

BEACH VOLLEY - WEB APP

Contents

1.	Risorse:	4
2.	Requisiti di Sistema	5
3.	Implementazione.....	6
4.	PostgreSQL database – diagramma ER.....	7
5.	Java SPRING API.....	8
5.1	Java SPRING API – UML	8
5.2	Organizzazione del codice: funzioni principali	9
1.	Collegamento tra oggetti e DBS	9
2.	SecurityController.....	10
3.	SecurityConfiguration.....	10
4.	Request validation: UserDataFlow, Gamejoiner, Gamecreate	11
5.3	Testing	14
6.	JS React Web APP	17
6.1	Organizzazione del codice	17
Funzioni principali.....		17
1.	Actions: GET data from API	17
2.	Actions: Navigazione	21
3.	Components	22
6.2	Componenti.....	23
1.	Login and Header.....	23
2.	Register.....	23
3.	Main page (GameList e UserDetails)	24
4.	Create Game.....	25
5.	Game Details	25
6.	User Profile	26
6.3	User Testing - Casi di test e risultati	27
1.	Creazione nuovo utente (R1).....	27
2.	Errori creazione nuovo utente (R1).....	28
3.	Creazione nuova partita (R5).....	29
4.	Creazione nuova partita con errori (R5)	30
5.	Iscrizione a una partita (R6).....	31

6.	Cancellazione iscrizione a partita < 24 ore dall'evento (R7)	32
7.	Cancellazione iscrizione a partita > 24 ore dall'evento (R7)	32
8.	Cancellazione partita creata (R5)	33
9.	Cancellazione partita creata da un altro utente (R2, R5)	34
10.	Iscrizione a partita completa (R6).....	34
11.	Inserimento dati utente (R3)	35
6.4	User Testing – Riassunto	35

1. Risorse:

Archivio codice:

<https://github.com/dicaros/BVapp/>

Documentazione API:

<https://documenter.getpostman.com/view/10259016/SWTD9crr?version=latest>

2. Requisiti di Sistema

Funzionamento:

Il sito permette agli utenti registrati di creare un evento (partita di beach volley) che si svolge in una determinata data e ora in un determinato centro sportivo scelto dall'utente (sulla base di una prenotazione effettivamente fatta nel mondo reale al centro sportivo indicato).

Ogni utente può vedere la lista degli eventi creati e può decidere di partecipare iscrivendosi ad una partita. Gli utenti hanno la possibilità di iscriversi fino a quando il numero previsto di partecipanti è raggiunto e la partita viene confermata.

Il sito è composto quindi principalmente da utenti, eventi (partite) e centri sportivi in cui si svolgono le partite.

Requisiti principali:

- R1. Registrazione di un nuovo utente con credenziali (username e password)
- R2. Gestione della sessione utente e della sicurezza (ogni utente può modificare solo i propri dati e i propri eventi.)
- R3. **Gestione del proprio profilo utente (nome, cognome, telefono, numero partite giocate)**
- ~~R4. Aggiunta di centri sportivi (nome, indirizzo, telefono, sito web, posizione gps) da parte degli utenti~~ *requisito rimosso in quanto i centri sportivi devono essere verificati dall'amministratore del sistema prima di essere inseriti.*
- R5. Creazione e cancellazione delle partite (organizzatore, prezzo per persona, data, ora, indicatore partita completa, descrizione, lista utenti iscritti.)
- R6. Possibilità di iscrizione alle partite finché il numero massimo di utenti è raggiunto.
- R7. Possibilità di cancellarsi da una partita fino a ventiquattro ore dall'inizio dell'evento.

3. Implementazione

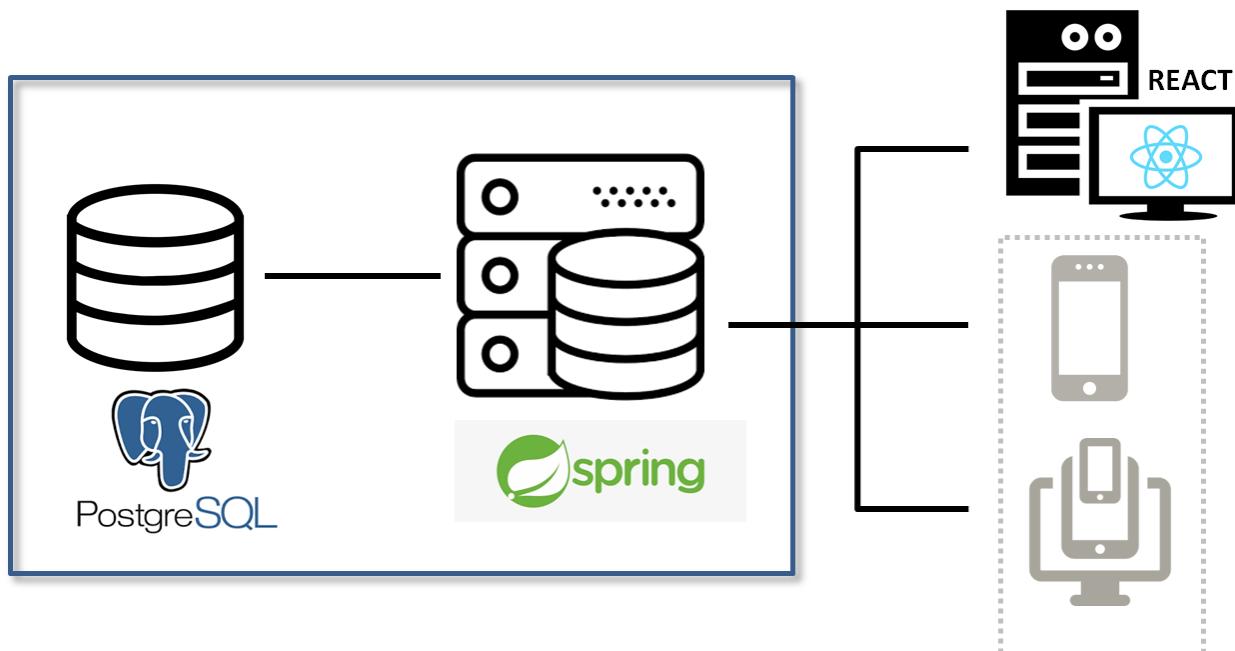
L'applicazione è strutturata su due livelli:

Back-end:

