



Progetto Academy 15

Iris Canole, Gabriele Garavelli, Gabriele
Marchetti, Samuele Melotto



Academy 15

Lezioni frontali:

- Design Pattern e framework (Spring, Hibernate) per la scrittura del BackEnd
- Architettura basata su MVC o SOAP
- Sviluppo del FrontEnd con framework (Bootstrap, Angular) e linguaggi di scripting (JavaScript, TypeScript)
- Tecniche di validazione, logging e test





Academy 15

Masterclass:

- Database con SQL e MySQL
- Versionamento con Git
- Lavoro Agile e metodologia SCRUM
- Ruolo del consulente

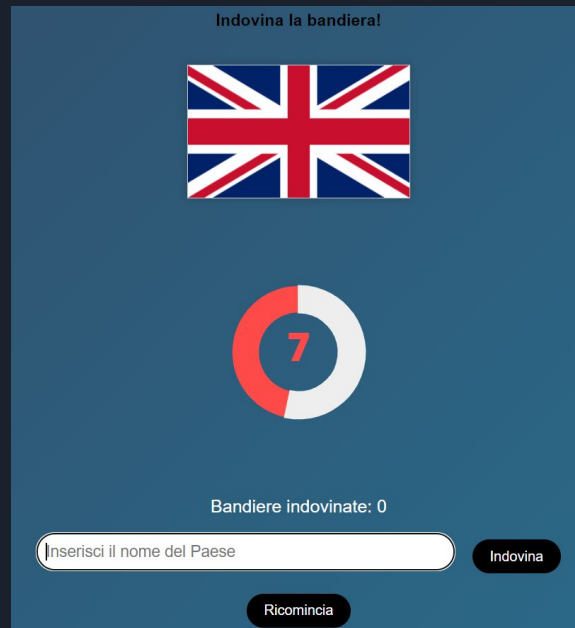


Guess the Flag



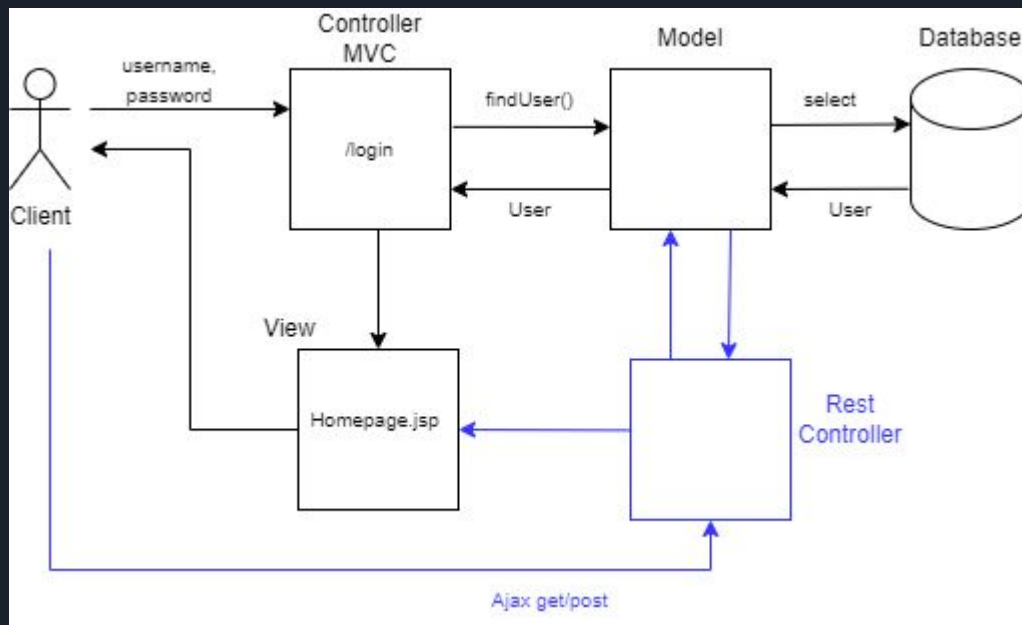
Un gioco in cui indovinare a quale Paese appartiene la bandiera visualizzata.

