Binding email}"

Header="Email"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

IsReadOnly="True"

Binding="{Binding firstName}"

Header="Фамилия"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

IsReadOnly="True"

Binding="{Binding lastName}"

Header="Имя"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTextColumn

IsReadOnly="True"

Binding="{Binding patronymic}"

Header="Отчество"

ElementStyle="{DynamicResource MaterialDesignDataGridTextColumnStyle}"/>

<DataGridTemplateColumn Header="Роль">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Role}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<ComboBox IsEditable="True"

SelectedItem="{Binding DataContext.SelectedRole,

RelativeSource={RelativeSource AncestorType=DataGrid}}"

IsSynchronizedWithCurrentItem="True">

<TextBlock>ROLE\_USER</TextBlock>

<TextBlock>ROLE\_ADMIN</TextBlock>

</ComboBox>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="saveBT" Visibility="Hidden"

Content="Сохранить" Background="{DynamicResource buttonBackground}"

BorderBrush="{DynamicResource buttonBackground}"

Command="{Binding DataContext.SelectUserSaveCommand,

RelativeSource={RelativeSource AncestorType=DataGrid}}"/>

<DataTemplate.Triggers>

<DataTrigger Binding="{Binding IsSelected,

RelativeSource={RelativeSource FindAncestor,

AncestorType={x:Type DataGridRow}}}" Value="True">

<Setter TargetName="saveBT" Property="Visibility" Value="Visible"/>

</DataTrigger>

</DataTemplate.Triggers>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

</StackPanel>

</Page>

UserMainPage.xaml

<Page x:Class="VolunteerCenterDBClient.Views.Pages.UserMainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VolunteerCenterDBClient.Views.Pages"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:main="clr-namespace:VolunteerCenterDBClient.Views"

mc:Ignorable="d"

DataContextChanged="Page\_DataContextChanged" Height="600">

<Grid x:Name="grid">

<ScrollViewer VerticalScrollBarVisibility="Visible">

<StackPanel>

<ListBox x:Name="listBox" ItemsSource="{Binding ItemsSourceEvents}">

<ListBox.ItemContainerStyle>

<Style TargetType="ListBoxItem">

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="IsSelected" Value="True"/>

</Trigger>

</Style.Triggers>

</Style>

</ListBox.ItemContainerStyle>

<ListBox.ItemTemplate>

<DataTemplate>

<materialDesign:Card Margin="0,10" Height="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="7.5\*" />

<ColumnDefinition Width="2.5\*" />

</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0"

Text="{Binding name}"

Margin="15,10,0,0.5" FontSize="40"

HorizontalAlignment="Left" Width="546"/>

<Button Grid.Column="1"

Command="{Binding DataContext.ChooseEventCommand,

RelativeSource={RelativeSource AncestorType=ListBox}}"

CommandParameter="{Binding SelectedIndex, ElementName=listBox}"

Content="Подать заявку" Height="Auto"

BorderBrush="{DynamicResource buttonBackground}"

Background="{DynamicResource buttonBackground}"

Margin="15,29,16,30" Grid.RowSpan="2"/>

<DockPanel Grid.Row="1" Grid.ColumnSpan="2">

<TextBlock

Text="Организатор:" Margin="15,5,2,5.5"/>

<TextBlock

Text="{Binding org.name}" Margin="0,5,25,5.5" />

<TextBlock

Text="Дата проведения:" Margin="0,5,2,5.5" />

<TextBlock

Text="{Binding dateStart}" Margin="0,5,25,5.5" />

</DockPanel>

</Grid>

</materialDesign:Card>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<ListBox x:Name="listBoxUserRequest" Visibility="Collapsed"

ItemsSource="{Binding ItemsSourceRequestedEvents}">

<ListBox.ItemTemplate>

<DataTemplate>

<materialDesign:Card Margin="0,10" Width="761" Height="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="7.5\*" />

<ColumnDefinition Width="2.5\*" />

</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0"

Text="{Binding name}"

Margin="15,10,0,0.5" FontSize="40"

HorizontalAlignment="Left" Width="546"/>

<Button Grid.Column="1" IsEnabled="False"

Content="Заявка уже подана" Height="Auto"

BorderBrush="{DynamicResource buttonBackground}"

Background="{DynamicResource buttonBackground}"

Margin="15,29,16,30" Grid.RowSpan="2"/>

<DockPanel Grid.Row="1" Grid.ColumnSpan="2">

<TextBlock

Text="Организатор:" Margin="15,5,2,5.5"/>

<TextBlock

Text="{Binding org.name}" Margin="0,5,25,5.5" />

<TextBlock

Text="Дата проведения:" Margin="0,5,2,5.5" />

<TextBlock

Text="{Binding dateStart}" Margin="0,5,25,5.5" />

</DockPanel>

</Grid>

</materialDesign:Card>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<ListBox x:Name="listBoxUserEvents" Visibility="Collapsed"

ItemsSource="{Binding ItemsSourceUserEvents}">

<ListBox.ItemTemplate>

<DataTemplate>

<materialDesign:Card Margin="0,10" Width="761" Height="100">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="7.5\*" />

<ColumnDefinition Width="2.5\*" />

</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0"

Text="{Binding name}"

Margin="15,10,0,0.5" FontSize="40"

HorizontalAlignment="Left" Width="546"/>

<Button Grid.Column="1" IsEnabled="False"

Content="Успешная запись!"