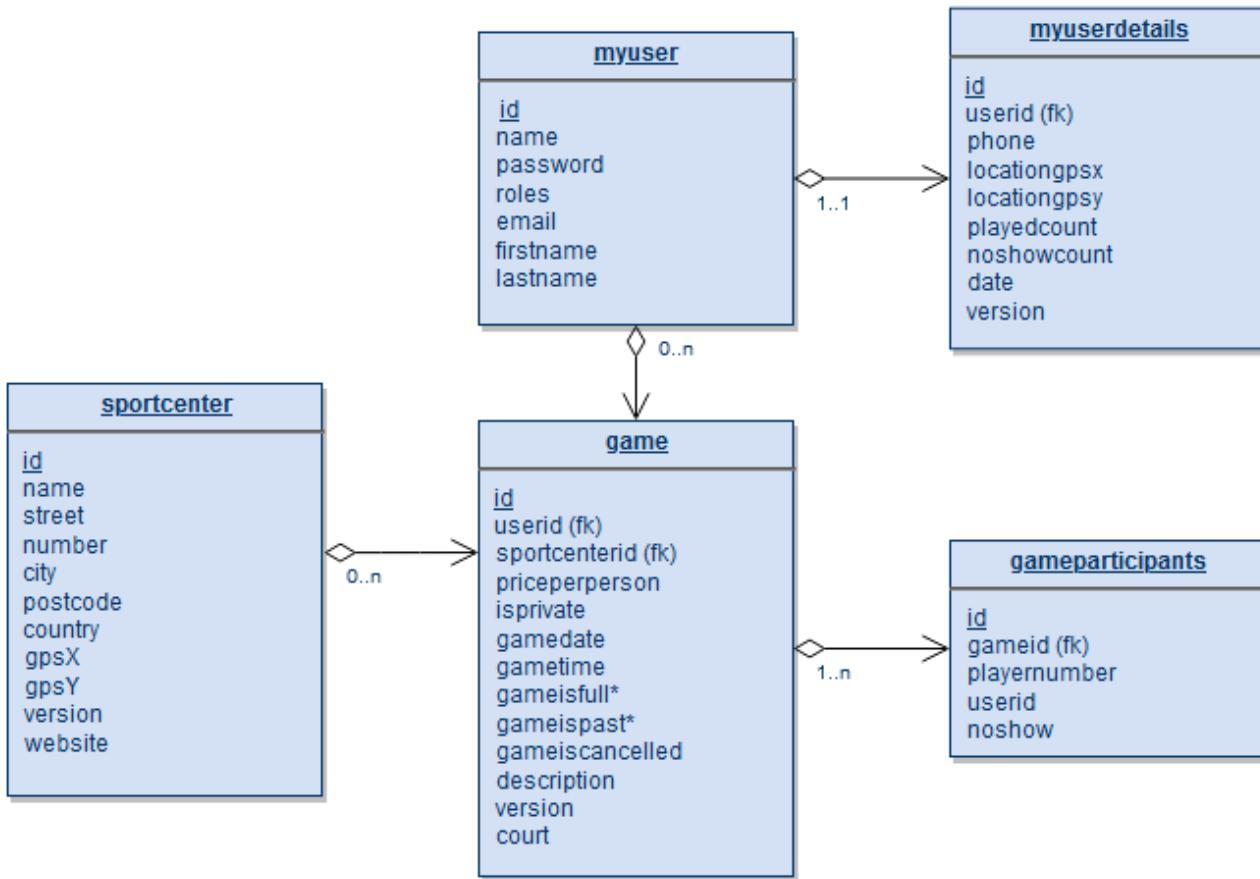
1. Database PostgreSQL che contiene tutti i dati inseriti dagli utenti
2. API Rest realizzata in Java utilizzando il framework SPRING

Front-end:

L'architettura del back-end permette l'utilizzo di diverse soluzioni per il front-end che possono utilizzare le chiamate http per comunicare con l'API. In questo caso si è utilizzato JS React.

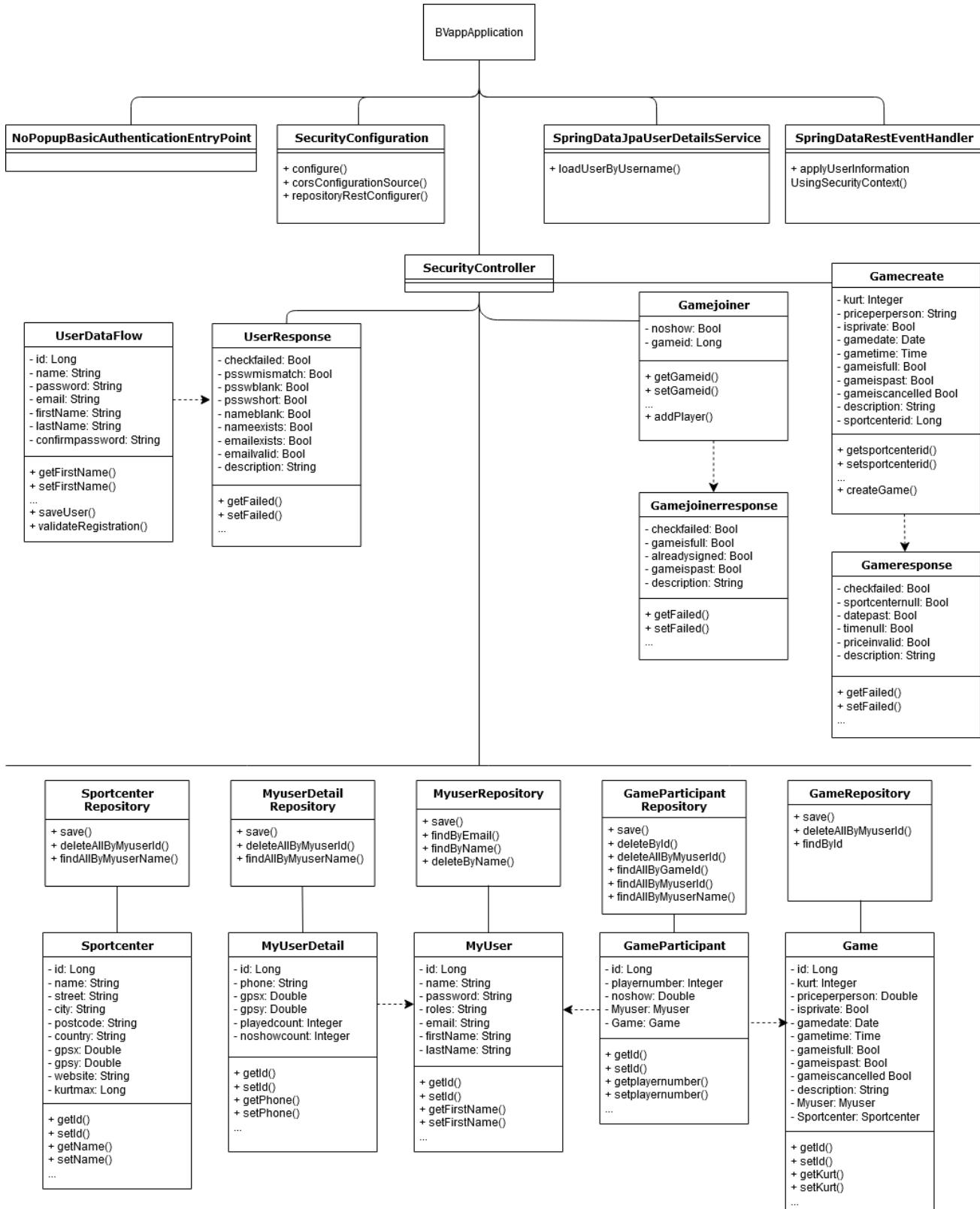


4. PostgreSQL database - diagramma ER



5. Java SPRING API

5.1 Java SPRING API – UML



5.2 Organizzazione del codice: funzioni principali

1. Collegamento tra oggetti e DBS

```
@Entity // declares that this class is meant for storage in a dfs table
public class Game {

    private @Id @GeneratedValue Long id; // automatically generated primary ID
    private Integer kurt;
    private Double priceperperson;
    private Boolean isprivate;
    private Date gamedate;
    private Time gametime;
    private Boolean gameisfull;
    private Boolean gameispast;
    private Boolean gameiscancelled;
    private String description;
```

Per ogni classe collegata al database PSQL viene utilizzata l'annotazione `@entity` che segnala il fatto che le varie istanze sono destinate ad essere salvate in una tabella di un database.

Il collegamento tra l'applicazione e il database è gestito attraverso dei repository in cui viene configurato anche l'accesso.

Per esempio il repository “GameparticipantRepository” autorizza il metodo “deleteById” solo agli utenti con ruolo: ADMIN.

```
//only users or admins can modify
@PreAuthorize("hasRole('ROLE_USER') or hasRole('ROLE_ADMIN')")
public interface SportCenterRepository extends
PagingAndSortingRepository<Sportcenter, Long> { // enable paging support

    @PreAuthorize("hasRole('ROLE_ADMIN')")
    void deleteById(Long id);
```

Un secondo esempio è quello del myuser repository che interagisce con i dati degli utenti dell'applicazione. In questo caso nessuna restrizione è imposta, per far sì che un utente non registrato possa registrarsi creando un nuovo utente. L'annotazione `@RepositoryRestResource` fa sì che nessuna operazione su myuser esponga un endpoint API ad eccezione di quelle “custom” create nel SecuriyController.

```
// user details are not visible to other users and are not exposed through the API
@RepositoryRestResource(exported = false)
public interface MyuserRepository extends JpaRepository<Myuser, Long> {

    // no access restrictions. This is needed to allow registration (see
    SecurityController)

    Myuser save(Myuser myuser);
```

2. SecurityController

Attraverso il controller vengono esposti endpoint API necessari per operazioni che non sono effettuabili attraverso l'interfaccia standard REST di SPRING.