Se non si conosce lo spelling corretto del Paese che si vuole inserire non disperate, l'algoritmo di Check String troverà il nome corretto al posto vostro.

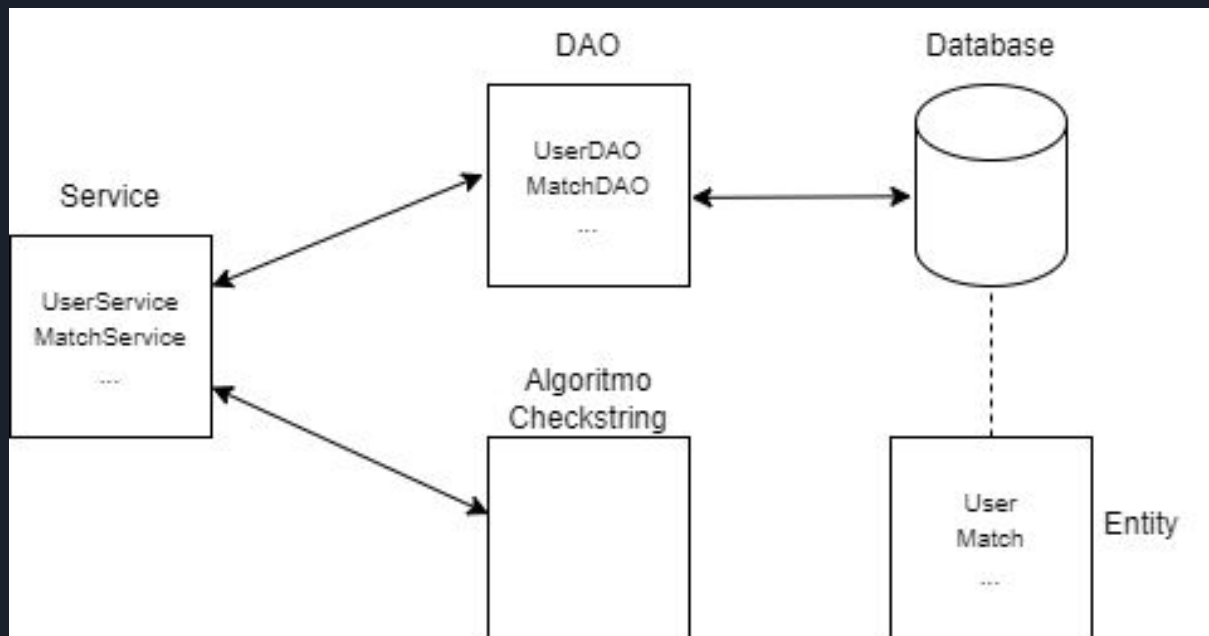


Architettura

Architettura MVC con utilizzo di servizi Rest per il recupero delle informazioni



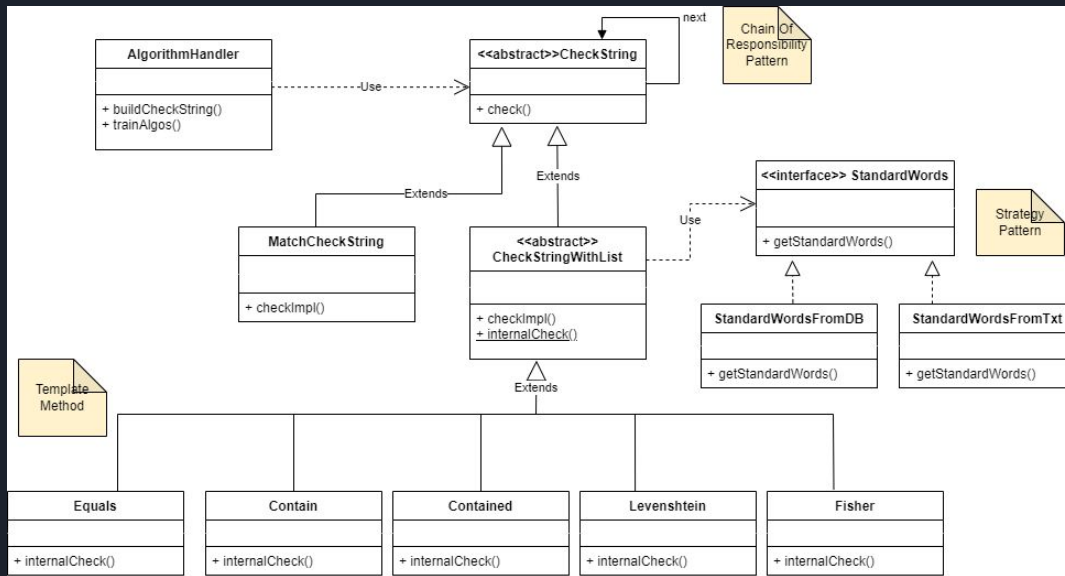
Struttura del BackEnd



UML

L'UML della porzione che gestisce l'algoritmo di Check String evidenzia l'utilizzo dei pattern:

- Strategy
- Chain of Responsibility
- Template Method



Pattern DAO

```
com.corso.dao
├── AlgorithmDAO.java
├── BandiereRisultatoDAO.java
├── BaseDAO.java
├── BeanDTO.java
├── MatchDAO.java
├── RisultatoDAO.java
├── SegnalazioneDAO.java
├── StandardWordDAO.java
├── UserDAO.java
└── com.corso.dao.impl
    ├── AlgorithmDAOImpl.java
    ├── BandiereRisultatoDAOImpl.java
    ├── BaseDAOImpl.java
    ├── MatchDAOImpl.java
    ├── RisultatoDAOImpl.java
    ├── SegnalazioneDAOImpl.java
    ├── StandardWordDAOImpl.java
    └── UserDAOImpl.java
```

```
public interface BaseDAO {

    public void update(BeanDTO o);

    public Object findOneByAttribute(Class c, String attribute, String value);

    public List<?> findAllByAttribute(Class c, String attribute, String value);

    public List<?> findAllByAttributeInt(Class c, String attribute, int value);

    public List<?> all(Class c);

    public Object find(Class c, Integer id);

    public Object find(Class c, String id);

    public BeanDTO create(BeanDTO o);

    public void remove(BeanDTO o);

}
```

```
public class MatchDAOImpl extends BaseDAOImpl implements MatchDAO{

    @Autowired
    private AlgorithmService algorithmService;

    public void create(Match match){
        super.create(match);
    }

    public void remove(Match match) {
        super.remove(match);
    }

    public Match find(Integer id){
        return (Match) super.find(Match.class, id);
    }

    public Match findByInput(String input) {
        return (Match) super.findOneByAttribute(Match.class, "Input", input);
    }

    public int countMatches(AlgorithmType type) {
        Algorithm algorithm = algorithmService.getAlgorithmByType(type);
        return super.findAllByAttribute(Match.class, "IdAlgoritmo", "" + algorithm.getId()).size();
    }

    @SuppressWarnings("unchecked")
    public List<Match> getAllMatches() {
        return (List<Match>) super.all(Match.class);
    }

}
```