Height="Auto"

BorderBrush="{DynamicResource buttonBackground}"

Background="{DynamicResource buttonBackground}"

Margin="15,29,16,30" Grid.RowSpan="2"/>

<DockPanel Grid.Row="1" Grid.ColumnSpan="2">

<TextBlock

Text="Организатор:" Margin="15,5,2,5.5"/>

<TextBlock

Text="{Binding org.name}" Margin="0,5,25,5.5" />

<TextBlock

Text="Дата проведения:" Margin="0,5,2,5.5" />

<TextBlock

Text="{Binding dateStart}" Margin="0,5,25,5.5" />

</DockPanel>

</Grid>

</materialDesign:Card>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</StackPanel>

</ScrollViewer>

<Grid Visibility="Collapsed"

x:Name="profileGrid" Margin="250,60">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBox

materialDesign:HintAssist.Hint="Фамилия"

x:Name="firstName"

Text="{Binding User.volunteer.firstName}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalAlignment="Center" Grid.Row="0" Grid.Column="0" Width="140" />

<TextBox

materialDesign:HintAssist.Hint="Имя"

x:Name="lastName"

Text="{Binding User.volunteer.lastName}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalAlignment="Center" Grid.Row="0" Grid.Column="1" Width="140" />

<TextBox

materialDesign:HintAssist.Hint="Отчество"

x:Name="patronymic"

Text="{Binding User.volunteer.patronymic}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalAlignment="Center" Grid.Row="1" Grid.Column="0" Width="140" />

<DatePicker

x:Name="dateOfBirth"

SelectedDate="{Binding User.volunteer.dateOfBirth}"

materialDesign:HintAssist.Hint="Дата рождения"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

BorderBrush="{DynamicResource buttonBackground}"

Style="{StaticResource MaterialDesignFloatingHintDatePicker}" Height="43" Grid.Row="1" Grid.Column="1" Margin="180,20,180,20.2" />

<TextBox

materialDesign:HintAssist.Hint="Пол"

x:Name="sex"

Text="{Binding User.volunteer.sex}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalAlignment="Center" Grid.Row="2" Grid.Column="0" Width="140" />

<TextBox

materialDesign:HintAssist.Hint="Гражданство"

x:Name="citizenship"

Text="{Binding User.volunteer.citizenship}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalAlignment="Center" Grid.Row="2" Grid.Column="1" Width="140" />

<TextBox

materialDesign:HintAssist.Hint="Институт"

x:Name="institute"

Text="{Binding User.volunteer.institute}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalAlignment="Center" Grid.Row="3" Grid.Column="0" Width="140" />

<TextBox

materialDesign:HintAssist.Hint="Группа"

x:Name="studyGroup"

Text="{Binding User.volunteer.studyGroup}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

VerticalAlignment="Center" Grid.Row="3" Grid.Column="1" Width="140" />

<TextBox

materialDesign:HintAssist.Hint="Год окончания"

x:Name="yearOfGraduation"

Text="{Binding User.volunteer.yearOfGraduation}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalAlignment="Center" Grid.Row="4" Grid.Column="0" Width="140" />

<TextBox

materialDesign:HintAssist.Hint="Телефон"

x:Name="telephone"

Text="{Binding User.volunteer.telephone}"

Style="{DynamicResource MaterialDesignFloatingHintTextBox}"

materialDesign:HintAssist.Foreground="{DynamicResource buttonBackground}"

materialDesign:TextFieldAssist.UnderlineBrush="{DynamicResource buttonBackground}"

VerticalAlignment="Center" Grid.Row="4" Grid.Column="1" Width="140" />

<Button

x:Name="saveButton"

Style="{DynamicResource MaterialDesignRaisedDarkButton}"

Command="{Binding ProfileUpdateCommand}"

Content="Сохранить" Grid.Row="5" Grid.ColumnSpan="2" Margin="400,19.8,400,20.2" Background="{DynamicResource buttonBackground}" BorderBrush="{DynamicResource buttonBackground}" />

</Grid>

</Grid>

</Page>

NavigationManager.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Navigation;

using VolunteerCenterDBClient.ViewModels;

namespace VolunteerCenterDBClient.Navigation

{

internal static class NavigationManager

{

private static NavigationService \_navigationService;

public static NavigationService NavigationService

{

private get =>

\_navigationService;

set

{

if (NavigationService is object)

{

throw new ApplicationException("NavigationService instance is already initialized!");

}

\_navigationService = value;

}

}

public static bool Navigate(FrameworkElement to)

{

try

{

NavigationService?.Navigate(to);

return true;

}

catch

{

return false;

}

}

public static bool Navigate<TView>(BaseViewModel viewModel)

where TView : FrameworkElement, new()