Tutti questi metodi utilizzano le annotation “RequestMapping” (standard per esporre un endpoint), “ResponseBody” per definire la risposta alla chiamata all’API e **@RequestBody** per definire il formato della chiamata.

Esempio:

```
// expose API endpoint for joining a game
@RequestMapping(value = "/api/gameparticipantspost", method =
RequestMethod.POST, produces = "application/json")
@ResponseBody
public Gamejoinerresponse signupforagame(@RequestBody Gamejoiner gamejoiner,
HttpServletRequest httpServletRequest, Principal principal) {

    MyuserRepository userrepo = context.getBean(MyuserRepository.class);
    GameRepository gamerepo = context.getBean(GameRepository.class);
    GameparticipantRepository gamepartrepo =
context.getBean(GameparticipantRepository.class);

    // get current user from the logged user ID
    Myuser myuser = userrepo.findByName(principal.getName());
    // save the game participant
    Gamejoinerresponse tryjoingame = gamejoiner.addPlayer(gamejoiner, myuser,
gamerepo, gamepartrepo, 4);
    // print the result from the request to the console
    System.out.println(tryjoingame.resultdescription);
    // return the result from the request as API response
    return tryjoingame;
```

3. SecurityConfiguration

Questa classe definisce le impostazioni di sicurezza generale dell’applicazione.

Viene definita la politica di authentication (basic in questo caso), viene abilitato il CORS (Cross Origin Resource Sharing) per evitare che il browser blocchi le richieste inviate dal front-end alla API, vengono definiti gli endpoint pubblici che non richiedono login (/signup) e i metodi REST permessi dalla API (i.e. GET, POST, DELETE, ...).

```

@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .cors().and() // enable cors
        .authorizeRequests()
            .antMatchers("/signup").permitAll()
        // grants unconditional access to certain paths
            .anyRequest().authenticated()
        // anything else requires authentication to be accessed
            .and()
        .formLogin()
            .defaultSuccessUrl("/api/", true)
        // defaults to /api on login
            .permitAll()
            .and()
        .httpBasic() // basic login
            .authenticationEntryPoint(new
NoPopupBasicAuthenticationEntryPoint())
        // prevents the default username/password browser popup from
        appearing when the React front-end tries to connect to the API
            .and()
        .csrf().disable()
        // !!!!!!!!!!!!!!! basic authentication is on and csrf disabled, to
        be changed for prod release (cross-site request forgery)
            .logout()
                .logoutSuccessUrl("/");
}

```

4. Request validation: UserDataFlow, Gamejoiner, Gamecreate

La validazione dei dati per creare un nuovo record nel database avviene attraverso delle classi specifiche.

Per esempio UserDataFlow riceve i dati forniti dall'utente in fase di registrazione e li convalida attraverso il metodo: validateRegistration(). ValidateRegistration restituisce un oggetto UserResponse che contiene i dettagli e l'esito di tutti i controlli effettuati sui dati che possono essere usati dal front-end per generare messaggi di errore specifici.

```

// validate a new user request and return the result
public UserResponse validateRegistration(String password1, String password2,
String name, String email, MyuserRepository repo)
{
    // create a new empty user response
    UserResponse usercheck = new UserResponse(false, false, false, false,
false, false, false, false, "");

    // name cannot be null or empty
    if(name == null || name.equals(""))
    {
        usercheck.setFailed(true);
        usercheck.setNameBlank(true);
        usercheck.setDescription(usercheck.resultdescription + "The name
field cannot be blank. ");

    }

    // the two passwords supplied must match
    if(!password1.equals(password2))
    {
        usercheck.setFailed(true);
        usercheck.setPasswordMismatch(true);
        usercheck.setDescription(usercheck.resultdescription + "The two
passwords must match. ");
    }

...

```

Se il controllo ha successo (checkfailed = false) il metodo saveUser() crea un nuovo utente nel database:

```

public UserResponse saveUser(MyuserRepository repo, MyUserDetailsRepository
repo2)
{
    UserResponse usercheck = this.validateRegistration(this.password,
this.confirmPassword, this.name, this.email, repo);

    // if the usercheck is successful then create a new user
    if(usercheck.checkfailed == false)
    {
        Myuser myuser = new Myuser(
            this.name, this.password, "ROLE_USER",
            this.email, this.firstname, this.lastname);
        myuser = repo.save(myuser);

        java.sql.Date nowdate = new
java.sql.Date(Calendar.getInstance().getTime().getTime());

        MyUserDetail myuserdetail = new MyUserDetail("", 0.0, 0.0,
0, 0, myuser, nowdate);
        myuserdetail = repo2.save(myuserdetail);
    }
    return usercheck;
}

```

Esempio di utilizzo di UserResponse nel front end:

Fill in your details:

Username*	pareto	the username already exist
First Name*	testname	
Last Name*	testsurname	
Email*	notanemail	please provide a valid email address
Password*	*****	the password must be at least 8 chars long
Repeat Password*		you have typed two different passwords

Register

I messaggi di errore sono basati sulla risposta fornita dalla API:

```
▼ JSON
  checkfailed: true
  psswmismatch: true
  psswblank: false
  psswshort: true
  nameblank: false
  nameexists: false
  emailexists: false
  emailvalid: true
  resultdescription: The two passwords must match. The password should be at least 8 characters long. Please enter a valid email address.
  description: The two passwords must match. The password should be at least 8 characters long. Please enter a valid email address.
```

5.3 Testing

Gli Unit test sono stati svolti con JUnit e utilizzando le funzionalità di testing fornite da SPRING.

I test sono in tutto 12, contenuti nella classe TestingAPI.class

Per ogni test viene creato e poi cancellato (all'inizio e alla fine del test) un utente in modo da poter eseguire test sui componenti che richiedono un login. Tutti i test che richiedono la creazione di un utente utilizzano i metodi: registerUser() e deleteCurrentUser().

I test sono i seguenti, con alcuni esempi di codice:

1. **accessNoCredentials()** - Accesso senza credenziali (risultato atteso: utente non autorizzato)

```

// test access to API without credentials
@Test
public void accessNoCredentials() throws Exception {
    this.mockMvc.perform(get("/"))
        .andExpect(status().isUnauthorized());
}

```

2. **loginUser()** - Accesso con credenziali corrette (utilizza un utente esistente, risultato atteso: autenticazione completata)
3. **loginInvalidUser()** - Accesso con credenziali non corrette (risultato atteso: autenticazione fallita)
4. **registerCorrect()** - Creazione nuovo utente (risultato atteso: creazione riuscita, nessun errore di validazione)
5. 6. **registerWrong_1(),registerWrong_2(), registerWrong_3()** – Registrazione utente con dati non validi (risultato atteso: creazione non riuscita, errori di validazione presenti)

```

// User registration validation error 1
@Test
public void registerWrong_1() throws Exception {
    // create a user
    this.registerUser();
    // create a duplicateuser with incorrect data
    UserDataFlow newuser = new UserDataFlow("mockusr", "mock", "user",
"pass", "pss123", "email@email.com");
    //MvcResult result =
        this.mockMvc
            .perform(post("/signup")
                .contentType(MediaType.APPLICATION_JSON)
                .content(asJsonString(newuser)))
                .andExpect(status().isOk())

    .andExpect(MockMvcResultMatchers.jsonPath("$.checkfailed").value(true))
    .andExpect(MockMvcResultMatchers.jsonPath("$.psswmismatch").value(true))
    .andExpect(MockMvcResultMatchers.jsonPath("$.psswblank").value(false))
    .andExpect(MockMvcResultMatchers.jsonPath("$.psswshort").value(true))
    .andExpect(MockMvcResultMatchers.jsonPath("$.nameblank").value(false))
    .andExpect(MockMvcResultMatchers.jsonPath("$.nameexists").value(true))
    .andExpect(MockMvcResultMatchers.jsonPath("$.emailexists").value(true))
    .andExpect(MockMvcResultMatchers.jsonPath("$.emailvalid").value(false));
        this.deleteCurrentUser();
}