JPA e Hibernate

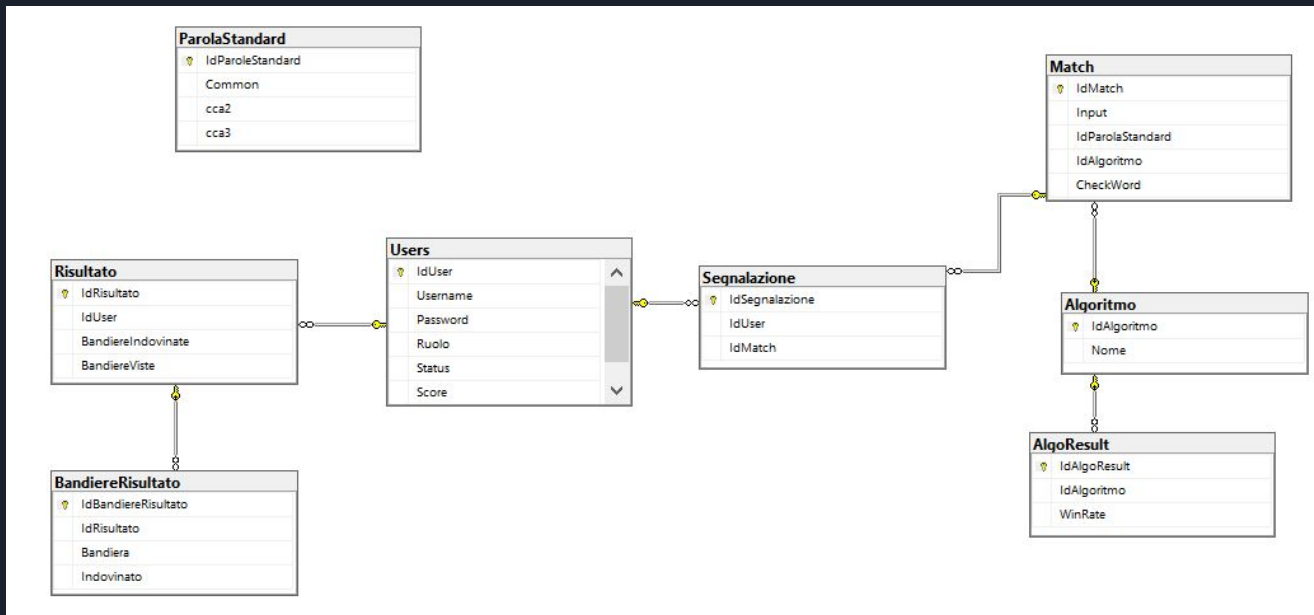
I DAO si collegano al Database tramite l'utilizzo di JPA e Hibernate. JPA permette di usare l'Entity Manager per comunicare con il DB in modo semplice. Mentre Hibernate permette di mappare le classi del Model con le tabelle nel DB.

```
15 @Entity
16 @Table(name = "Users")
17 public class User implements Comparable<User>, BeanDTO{
18
19     @Id
20     @GeneratedValue(strategy=GenerationType.IDENTITY)
21     @Column(name = "idUser")
22     private int id;
23
24     @Column(name="Username")
25     private String username;
26
27     @Column(name="Password")
28     private String password;
29
30     @Column(name="Ruolo")
31     private String ruolo;
32
33     @Column(name="Status")
34     private boolean status;
35
36     @Column(name="Score")
37     private int score;
38 }
```

```
14 public abstract class BaseDAOImpl implements BaseDAO{
15
16     EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("match_pu");
17     EntityManager manager = entityManagerFactory.createEntityManager();
18
19     @Transactional
20     public void update(BeanDTO o) {
21         try {
22             manager.getTransaction().begin();
23             manager.merge(o);
24             manager.getTransaction().commit();
25         }
26         catch (Exception e) {}
27     }
28 }
```