{

try

{

NavigationService?.Navigate(new TView { DataContext = viewModel });

return true;

}

catch

{

return false;

}

}

public static bool CanGoBack =>

NavigationService?.CanGoBack ?? false;

public static bool GoBack()

{

try

{

if (!CanGoBack)

{

return false;

}

NavigationService?.GoBack();

return true;

}

catch

{

return false;

}

}

}

}

RelayCommand.cs

using System;

using System.Windows.Input;

namespace VolunteerCenterDBClient.Commands

{

public sealed class RelayCommand : ICommand

{

private readonly Action<object> \_execute;

private readonly Func<object, bool> \_canExecute;

public event EventHandler CanExecuteChanged

{

add

{

CommandManager.RequerySuggested += value;

}

remove

{

CommandManager.RequerySuggested -= value;

}

}

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return \_canExecute == null || \_canExecute(parameter);

}

public void Execute(object parameter)

{

\_execute(parameter);

}

}

}

BaseViewModel.cs

using System.ComponentModel;

namespace VolunteerCenterDBClient.ViewModels

{

internal class BaseViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

protected void OnPropertiesChanged(params string[] propertiesNames)

{

foreach (var propertyName in propertiesNames)

{

OnPropertyChanged(propertyName);

}

}

}

}

MainViewModel.cs

using VolunteerCenterDBClient.Navigation;

using VolunteerCenterDBClient.ViewModels.Pages;

using VolunteerCenterDBClient.Views.Pages;

namespace VolunteerCenterDBClient.ViewModels

{

internal sealed class MainViewModel : BaseViewModel

{

public MainViewModel()

{

NavigationManager.Navigate<AuthorizationPage>(new AuthorizationPageViewModel());

}

}

}

AdminPageViewModel.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Windows.Controls;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Text.Json;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using VolunteerCenterDBClient.Commands;

using VolunteerCenterDBClient.Navigation;

using VolunteerCenterDBClient.Models;

using VolunteerCenterDBClient.Views;

namespace VolunteerCenterDBClient.ViewModels.Pages

{

class AdminPageViewModel : BaseViewModel

{

private string \_token;

private Volunteer \_selectedVol;

public Volunteer SelectedVol

{

get =>

\_selectedVol;

set

{

\_selectedVol = value;

OnPropertyChanged(nameof(SelectedVol));

}

}

private Event \_selectedEvent;

public Event SelectedEvent

{

get =>

\_selectedEvent;

set

{

\_selectedEvent = value;

OnPropertyChanged(nameof(SelectedEvent));

}

}

private Organization \_selectedOrg;

public Organization SelectedOrg

{

get =>

\_selectedOrg;

set

{

\_selectedOrg = value;

OnPropertyChanged(nameof(SelectedOrg));

}

}

private bool \_lastUserModified;

private User \_selectedUser;

public User SelectedUser

{

get =>

\_selectedUser;

set

{

if (\_selectedUser != null && !\_lastUserModified)

{

\_selectedUser.role = new Role(\_oldUserRole.Equals("ROLE\_USER") ? 2 : 1, \_oldUserRole.ToString());

OnPropertiesChanged(nameof(SelectedUser));

}

\_lastUserModified = false;

\_selectedUser = value;

OnPropertyChanged(nameof(SelectedUser));

}

}

private string \_oldUserRole = string.Empty;

private string \_selectedRole = string.Empty;

public Object SelectedRole

{

get =>

\_selectedRole;

set

{

\_oldUserRole = SelectedUser.Role;

\_selectedRole = ((TextBlock)value).Text;

SelectedUser.role = new Role(\_selectedRole.Equals("ROLE\_USER") ? 2 : 1, \_selectedRole.ToString());

OnPropertyChanged(nameof(SelectedRole));

}

}

public AdminPageViewModel(string token) { \_token = token; GetAllVolunteersAsync(); }

#region Observes

private ObservableCollection<Volunteer> \_volunteers;

private ObservableCollection<Event> \_events;

private ObservableCollection<Organization> \_orgs;

private ObservableCollection<User> \_users;

public IEnumerable<string> UserRoles => new[] { "ROLE\_USER", "ROLE\_ADMIN" };

public ObservableCollection<Volunteer> ItemsSourceVols

{

get =>

\_volunteers;

set

{

\_volunteers = value;

OnPropertyChanged(nameof(ItemsSourceVols));

}

}

public ObservableCollection<Event> ItemsSourceEvents

{

get =>

\_events;

set

{

\_events = value;

OnPropertyChanged(nameof(ItemsSourceEvents));

}

}

public ObservableCollection<Organization> ItemsSourceOrgs

{

get =>

\_orgs;

set

{

\_orgs = value;

OnPropertyChanged(nameof(ItemsSourceOrgs));

}

}

public ObservableCollection<User> ItemsSourceUsers

{

get =>

\_users;

set

{

\_users = value;

OnPropertyChanged(nameof(ItemsSourceUsers));