```

8. **createGameSuccess()** – Creazione di una nuova partita (risultato atteso: partita creata, nessun errore di validazione)

9. **joinGame_success()** – Aggiungere un utente ad una partita esistente (risultato atteso: utente aggiunto, nessun errore di validazione).
10. **joinGame_duplicate()** – Tentativo di aggiungere un utente ad una partita fallito: l'utente è già registrato (risultato atteso: errori di validazione riportati correttamente)

```

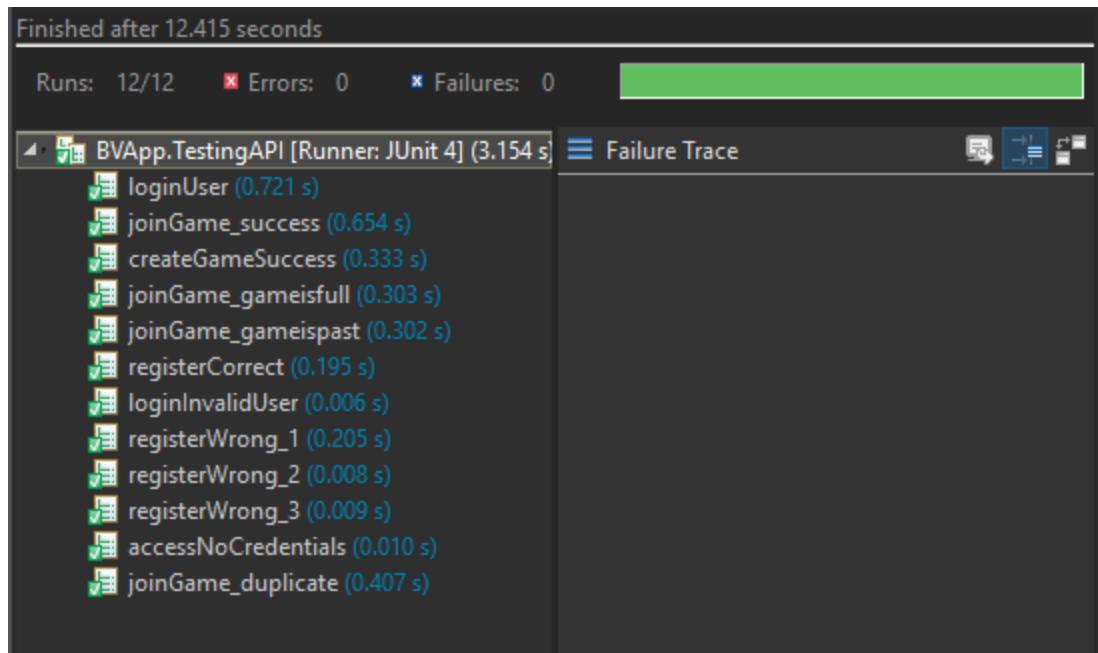
    @Test
    public void joinGame_duplicate() throws Exception {
        this.registerUser();
        //MvcResult result =
        Gamejoiner player = new Gamejoiner(false, (long) 347);

        this.mockMvc
            .perform(post("/api/gameparticipantspost").with(httpBasic("mockusr",
"password")))
                .contentType(MediaType.APPLICATION_JSON)
                .content(asJsonString(player)))
                    .andExpect(status().isOk());
        this.mockMvc
            .perform(post("/api/gameparticipantspost").with(httpBasic("mockusr",
"password")))
                .contentType(MediaType.APPLICATION_JSON)
                .content(asJsonString(player)))
                    .andExpect(status().isOk())
            .andExpect(MockMvcResultMatchers.jsonPath("$.checkfailed").value(true))
            .andExpect(MockMvcResultMatchers.jsonPath("$.gameisfull").value(false))
            .andExpect(MockMvcResultMatchers.jsonPath("$.alreadysigned").value(true))
            .andExpect(MockMvcResultMatchers.jsonPath("$.gameispast").value(false));
        this.deleteCurrentUser();
    }
}

```

11. **joinGame_gameisfull ()** – Tentativo di aggiungere un utente ad una partita fallito: la partita è completa (risultato atteso: errori di validazione riportati correttamente)
12. **joinGame_gameispast()** - Tentativo di aggiungere un utente ad una partita fallito: la data della partita è nel passato (risultato atteso: errori di validazione riportati correttamente)

Tutti i test sono stati completati con successo:

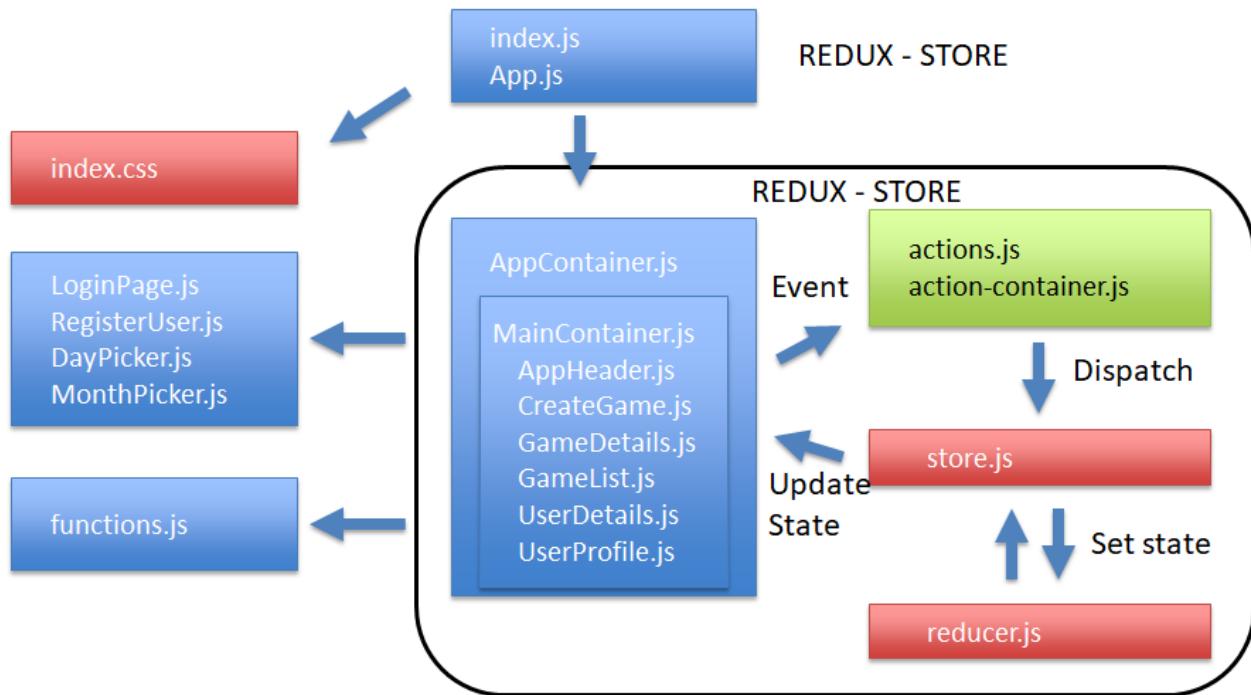


La API è stata anche testata con Postman, alcuni dei risultati ottenuti sono pubblicati a questo link:

<https://documenter.getpostman.com/view/10259016/SWTD9crr?version=latest#3209a188-134c-4e3a-8d64-69f3f845581b>

6. JS React Web APP

6.1 Organizzazione del codice



Funzioni principali

1. Actions: GET data from API

Tutte le chiamate GET all'API (dopo il login) sono gestite dalla funzione loaddata().

```
export function loadData(request, url, params) {
    return (dispatch) => {
        dispatch(isError(false));
        // start GET request
        fetch(url+request+params , {
            method: 'GET',
            credentials: 'include',
            headers: {
                'Content-Type': 'application/json',
            }})
    }
}
```