Database

Struttura del Database implementato con SQL Server

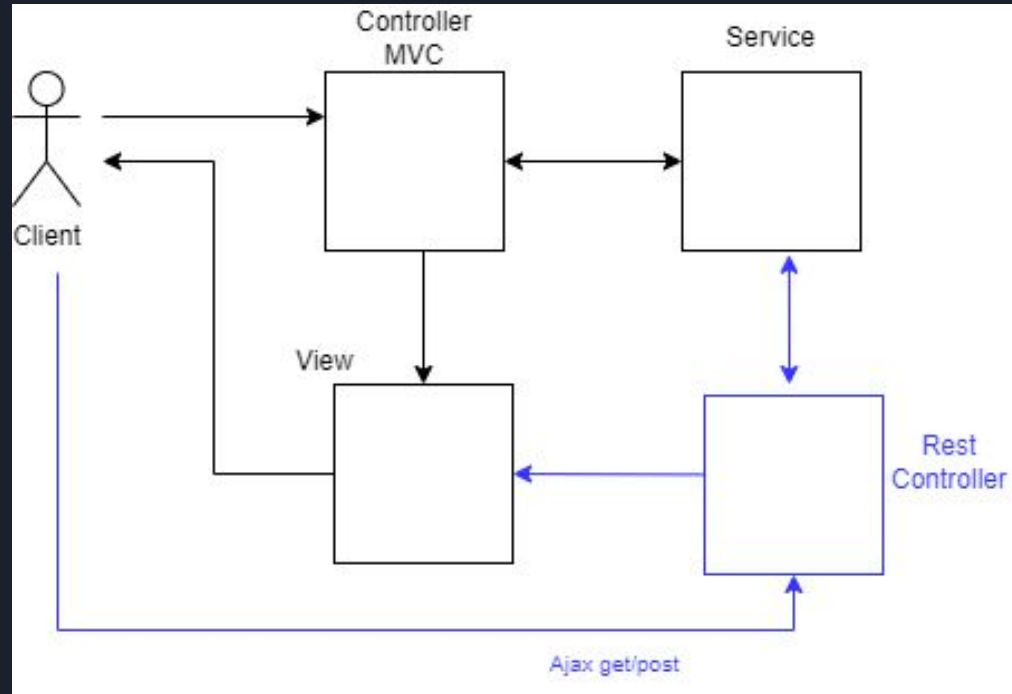


Spring

Adottare Spring consente di creare diversi Bean da riutilizzare. Questo torna utile con le classi dei Service, dei DAO e degli algoritmi.

```
49 @Configuration
50 @ComponentScan(basePackages="com.corso")
51 @EnableTransactionManagement
52 public class Beans {
53
54     @Bean(name="matchDao")
55     public MatchDAO getMatchDAO() {
56         return new MatchDAO();
57     }
58
59     @Bean(name="algorithmDao")
60     public AlgorithmDAO getAlgorithmDAO() {
61         return new AlgorithmDAO();
62     }
63
64     @Bean(name="userDao")
65     public UserDAO getUserDAO() {
66         return new UserDAO();
67     }
68 }
```

Struttura del FrontEnd



SpringMVC e Rest controller

SpringMVC viene utilizzato dai Controller per restituire le corrette pagine (JSP) quando invocate. I Rest Controller fanno comunicare le pagine con i servizi in modo asincrono.

```
20 @Controller
21 @RequestMapping("/")
22 public class AdminController {
23
24     @Autowired
25     MatchService matchService;
26
27     @Autowired
28     UserService userService;
29
30     @Autowired
31     SegnalazioneService segnalazioneService;
32
33     @Autowired
34     AlgorithmService algorithmService;
35
36     @Autowired
37     StandardWordService swService;
38
39     @Autowired
40     AlgorithmHandler ah;
41
42     @LoginLog
43     @GetMapping("/home_admin")
44     public String getHomeAdmin(Model m, HttpSession s) {
45         m.addAttribute("user", s.getAttribute("user"));
46         return "homeAdmin";
47     }
48 }
```

```
27 @RestController
28 @RequestMapping("/")
29 public class AdminRestController {
30
31     @Autowired
32     MatchService matchService;
33
34     @Autowired
35     UserService userService;
36
37     @Autowired
38     SegnalazioneService segnalazioneService;
39
40     @Autowired
41     AlgorithmService algorithmService;
42
43     @Autowired
44     StandardWordService swService;
45
46     @Autowired
47     AlgorithmHandler ah;
48
49     @GetMapping(path={"/matchlist"}, produces= {MediaType.APPLICATION_JSON_VALUE})
50     public List<Match> getMatchList() {
51         List<Match> matchList = matchService.getAllUncheckedMatches();
52         return matchList;
53     }
54 }
```

Un'applicazione **sicura**

Gestione della sessione utente e controllo delle autorizzazioni tramite Filter



Non hai ancora eseguito l'accesso oppure la sessione è scaduta



Non sei autorizzato ad accedere a questa risorsa



Accesso

Username

user1

Password

.....