}

}

#endregion

#region Commands

public ICommand \_getAllVolunteersCommand;

public ICommand \_getAllEventsCommand;

public ICommand \_getAllOrgsCommand;

public ICommand \_getAllUsersCommand;

public ICommand \_addVolCommand;

public ICommand \_addEvenCommandt;

public ICommand \_addOrgCommand;

public ICommand \_selectVolCommand;

public ICommand \_selectEventCommand;

public ICommand \_selectOrgCommand;

public ICommand \_selectUserSaveCommand;

public ICommand \_findByFirstNameVolCommand;

public ICommand \_findByEventNameCommand;

public ICommand \_findByOrgNameCommand;

private ICommand \_goBackCommand;

public ICommand GetAllVolunteersCommand =>

\_getAllVolunteersCommand ?? (\_getAllVolunteersCommand = new RelayCommand(async \_ => await GetAllVolunteersAsync()));

public ICommand GetAllEventsCommand =>

\_getAllEventsCommand ?? (\_getAllEventsCommand = new RelayCommand(async \_ => await GetAllEventsAsync()));

public ICommand GetAllOrgsCommand =>

\_getAllOrgsCommand ?? (\_getAllOrgsCommand = new RelayCommand(async \_ => await GetAllOrgsAsync()));

public ICommand GetAllUsersCommand =>

\_getAllUsersCommand ?? (\_getAllUsersCommand = new RelayCommand(async \_ => await GetAllUsersAsync()));

public ICommand AddVolCommand =>

\_addVolCommand ?? (\_addVolCommand = new RelayCommand(async \_ => await AddVolunteerAsync()));

public ICommand AddEventCommand =>

\_addEvenCommandt ?? (\_addEvenCommandt = new RelayCommand(async \_ => await AddEventAsync()));

public ICommand AddOrgCommand =>

\_addOrgCommand ?? (\_addOrgCommand = new RelayCommand(async \_ => await AddOrgAsync()));

public ICommand SelectVolCommand =>

\_selectVolCommand ?? (\_selectVolCommand = new RelayCommand(async obj => await SelectVolAsync(obj)));

public ICommand SelectEventCommand =>

\_selectEventCommand ?? (\_selectEventCommand = new RelayCommand(async obj => await SelectEventAsync(obj)));

public ICommand SelectOrgCommand =>

\_selectOrgCommand ?? (\_selectOrgCommand = new RelayCommand(async obj => await SelectOrgAsync(obj)));

public ICommand SelectUserSaveCommand =>

\_selectUserSaveCommand ?? (\_selectUserSaveCommand = new RelayCommand(async \_ => await SelectUserSaveAsync()));

public ICommand FindByFirstNameVolCommand =>

\_findByFirstNameVolCommand ?? (\_findByFirstNameVolCommand = new RelayCommand(async obj => await FindByFirstNameVolunteer(obj)));

public ICommand FindByEventNameCommand =>

\_findByEventNameCommand ?? (\_findByEventNameCommand = new RelayCommand(async obj => await FindByEventName(obj)));

public ICommand FindByOrgNameCommand =>

\_findByOrgNameCommand ?? (\_findByOrgNameCommand = new RelayCommand(async obj => await FindByOrgName(obj)));

public ICommand GoBackCommand =>

\_goBackCommand ?? (\_goBackCommand = new RelayCommand(\_ => NavigationManager.GoBack()));

#endregion

#region Tasks(HttpRequests)

#region Volunteers

private async Task GetAllVolunteersAsync()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/volunteers/get\_views");

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

string result;

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

result = await streamReader.ReadToEndAsync();

ItemsSourceVols = JsonSerializer.Deserialize<ObservableCollection<Volunteer>>(result);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task FindByFirstNameVolunteer(Object obj)

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/volunteers/find?first\_name=" + (string)obj);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

var list = JsonSerializer.Deserialize<ObservableCollection<Volunteer>>(result);

if (list.Count == 0)

MessageBox.Show("Волонтера с фамилией \"" + (string)obj + "\" нет");

else

ItemsSourceVols = list;

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task<ObservableCollection<Volunteer>> GetVolunteersByEventAsync(long id)

{

string address = "http://localhost:8080/api/admin/volunteers/get\_by\_event?event\_id=" + id;

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(address);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

return JsonSerializer.Deserialize<ObservableCollection<Volunteer>>(result);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

return null;

}

private async Task AddVolunteerAsync()

{

VolunteerWindow addWindow = new VolunteerWindow();

if (addWindow.ShowDialog() == true)

{

await AddVolunteerAsync(addWindow.Volunteer);

}

else

{

MessageBox.Show("Не удалось добавить волонтера!");

}

}

private async Task AddVolunteerAsync(Volunteer volunteer)

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/volunteers/adding");

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

string json = JsonSerializer.Serialize<Volunteer>(volunteer);

await streamWriter.WriteAsync(json);

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

if ((await streamReader.ReadToEndAsync()).Equals("OK"))

{

MessageBox.Show("Волонтер успешно добавлен!");

await GetAllVolunteersAsync();

}

else

{

MessageBox.Show("Волонтер не добавлен!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task SelectVolAsync(Object obj)