LoadData() verifica che l'utente sia ancora connesso (!error 401) e nel caso non lo sia effettua il dispatch loginsuccessful=false.

```

// evaluate get request
.then((res) => {
    if(res.status == '401') {
        console.log("unauthorized " + res.status);
        dispatch(loginsuccessfull(false));
        if(request == 'api/user') {
            dispatch(setuser('Guest'));
        }
    }
    else {
        console.log("success " + res.status);
        if(request == 'api/user')
        {
            dispatch(loginsuccessfull(true));
        }
    }
})

```

Il risultato della chiamata all'API viene serializzato in formato json.

```

        }
        return res.json();
    })
}

```

In base alla chiamata fatta (parametro “request”) l’informazione ricevuta viene inviata allo store e immagazzinata in un’oggetto specifico (user, game, sportcenter, ...).

```

// get result based on keyword
.then((items) => {
    if(request == 'api/games')
        dispatch(itemsFetchDataSuccess(items))
    if(request == 'api/sportcenters')
        dispatch(sportcenterFetchDataSuccess(items));
    if(request == 'api/myUserDetails' || request == 'api/myuser
det')
        dispatch(userFetchDataSuccess(items));
    if(request == 'api/gameparticipantget')
        dispatch(gameparticipantFetchDataSuccess(items));
    if(request == 'api/singlegame')
        dispatch(singlegameFetchDataSuccess(items))
})

```

La chiamata per i dati utente viene effettuata da ogni pagina e serve anche a verificare che l'utente sia ancora connesso.

```
        if(request == 'api/user')
        {
            dispatch(myuserFetchDataSuccess(items));
            if(store.getState().loginsuccessfull)
                dispatch(setuser(items.name));
        }
        if(request == 'api/gameparticipantsbyuser')
            dispatch(usergamesFetchDataSuccess(items));
        }
    )
```

Un errore “undefined” segnala che l’API non è raggiungibile.

```
// error handling
    .catch(error => {
        if(typeof error.status == 'undefined' && request == 'api/user')
        {
            dispatch(isError(true));
            console.log("connection to server failed: " + error)
        }
    })
```

2. Actions: Navigazione

La navigazione tra le pagine viene gestita attraverso la funzione setNavigate(string) e un array di pagine visitate che si comporta come uno stack.

Se la funzione è chiamata senza parametro si rimuove il riferimento in cima alla pila (chiusura finestra, torna alla pagina precedente), mentre in presenza del parametro viene aggiunto il riferimento alla nuova pagina.

Ad ogni utilizzo della funzione setNavigate vengono ricaricate le informazioni dell'utente, per verificare che sia ancora connesso alla API.

```
export function setNavigate(input) {
    return(dispatch) => {var tempvar = []

        dispatch(isLoading(true));
        tempvar = store.getState().navigate;
        if(input == '')
            tempvar.pop();
        else if (tempvar[tempvar.length-1] != input)
            tempvar.push(input);

        dispatch(loadData('api/user', url, ''))

        if(tempvar[tempvar.length-1] == 'games')
        {
            dispatch(loadData('api/games', url, '?page=0&size=1000&sort=gamedate&sort=gametime'))
            dispatch(loginsuccessfull(true));
            dispatch(loadData('api/myuserdet', url, ''))
        }
        ...
        if(tempvar[tempvar.length-1] == 'gamedetails')
        {
            dispatch(loadData('api/gameparticipantsget', url, '?id=' + store.getState().currentgame))
            dispatch(loadData('api/singlegame', url, '?id=' + store.getState().currentgame))
        }
        dispatch(isLoading(false))

    return {
        type: SET_NAVIGATE,
        navigate: tempvar,
    };
}
```

In base alla nuova pagina di destinazione viene effettuata la chiamata a loaddata in modo da aggiornare i dati rilevanti.

Alla fine lo store viene aggiornato con l'array completo.

Il comando isLoading manda il segnale ai components in modo che il messaggio di caricamento in corso venga visualizzato mentre l'applicazione sta leggendo i dati dall'API.

3. Components

Il componente MainComponent.js gestisce la visibilità dei vari componenti.

Si verifica che l'utente abbia effettuato il login e quale sia la pagina richiesta. Inoltre si verifica che i dati siano disponibili, ovvero che l'oggetto contenente i dati non abbia stato: "undefined".

Nel caso in cui l'utente non abbia effettuato il login l'applicazione mostrerà la pagina di login.

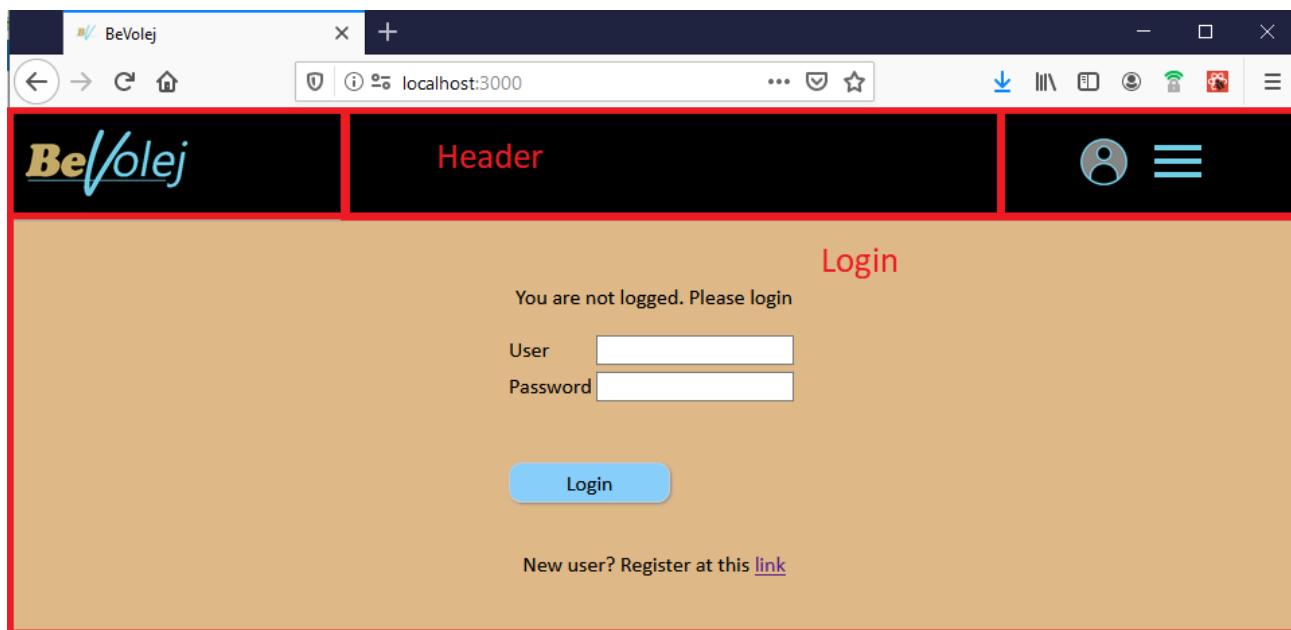
```
if(this.props.loginsuccess && !this.props.isLoading)
{
    if(this.props.navigate[this.props.navigate.length-1] == 'games' &&
        typeof this.props.items._embedded != 'undefined' &&
        typeof this.props.useritems[0].id != 'undefined')
    {
        return(<div className='maindiv'>
            <UserComponent /><GameComponent /></div>)
    }

    if(!this.props.loginsuccess && !this.props.isLoading &&
        !this.props.isError &&
        this.props.navigate[this.props.navigate.length-1] != 'register')
    {
        return(<LoginComponent />)
    }
}
```