Password errata


Login

[Non sei ancora registrato? Registrati](#)

Password salvate nel database
con crittografia SHA-256

Validazione delle form

Lato Client (Javascript)



435?^

Indovina

Non puoi inserire numeri o simboli speciali

Lato Server (Spring)

```
@NotEmpty
private String username;

@NotEmpty
@Pattern(regexp = "^(?=.*[0-9])(?=.*[a-z]).{5,}$", message="minimo 5 caratteri, almeno una lettera e un numero")
private String password;

@NotEmpty
@Pattern(regexp = "^(?=.*[0-9])(?=.*[a-z]).{5,}$", message="minimo 5 caratteri, almeno una lettera e un numero")
private String confirmPassword;
```

JavaScript e JQuery

JavaScript e JQuery vengono utilizzati per aggiornare la pagine e le informazioni in modo dinamico, facendo anche uso di chiamate asincrone.

```
$.get("matchlist", function(responseJson) {
  var $ul = $("#matchlist");
  $ul.find("li").remove();
  if(isJsonEmpty(responseJson))
  {
    var $li = $("- ").val("nullo").text("Nessun Match da Confermare").appendTo($ul);
    $li.addClass('list-group-item d-flex justify-content-between align-items-center match-item');
  }
  $.each(responseJson, function(index, match) {
    var $li = $("- ").html(match.input + " &arr; " + match.standardword).appendTo($ul);
    $li.addClass('list-group-item d-flex justify-content-between align-items-center match-item');
    var $button1 = $("").val(match.id + "ba").text("Conferma").appendTo($li);
    $button1.addClass('btn btn-success btn-sm accept-button tablebtn');
    $button1.click(function() {
      var param = {matchid : match.id};
      $.post("checkmatch", $.param(param));
      $li.remove();
    });

    var $button3 = $("").text("Correggi").appendTo($li);
    $button3.addClass('btn btn-success incredibili btn-sm modify-button tablebtn');
    $button3.click(function() {
      showConfirmationModalForMatch(match.input);
    });

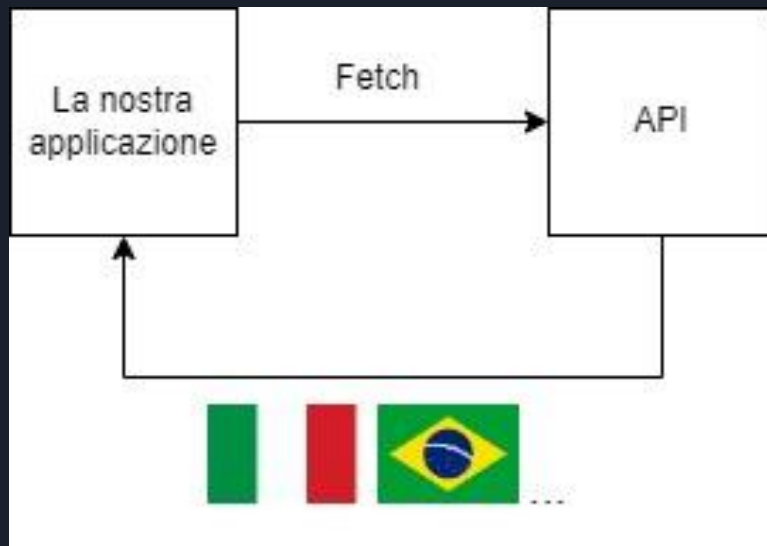
    var $button2 = $("").text("Elimina").appendTo($li);
    $button2.addClass('btn btn-danger btn-sm reject-button ');
    $button2.click(function() {
      var param = {matchid : match.id};
      $.post("removematch", $.param(param));
      $li.remove();
    });
  });
});

```

```
function showConfirmationModalForMatch(input) {
  matchInputInModal = input;
  const correctAnswerElement = document.getElementById("inputMatch");
  correctAnswerElement.textContent = "Scegliere il match per l'input: " + matchInputInModal;
  $(document).on("click", "#modalMatchConfirm", confermaMatch);
  $('#matchModal').modal('show');
}
```


REST Countries API

Come abbiamo ottenuto le bandiere dei paesi? Abbiamo utilizzato le API di <https://restcountries.com/v3.1/all>



Bootstrap

Utilizzo di Bootstrap per ottenere strutture e bottoni pronti da utilizzare.