{

VolunteerWindow volWindow = new VolunteerWindow(obj as Volunteer);

bool? result = volWindow.ShowDialog();

if (result == true)

{

await AddVolunteerAsync(volWindow.Volunteer);

}

else if (result == false && volWindow.IsDeleting)

{

await SelectedVolunteerDeleteAsync((obj as Volunteer).id);

}

}

private async Task SelectedVolunteerDeleteAsync(long id)

{

string address = "http://localhost:8080/api/admin/volunteers/delete/{" + id + "}";

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(address);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "POST";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

byte[] dataStream = Encoding.UTF8.GetBytes("{" + id + "}");

httpWebRequest.ContentLength = dataStream.Length;

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

await streamWriter.WriteAsync("{" + id + "}");

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

if (result.Equals("\"OK\""))

{

MessageBox.Show("Волонтер успешно удален!");

await GetAllVolunteersAsync();

}

else

{

MessageBox.Show("Ошибка при удалении волонтера!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

#endregion

#region Events

private async Task GetAllEventsAsync()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/events/get");

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

string result;

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

result = await streamReader.ReadToEndAsync();

ItemsSourceEvents = JsonSerializer.Deserialize<ObservableCollection<Event>>(result);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task AddEventAsync()

{

await GetAllOrgsAsync();

List<string> orgs = new List<string>(ItemsSourceOrgs.Count);

foreach (Organization org in ItemsSourceOrgs)

orgs.Add(org.name);

EventWindow addWindow = new EventWindow(orgs);

if (addWindow.ShowDialog() == true)

{

await AddEventAsync(addWindow.Event);

}

else

{

MessageBox.Show("Не удалось добавить мероприятие!");

}

}

private async Task FindByEventName(Object obj)

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/events/find?name=" + (string)obj);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

var list = JsonSerializer.Deserialize<ObservableCollection<Event>>(result);

if (list.Count == 0)

MessageBox.Show("Мероприятия/й " + (string)obj + " нет");

else

ItemsSourceEvents = list;

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task AddEventAsync(Event @event)

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/events/adding");

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

string json = JsonSerializer.Serialize<Event>(@event);

await streamWriter.WriteAsync(json);

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

if ((await streamReader.ReadToEndAsync()).Equals("OK"))

{

MessageBox.Show("Мероприятие успешно добавлено!");

await GetAllEventsAsync();

}

else

{

MessageBox.Show("Мероприятие не добавлено!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task SelectEventAsync(Object obj)

{

await GetAllOrgsAsync();

List<string> orgs = new List<string>(ItemsSourceOrgs.Count);

foreach (Organization org in ItemsSourceOrgs)

orgs.Add(org.name);

EventWindow evtWindow = new EventWindow(obj as Event, orgs, new ObservableCollection<Volunteer>((obj as Event).volunteers),

new ObservableCollection<Volunteer>((obj as Event).requestedVols));

bool? result = evtWindow.ShowDialog();

if (result == true)

{

Event @event = evtWindow.Event;

@event.org = ItemsSourceOrgs.First<Organization>(\_ => \_.name.Equals(evtWindow.Event.orgName));

await AddEventAsync(@event);

}

else if (result == false && evtWindow.IsDeleting)

{

await SelectedEventDeleteAsync((obj as Event).id);

}

}

private async Task SelectedEventDeleteAsync(long id)

{

string address = "http://localhost:8080/api/admin/events/delete/{" + id + "}";

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(address);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "POST";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

byte[] dataStream = Encoding.UTF8.GetBytes("{" + id + "}");

httpWebRequest.ContentLength = dataStream.Length;

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

await streamWriter.WriteAsync("{" + id + "}");

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

String result = await streamReader.ReadToEndAsync();

if (result.Equals("\"OK\""))

{

MessageBox.Show("Мероприятие успешно удалено!");

await GetAllVolunteersAsync();

}

else

{

MessageBox.Show("Ошибка при удалении мероприятия!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task<ObservableCollection<Event>> FindEventsByOrgIdAsync(long id)

{

string address = "http://localhost:8080/api/admin/orgs/get\_events?id=" + id;

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(address);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

return JsonSerializer.Deserialize<ObservableCollection<Event>>(result);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

return null;

}

#endregion

#region Orgs

private async Task GetAllOrgsAsync()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/orgs/get");

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

string result;

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

result = await streamReader.ReadToEndAsync();

ItemsSourceOrgs = JsonSerializer.Deserialize<ObservableCollection<Organization>>(result);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task FindByOrgName(Object obj)

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/orgs/find?name=" + (string)obj);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

var list = JsonSerializer.Deserialize<ObservableCollection<Organization>>(result);

if (list.Count == 0)

MessageBox.Show("Организатора \"" + (string)obj + "\" нет");

else

ItemsSourceOrgs = list;

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task AddOrgAsync()

{

OrgWindow addWindow = new OrgWindow();

if (addWindow.ShowDialog() == true)

{

await AddOrgAsync(addWindow.Org);

}

else

{

MessageBox.Show("Не удалось добавить волонтера!");