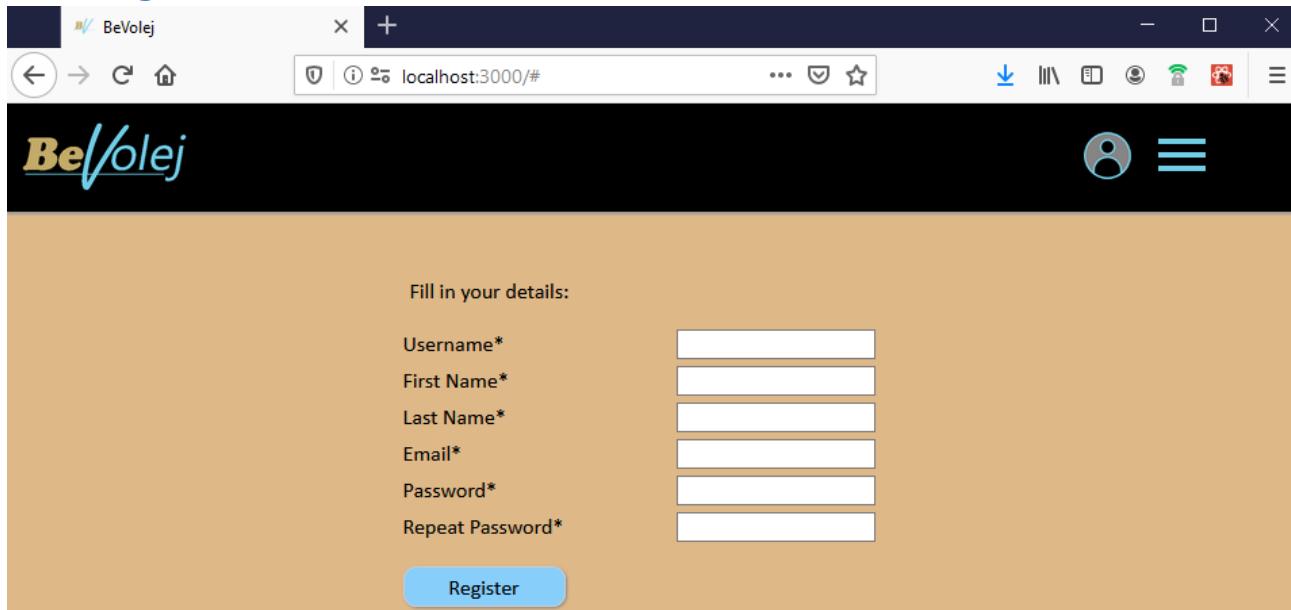
6.2 Componenti

Lo stile e la visibilità di ognuno dei componenti cambia a seconda della risoluzione dello schermo in modo da ottimizzare la visualizzazione anche su mobile.

1. Login and Header



2. Register



3. Main page (GameList e UserDetails)

The screenshot shows the BeVolej application interface. At the top right, there is a user profile icon and a dropdown menu with options: Logout, Game list, Create game, and pareto (which is highlighted). The main content area has a header "Games near your location" with a "+ Create New" button. Below this, a list of games is displayed:

Date	Time	Location	Organized by
Today	20:00	@Beachklub Pankrac, Horackova 1100	pareto
2020-02-21	14:30	@DOMYNO Sports Academy, Novodvorská	admin
2020-02-21	15:30	@Gutovka, Gutova 39	sraffa
2020-02-22	14:00	@DOMYNO Sports Academy, Novodvorská	pacio
2020-02-23	16:00	@Sportovni areal Beachklub Ladvi, Chabarovicka 1125/4	pareto
2020-02-25	16:00	@Gutovka, Gutova 39	pareto
2020-02-29	19:00	@Gutovka, Gutova 39	pareto
2020-03-05	15:00	@Gutovka, Gutova 39	pareto
2020-03-11	13:30	@DOMYNO Sports Academy Novodvorská	pacio

On the left side, there is a sidebar for the user "Vilfredo Pareto (pareto)" showing statistics: Games Played (0), Games Missed (0), Phone number, and Email (pareto@uni.it). The background of the application is orange.

Mobile:

The mobile screenshot shows the BeVolej application running on a smartphone. The top status bar indicates the carrier (T-Mobile), signal strength, battery level (50%), and the time (15:43). The address bar shows the URL 192.168.0.103:3000/. The main screen features the "BeVolej" logo, a user profile icon, and a three-dot menu icon. A "+ Create New" button is at the top left. The "Games near your location" section is displayed, listing the same events as the desktop version. The event for February 22nd at 14:00 is highlighted with a blue background. At the bottom, there is a navigation bar with icons for back, forward, refresh, a search bar containing the number 7, and a menu icon.

4. Create Game

X

Where?

Beachklub Pankrac (Horackova 1100 - <https://www.beachklub.cz/>)>

DOMYNO Sports Academy (Novodvorská - <http://www.domyno.cz/>)>

Gutovka (Gutova 39 - <https://www.gutovka.cz/>)>

Sportovni areal Beachklub Ladvi (Chabarovicka 1125/4 - <https://beachklubladvi.cz/>)>

Which day?

<<	February	>>
1	2	3
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18
19	20	21
22	23	24
25	26	27
28	29	

What time?

20:00

Which court?

1

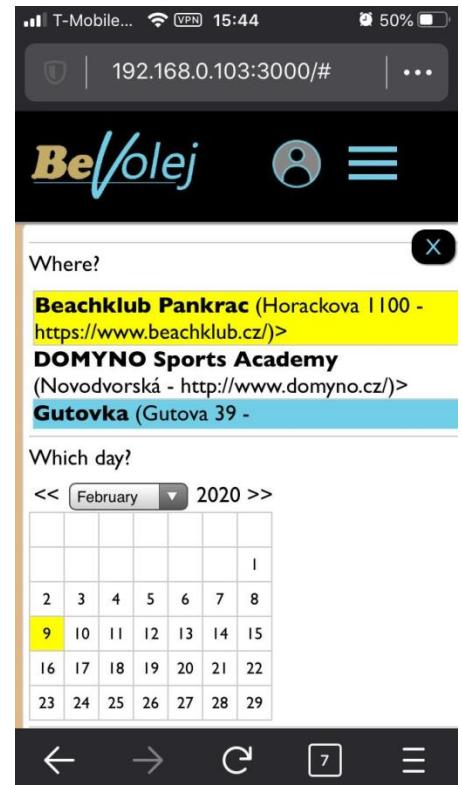
How much per person?

CZK

Private game?

Extra comments (500 chars max)

Done



5. Game Details

The screenshot shows a web browser window with the URL `localhost:3000/#`. The main content is a game detail page for "DOMYNO Sports Academy 250". The page includes the following information:

- Organizer:** DOMYNO Sports Academy 250
- Date:** 2020-11-18
- Start Time:** Starts at 02:30
- Court:** Court n. 15
- Cost:** You will pay: 600 CZK
- Message from the organizer:** Message from the organizer: (link)
- Status:** This game is public
- Organizer Information:** This game is organized by Vilfredo Pareto (link to cancel registration)

The page also lists four players for the game:

Player Number	Player Name	Action
1	Piero Sraffa	
2	Enrico Barone	
3	Luca Pacioli	
4	Vilfredo Pareto	cancel registration

Mobile:

The left screenshot shows a game detail screen for "DOMYNO Sports Academy 250" on 2020-11-18 at 02:30 in Court n. 15. It costs 600 CZK and is organized by Vilfredo Pareto. A message from the organizer is present. The right screenshot shows the "Players for this game" section, listing Enrico Barone (2), Luca Pacioli (3), and Vilfredo Pareto (4). Each player has a placeholder profile picture.

6. User Profile

The user profile page for Vilfredo Pareto (pareto) shows basic information: profile picture, name, member since (2020-01-20), and stats (0 games played, 0 missed). It also lists recent games (organized by pareto), upcoming games (organized by sraffa, pareto, or pareto), and a sidebar for "Your games" and "Your data".