}

}

private async Task AddOrgAsync(Organization org)

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/orgs/adding");

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

string json = JsonSerializer.Serialize<Organization>(org);

await streamWriter.WriteAsync(json);

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

if ((await streamReader.ReadToEndAsync()).Equals("OK"))

{

MessageBox.Show("Организатор успешно добавлен!");

await GetAllOrgsAsync();

}

else

{

MessageBox.Show("Организатор не добавлен!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task SelectOrgAsync(Object obj)

{

OrgWindow orgWindow = new OrgWindow(obj as Organization, (await FindEventsByOrgIdAsync((obj as Organization).id)));

bool? result = orgWindow.ShowDialog();

if (result == true)

{

await AddOrgAsync(orgWindow.Org);

}

else if (result == false && orgWindow.IsDeleting)

{

await SelectedOrgDeleteAsync((obj as Organization).id);

}

}

private async Task SelectedOrgDeleteAsync(long id)

{

string address = "http://localhost:8080/api/admin/orgs/delete/{" + id + "}";

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(address);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "POST";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

byte[] dataStream = Encoding.UTF8.GetBytes("{" + id + "}");

httpWebRequest.ContentLength = dataStream.Length;

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

await streamWriter.WriteAsync("{" + id + "}");

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

String result = await streamReader.ReadToEndAsync();

if (result.Equals("\"OK\""))

{

MessageBox.Show("Организатор успешно удален!");

await GetAllVolunteersAsync();

}

else

{

MessageBox.Show("Ошибка при удалении организатора!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

#endregion

#region Users

private async Task GetAllUsersAsync()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/admin/users/get");

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

ItemsSourceUsers = JsonSerializer.Deserialize<ObservableCollection<User>>(result);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task SelectUserSaveAsync()

{

string address = "http://localhost:8080/api/admin/users/set\_role?id=" + SelectedUser.id + "&role=" + SelectedUser.Role;

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(address);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "Post";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

byte[] dataStream = Encoding.UTF8.GetBytes("?id=" + SelectedUser.id + "&role=" + SelectedUser.Role);

httpWebRequest.ContentLength = dataStream.Length;

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

await streamWriter.WriteAsync("?id=" + SelectedUser.id + "&role=" + SelectedUser.Role);

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

String result = await streamReader.ReadToEndAsync();

if (result.Equals("OK"))

{

MessageBox.Show("Роль пользователя изменена!");

\_lastUserModified = true;

await GetAllVolunteersAsync();

}

else

{

MessageBox.Show("Ошибка при изменении роли пользователя!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

#endregion

#endregion

}

}

AuthorizationPageViewModel.cs

using System.Collections.Generic;

using System.IO;

using System.Net;

using System.Text.Json;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using VolunteerCenterDBClient.Commands;

using VolunteerCenterDBClient.Models;

using VolunteerCenterDBClient.Navigation;

using VolunteerCenterDBClient.Views.Pages;

namespace VolunteerCenterDBClient.ViewModels.Pages

{

internal sealed class AuthorizationPageViewModel : BaseViewModel

{

private string \_passwordConfirmation = string.Empty;

private ICommand \_authorizeCommand;

private ICommand \_registerCommand;

User user = new User(string.Empty, string.Empty, string.Empty, string.Empty, string.Empty, string.Empty);

public string Username

{

private get =>

user.username;

set

{

user.username = value;

OnPropertyChanged(nameof(Username));

}

}

public string Password

{

private get =>

user.password;

set

{

user.password = value;

OnPropertyChanged(nameof(Password));

}

}

public string FirstName

{

private get =>

user.firstName;

set

{

user.firstName = value;

OnPropertyChanged(nameof(FirstName));

}

}

public string LastName

{

private get =>

user.lastName;

set

{

user.lastName = value;

OnPropertyChanged(nameof(LastName));

}

}

public string Patronymic

{

private get =>

user.patronymic;

set

{

user.patronymic = value;

OnPropertyChanged(nameof(Patronymic));

}

}

public string Email

{

private get =>

user.email;

set

{

user.email = value;

OnPropertyChanged(nameof(Email));

}

}

public string PasswordConfirmation

{

get =>

\_passwordConfirmation;

set

{

\_passwordConfirmation = value;

OnPropertyChanged(nameof(PasswordConfirmation));

}

}

public ICommand AuthorizeCommand =>

\_authorizeCommand ?? (\_authorizeCommand = new RelayCommand(async \_ => await AuthorizeAsync(),

\_ => CanAuthorize()));

public ICommand RegisterCommand =>

\_registerCommand ?? (\_registerCommand = new RelayCommand(async \_ => await RegisterAsync(),

\_ => CanRegister()));

private async Task AuthorizeAsync()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/auth");

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

string json = JsonSerializer.Serialize(new

{

username = user.username,

password = user.password

});

await streamWriter.WriteAsync(json);

}

var httpResponse = (HttpWebResponse) await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

Dictionary<string, string> jsonToken = JsonSerializer.Deserialize<Dictionary<string, string>>(result);

if (jsonToken["role"] == "ROLE\_ADMIN")

NavigationManager.Navigate<AdminMainPage>(new AdminPageViewModel(jsonToken["token"]));

else

NavigationManager.Navigate<UserMainPage>(new UserMainPageViewModel(jsonToken["token"], long.Parse(jsonToken["id"])));

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task RegisterAsync()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/register");

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

string json = JsonSerializer.Serialize<User>(user);

await streamWriter.WriteAsync(json);

}

var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();

string result;

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

result = await streamReader.ReadToEndAsync();

if (result.Equals("OK"))

{

await FindVolunteerForUser();

MessageBox.Show("Вы были зарегистрированы, пожалуйста, авторизуйтесь!");

}

else

MessageBox.Show("Такой пользователь уже существует!");

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

}

private async Task FindVolunteerForUser()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/auth");

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

string json = JsonSerializer.Serialize(new

{

username = user.username,

password = user.password

});

await streamWriter.WriteAsync(json);

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

Dictionary<string, string> jsonToken = JsonSerializer.Deserialize<Dictionary<string, string>>(result);

HttpWebRequest httpWebRequest1 = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/user/volunteer/set?id="

+ jsonToken["id"]);

httpWebRequest1.ContentType = "application/json";

httpWebRequest1.Method = "POST";

httpWebRequest1.Headers.Add("Authorization", "Bearer\_" + jsonToken["token"]);

var result1 = (HttpWebResponse)await httpWebRequest1.GetResponseAsync();

}

}

private bool CanAuthorize()

{

return Username.Length > 0 && Password.Length > 0;

}

private bool CanRegister()

{

return !string.IsNullOrEmpty(FirstName)

&& !string.IsNullOrEmpty(LastName)

&& !string.IsNullOrEmpty(Patronymic)

&& !string.IsNullOrEmpty(Email)

&& !string.IsNullOrEmpty(Username)

&& !string.IsNullOrEmpty(Password)

&& Password.Equals(PasswordConfirmation);

}

}

}

UserMainPageViewModel.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

using System.Net;

using System.Text.Json;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using VolunteerCenterDBClient.Commands;

using VolunteerCenterDBClient.Models;

using VolunteerCenterDBClient.Navigation;

namespace VolunteerCenterDBClient.ViewModels.Pages

{

class UserMainPageViewModel : BaseViewModel

{

private string \_token;

private long \_id;

public ObservableCollection<Event> \_events;

public ObservableCollection<Event> ItemsSourceEvents

{

get =>

\_events;

set

{

\_events = value;

OnPropertyChanged(nameof(ItemsSourceEvents));

}

}

public ObservableCollection<Event> \_userEvents;

public ObservableCollection<Event> ItemsSourceUserEvents

{

get =>

\_userEvents;

set

{

\_userEvents = value;

OnPropertyChanged(nameof(ItemsSourceUserEvents));

}

}

public ObservableCollection<Event> \_requestedEvents;

public ObservableCollection<Event> ItemsSourceRequestedEvents

{

get =>

\_requestedEvents;

set

{

\_requestedEvents = value;

OnPropertyChanged(nameof(ItemsSourceRequestedEvents));

}

}

public User \_user;

public User User

{

get =>

\_user;

set

{

\_user = value;

OnPropertyChanged(nameof(User));

}

}

private int \_selected;

public int Selected

{

get =>

\_selected;

set

{

\_selected = value;

OnPropertyChanged(nameof(Selected));

}

}

public UserMainPageViewModel(string token, long id) { \_token = token; \_id = id; AllEventsCommand.Execute(null); }

#region Commands

private ICommand \_allEventsCommand;

private ICommand \_myEventsCommand;

private ICommand \_profileCommand;

private ICommand \_profileUpdateCommand;

private ICommand \_goBackCommand;

private ICommand \_chooseEventCommand;

public ICommand AllEventsCommand =>

\_allEventsCommand ?? (\_allEventsCommand = new RelayCommand(async \_ => await GetAllEvents()));

public ICommand MyEventsCommand =>

\_myEventsCommand ?? (\_myEventsCommand = new RelayCommand(async \_ => await GetUserEvents()));

public ICommand ProfileCommand =>

\_profileCommand ?? (\_profileCommand = new RelayCommand(async \_ => await GetProfile()));

public ICommand ProfileUpdateCommand =>

\_profileUpdateCommand ?? (\_profileUpdateCommand = new RelayCommand(async \_ => await UpdateProfile()));

public ICommand GoBackCommand =>

\_goBackCommand ?? (\_goBackCommand = new RelayCommand(\_ => NavigationManager.GoBack()));

public ICommand ChooseEventCommand =>

\_chooseEventCommand ?? (\_chooseEventCommand = new RelayCommand(async obj => await RequestEvent(obj)));

#endregion

#region Tasks

private async Task GetAllEvents()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/user/events/all?id=" + \_id);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

ItemsSourceEvents = JsonSerializer.Deserialize<ObservableCollection<Event>>(result);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

GoBackCommand.Execute(null);