Recent Games:

- 2020-01-31, 20:00 @Gutovka, Gutova 39 Organized by: pareto
- 2020-01-31, 20:00 @Gutovka, Gutova 39 Organized by: pareto
- 2020-01-14, 20:00 @Beachklub Pankrac, Horackova 1100 Organized by: pareto

Upcoming Games:

- 2032-02-02, 20:00 @Gutovka, Gutova 39 Organized by: sraffa
- 2031-02-26, 20:00 @Sportovni areal Beachklub Ladvi, Chabarovicka 1125 Organized by: pareto
- 2020-11-18, 02:30 @DOMYNO Sports Academy, Novodvorská Organized by: pareto
- 2020-10-20, 20:00 @Beachklub Pankrac, Horackova 1100 Organized by: pareto

6.3 User Testing - Casi di test e risultati

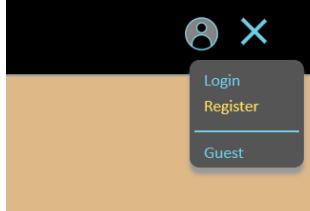
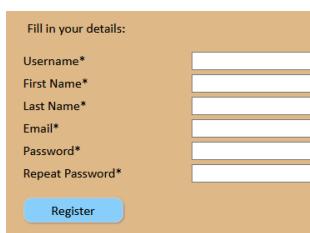
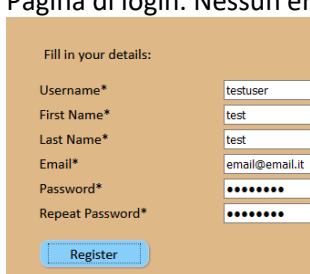
1. Creazione nuovo utente (R1)
2. Errori creazione nuovo utente (R1)
3. Creazione nuova partita (R5)
4. Creazione nuova partita con errori (R5)
5. Cancellazione partita creata (R5)
6. Cancellazione partita creata da un altro utente (R2, R5)
7. Iscrizione a partita completa (R6)
8. Cancellazione iscrizione a partita > 24 ore dall'evento (R7)
9. Cancellazione iscrizione a partita < 24 ore dall'evento (R7)
10. Inserimento dati utente (R3)

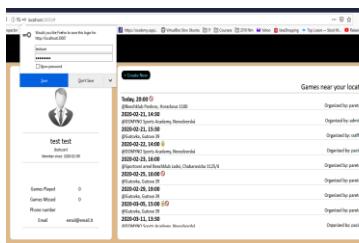
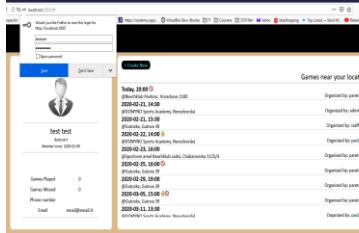
1. Creazione nuovo utente (R1)

Data: 14 Feb 2020

Prerequisiti al test: nessuno

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Premere l'opzione "register" nel menu dell'header.	Il menu si chiude. Compare il form registrazione utente.	Menu chiuso, form registrazione utente visibile.  
2	Inserire dati corretti: Username: testuser Email: email@email.it Password: password Rep.psw: password Premere il tasto "register".	Torna alla pagina di login. Nessun messaggio di errore visibile.	pagina di login. Nessun errore. 

			 
3	Effettuare il login per verificare che l'utente sia stato creato correttamente.	L'applicazione apre la pagina principale.	Pagina principale visibile.  

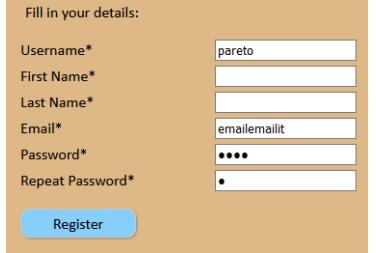
2. Errori creazione nuovo utente (R1)

Data: 14 Feb 2020

Prerequisiti al test: nessuno

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Premere l'opzione "register" nel menu dell'header.	Il menu si chiude. Compare il form registrazione utente.	Menu chiuso, form registrazione utente visibile. 
2	Inserire dati non corretti: Username: pareto (duplicato) Email: emailemailit Password: pass	Gli errori inseriti sono visibili.	Gli errori inseriti sono riportati dall'interfaccia.

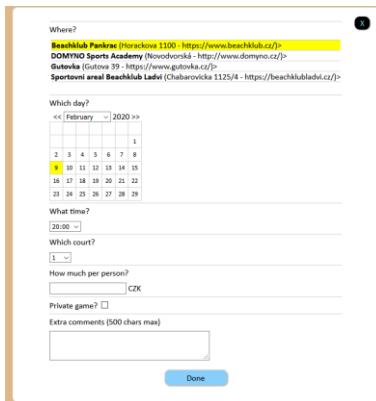
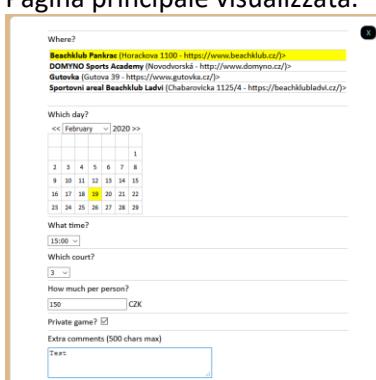
	<p>Rep.pssw: p Premere il tasto “register”.</p>	 <p>Fill in your details:</p> <p>Username* <input type="text" value="pareto"/></p> <p>First Name* <input type="text"/></p> <p>Last Name* <input type="text"/></p> <p>Email* <input type="text" value="emailemailit"/></p> <p>Password* <input type="password" value="****"/></p> <p>Repeat Password* <input type="password" value="•"/></p> <p>Register</p>
		 <p>Fill in your details:</p> <p>Username* <input type="text"/></p> <p>First Name* <input type="text"/></p> <p>Last Name* <input type="text"/></p> <p>Email* <input type="text"/></p> <p>Password* <input type="password"/></p> <p>Repeat Password* <input type="password"/></p> <p>Register</p> <ul style="list-style-type: none"> <input type="text"/> the username already exist <input type="text"/> please provide a valid email address <input type="text"/> the password must be at least 8 chars long <input type="text"/> you have typed two different passwords

3. Creazione nuova partita (R5)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo																													
1	Premere il tasto “Create new”	La pagina creazione nuova partita compare.	 <p>Where? <input type="text" value="Beachclub Parkresi (Horacekovo 1100 - https://www.beachklub.cz/p... DOMYNO Sports Academy (Novodvorská - http://www.domyno.cz/)... Gutovka (Gutova 39 - https://www.gutovka.cz/)... Sportovni areal Beachclub Ladvi (Chabarovicka 1125/4 - https://beachklubladvi.cz/...)"/></p> <p>Which day? <input type="button" value="February <> 2020 >>"/></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>1</td></tr> <tr><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr> <tr><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td></tr> <tr><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td></tr> </table> <p>What time? <input type="text" value="20:00 ->"/></p> <p>Which court? <input type="text" value="1 ->"/></p> <p>How much per person? <input type="text" value="CZK"/></p> <p>Private game? <input type="checkbox"/></p> <p>Extra comments (500 chars max) <input type="text"/></p> <p>Done</p>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1																																
2	3	4	5	6	7	8																										
9	10	11	12	13	14	15																										
16	17	18	19	20	21	22																										
23	24	25	26	27	28	29																										
2	Inserire dati corretti e premere “Done” Giorno: 19 Feb 2020 Ora: 15:00 Court: 3 Price: 150 Private game: yes Extra comments: test	L’applicazione passa alla schermata iniziale. La partita creata è visibile con data e ora corrette. L’indicatore partita privata è visibile.	 <p>Where? <input type="text" value="Beachclub Parkresi (Horacekovo 1100 - https://www.beachklub.cz/p... DOMYNO Sports Academy (Novodvorská - http://www.domyno.cz/)... Gutovka (Gutova 39 - https://www.gutovka.cz/)... Sportovni areal Beachclub Ladvi (Chabarovicka 1125/4 - https://beachklubladvi.cz/...)"/></p> <p>Which day? <input type="button" value="February <> 2020 >>"/></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>1</td></tr> <tr><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr> <tr><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td></tr> <tr><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td></tr> </table> <p>What time? <input type="text" value="15:00 ->"/></p> <p>Which court? <input type="text" value="3 ->"/></p> <p>How much per person? <input type="text" value="CZK"/></p> <p>Private game? <input type="checkbox"/></p> <p>Extra comments (500 chars max) <input type="text" value="test"/></p>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1																																
2	3	4	5	6	7	8																										
9	10	11	12	13	14	15																										
16	17	18	19	20	21	22																										
23	24	25	26	27	28	29																										

La partita creata è visibile con data e ora corrette. L’indicatore

			partita privata è visibile
3	Cliccare sulla partita per verificarne i dettagli.	La pagina dei dettagli della partita si apre. Nessun giocatore è iscritto alla partita. I parametri inseriti sono corretti.	Pagina dettagli aperta. Nessun giocatore iscritto. Parametri verificati e corretti.