}

}

private async Task GetUserEvents()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/user/events/my\_events?id=" + \_id);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

var response = JsonSerializer.Deserialize<Dictionary<string, List<Event>>>(result);

ItemsSourceUserEvents = new ObservableCollection<Event>(response["events"]);

ItemsSourceRequestedEvents = new ObservableCollection<Event>(response["requestedEvents"]);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

GoBackCommand.Execute(null);

}

}

private async Task GetProfile()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/user/profile?id=" + \_id);

httpWebRequest.ContentType = "application/x-www-form-urlencoded";

httpWebRequest.Method = "GET";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

string result = await streamReader.ReadToEndAsync();

User = JsonSerializer.Deserialize<User>(result);

}

}

catch (IOException)

{

MessageBox.Show("Не удалось прочитать данные с сервера!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

GoBackCommand.Execute(null);

}

}

private async Task UpdateProfile()

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/user/profile/update");

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))

{

User.firstName = User.volunteer.firstName;

User.lastName = User.volunteer.lastName;

User.patronymic = User.volunteer.patronymic;

string json = JsonSerializer.Serialize<User>(User);

await streamWriter.WriteAsync(json);

}

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

if ((await streamReader.ReadToEndAsync()).Equals("OK"))

{

MessageBox.Show("Данные успешно изменены!");

await GetProfile();

}

else

{

MessageBox.Show("Не получилось изменить данные!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

GoBackCommand.Execute(null);

}

}

private async Task RequestEvent(Object obj)

{

HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8080/api/user/events/request?user\_id="

+ \_id + "&event\_id=" + ItemsSourceEvents[(int)obj].id);

httpWebRequest.ContentType = "application/json";

httpWebRequest.Method = "POST";

httpWebRequest.Headers.Add("Authorization", "Bearer\_" + \_token);

try

{

var httpResponse = (HttpWebResponse)await httpWebRequest.GetResponseAsync();

using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))

{

if ((await streamReader.ReadToEndAsync()).Equals("OK"))

{

MessageBox.Show("Заявка на мероприятие \"" + ItemsSourceEvents[(int)obj].name + "\" подана!");

await GetAllEvents();

}

else

{

MessageBox.Show("Не получилось подать заявку!");

}

}

}

catch (IOException)

{

MessageBox.Show("Не удалось связаться с сервером!");

}

catch (WebException)

{

MessageBox.Show("Не удалось связаться с сервером!");

GoBackCommand.Execute(null);

}

}

#endregion

}

}

PasswordHelper.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

namespace VolunteerCenterDBClient.ViewModels

{

public static class PasswordHelper

{

public static readonly DependencyProperty PasswordProperty =

DependencyProperty.RegisterAttached("Password",

typeof(string), typeof(PasswordHelper),

new FrameworkPropertyMetadata(string.Empty, OnPasswordPropertyChanged));

public static readonly DependencyProperty AttachProperty =

DependencyProperty.RegisterAttached("Attach",

typeof(bool), typeof(PasswordHelper), new PropertyMetadata(false, Attach));

private static readonly DependencyProperty IsUpdatingProperty =

DependencyProperty.RegisterAttached("IsUpdating", typeof(bool),

typeof(PasswordHelper));

public static void SetAttach(DependencyObject dp, bool value)

{

dp.SetValue(AttachProperty, value);

}

public static bool GetAttach(DependencyObject dp)

{

return (bool)dp.GetValue(AttachProperty);

}

public static string GetPassword(DependencyObject dp)

{

return (string)dp.GetValue(PasswordProperty);

}

public static void SetPassword(DependencyObject dp, string value)

{

dp.SetValue(PasswordProperty, value);

}

private static bool GetIsUpdating(DependencyObject dp)

{

return (bool)dp.GetValue(IsUpdatingProperty);

}

private static void SetIsUpdating(DependencyObject dp, bool value)

{

dp.SetValue(IsUpdatingProperty, value);

}

//необязательно

private static void OnPasswordPropertyChanged(DependencyObject sender,

DependencyPropertyChangedEventArgs e)

{

PasswordBox passwordBox = sender as PasswordBox;

passwordBox.PasswordChanged -= PasswordChanged;

if (!(bool)GetIsUpdating(passwordBox))

{

passwordBox.Password = (string)e.NewValue;

}

passwordBox.PasswordChanged += PasswordChanged;

}

private static void Attach(DependencyObject sender,

DependencyPropertyChangedEventArgs e)

{

PasswordBox passwordBox = sender as PasswordBox;

if (passwordBox == null)

return;

if ((bool)e.OldValue)

{

passwordBox.PasswordChanged -= PasswordChanged;

}

if ((bool)e.NewValue)

{

passwordBox.PasswordChanged += PasswordChanged;

}

}

private static void PasswordChanged(object sender, RoutedEventArgs e)

{

PasswordBox passwordBox = sender as PasswordBox;

SetIsUpdating(passwordBox, true); //необязательно

SetPassword(passwordBox, passwordBox.Password);

SetIsUpdating(passwordBox, false); //необязательно

}

}

}