4. Creazione nuova partita con errori (R5)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Premere il tasto “Create new”	La pagina creazione nuova partita compare.	Pagina visibile.
2	Inserire dati non corretti e premere “Done” Giorno: 05 Feb 2020 Ora: 21:00 Price: lasciare il campo vuoto	Vengono visualizzati i messaggi di errore: - Partita con data nel passato - Prezzo non valido	Pagina principale visualizzata. Messaggi di errore visibili.

3	Premere il tasto x e verificare che: 1. l'applicazione torna alla schermata principale 2. la partita non è visibile	Schermata principale visibile. Nessuna partita creata.	

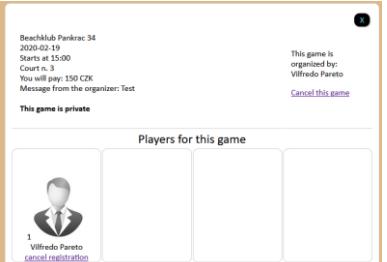
5. Iscrizione a una partita (R6)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido e test 3 eseguito con successo

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Cliccare sulla partita creata al test 3 (19/02/2020)	<p>La pagina dettagli partita si apre.</p> <p>Verificare che il link registrazione partita sia visibile.</p>	<p>Dettagli partita visibili.</p> <p>Link registrazione partita visibile.</p>

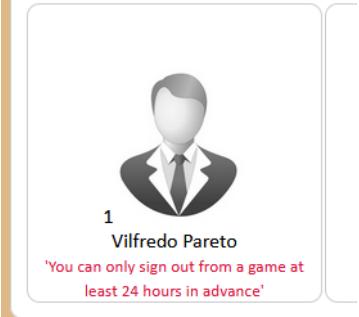
2	Premere il link: join this game	L'applicazione passa alla pagina principale.	Pagina principale aperta.
3	Cliccare sulla partita creata al test 3 (19/02/2020)	L'utente è stato aggiunto nella lista dei partecipanti	L'utente è stato aggiunto nella lista dei partecipanti 

6. Cancellazione iscrizione a partita < 24 ore dall'evento (R7)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido e creazione partita che inizia entro 24 ore (15 Feb 2020, ore 07:00)

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Iscriversi alla partita con inizio a meno di 24 ore	Verificare che il link per cancellarsi della partita non sia visibile e al suo posto appaia un messaggio che informa che non sia possibile cancellarsi a meno di 24 ore dall'inizio della partita.	Il link non è visibile e l'avviso è visualizzato correttamente 

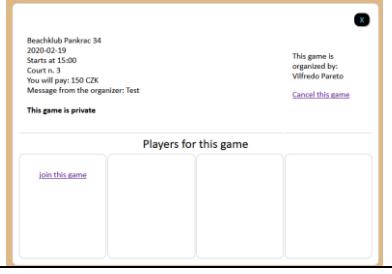
7. Cancellazione iscrizione a partita > 24 ore dall'evento (R7)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Cliccare sulla partita creata al test 3 (19/02/2020)	La pagina dettagli partita si apre.	Dettagli partita visibili. Link registrazione partita

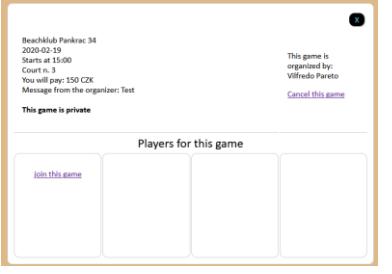
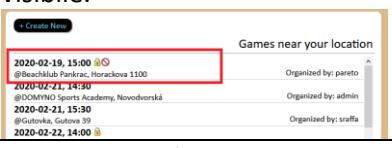
		Verificare che il link registrazione partita sia visibile e l'utente registrato.	visible. 
2	Premere il link: cancel registration	L'applicazione passa alla pagina principale.	Pagina principale aperta.
3	Cliccare sulla partita creata al test 3 (19/02/2020)	L'utente è stato rimosso dalla lista dei partecipanti	L'utente è stato rimosso dalla lista dei partecipanti 

8. Cancellazione partita creata (R5)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido e test 3 eseguito con successo

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Cliccare sulla partita creata al test 3 (19/02/2020)	La pagina dettagli partita si apre. Verificare che il link cancellazione partita sia visibile.	Dettagli partita visibili. Link cancellazione partita visibile. 
2	Premere il link: cancel this game	L'applicazione passa alla pagina principale. La partita presenta l'indicatore di partita cancellata.	Pagina principale aperta. Indicatore di partita cancellata visibile. 
3	Cliccare sulla partita creata al	Non è più possibile iscriversi	Link iscrizione/cancellazione

	test 3 (19/02/2020)	alla partita.	partita assente. 
--	---------------------	---------------	---

9. Cancellazione partita creata da un altro utente (R2, R5)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido e almeno una partita creata da un utente diverso da quello corrente

Risultato: Test completato con successo

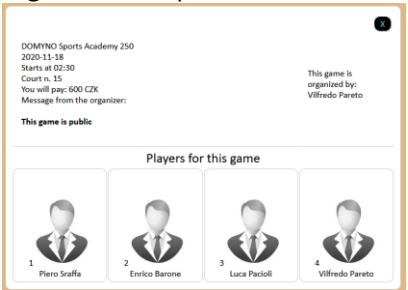
Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Cliccare sulla partita scelta	La pagina dettagli partita si apre. Verificare che il link cancellazione partita sia assente.	Dettagli partita visibili. Link cancellazione partita assente. 

10. Iscrizione a partita completa (R6)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido e almeno una partita creata in cui si siano registrati quattro utenti

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Cliccare sulla partita scelta	La pagina dettagli partita si apre. Verificare che non sia possibile iscriversi alla partita	Dettagli partita visibili. Non è presente alcun link per registrarsi alla partita. 

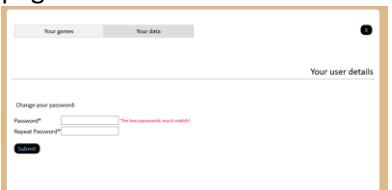
--	--	--

11. Inserimento dati utente (R3)

Data: 14 Feb 2020

Prerequisiti al test: login effettuato con un utente valido

Risultato: Test completato con successo

Step #	Descrizione	Risultato Atteso	Risultato effettivo
1	Cliccare sull'icona utente per accedere alla sezione dettagli utente.	La pagina dettagli utente si apre.	Dettagli utente visibili. 
2	Cliccare su "change password". Inserire due password diverse e premere "Submit".	Messaggio di errore. L'applicazione non cambia pagina.	Messaggio di errore visibile. L'applicazione non ha cambiato pagina. 
3	Inserire una nuova password. Effettuare il login ed il logout per verificare che la password sia cambiata.	E' possibile effettuare il login utilizzando la nuova password.	Login effettuato con successo.

6.4 User Testing – Riassunto

Sono stati eseguiti 11 test di cui 11 con esito positivo.

Durante il test non si sono verificati errori o deviazione rispetto ai risultati attesi.