Benvenuto Admin

Classifica giocatori

| | |
|-----------|-----|
| 1. pietro | 21% |
| 2. paolo | 18% |
| 3. marco | 0% |

Classifica algoritmi

| | |
|-----------------|-----|
| 1. Fisher | |
| 2. Levenshtein2 | |
| 3. Levenshtein1 | 71% |
| 4. Contained | 60% |
| 5. Contained | 55% |

Elenco dei match

| | | | |
|----------------------|----------|----------|---------|
| Egitto → Egypt | Correggi | Conferma | Elimina |
| vnsfdokx → null | Correggi | Conferma | Elimina |
| Ec uador → Ecuador | Correggi | Conferma | Elimina |
| arabia saudit → null | Correggi | Conferma | Elimina |

Lista dei giocatori

| | |
|--------|-------|
| paolo | Banna |
| pietro | Banna |
| marco | Banna |

```
23<div class="container mt-4">
24<div class="row">
25<div class="col-lg-4 col-md-6">
26<div class="card mb-4">
27<div class="title-and-load">
28<h2 class="card-title">Classifica giocatori</h2>
29<button id="buttonLoadRanking" type="button" class="btn btn-primary Load-btn">&#8635;</button>
30</div>
31<div class="card-body scrollable-card">
32<ul class="list-group match-list" id="ranking">
33</ul>
34</div>
35</div>
36</div>
```



Lavoro Agile - Metodologia SCRUM

Product Backlog



Individuare i task e ordinarli in base alla priorità

Suddivisione del lavoro in **Sprint**



Assegnazione dei task ai membri del team

Daily Meeting



Aggiornamento dei task e aiuto reciproco

Jira Software



Come abbiamo gestito i task? Abbiamo usato Jira.

| DA COMPLETARE 4 | IN CORSO 3 | COMPLETATO 22 ✓ |
|---|---|--|
| <div>classifica algoritmi backend</div> <div>✓ TI-27</div> <div></div> | <div>aggiungere sistema di logging</div> <div>✓ TI-33</div> <div>GG</div> | <div>interfaccia gioco</div> <div>✓ TI-13</div> <div>✓ IC</div> |
| <div>classifica algoritmi ajax</div> <div>✓ TI-26</div> <div></div> | <div>navbar utente username</div> <div>✓ TI-29</div> <div>SM</div> | <div>addestrare algoritmi</div> <div>✓ TI-7</div> <div>✓ GG</div> |
| <div>pagina principale carosello</div> <div>✓ TI-30</div> <div>SM</div> | <div>controllare che gli utenti non siano bannati durante il login</div> <div>✓ TI-37</div> <div></div> | <div>registrazione in controller</div> <div>✓ TI-12</div> <div>✓ G</div> |



GIT e GitHub

Per tenere traccia dei cambiamenti nel codice abbiamo usato GIT e per condividerlo tra i membri del team GitHub. Questi sono i comandi che abbiamo utilizzato:

`git clone`

`git push`

`git merge`

`git pull`

`git checkout`

`git add`

`git commit`

`git branch`



Criticità

Utilizzo di strumenti di sviluppo diversi tra i membri del team



Decisione di avere tutti gli stessi strumenti di sviluppo per poterci aiutare al meglio nel caso di problemi

Suddivisione iniziale del lavoro: 2 di noi per il backend e 2 per il frontend



Inversione dei ruoli così che tutti prendessero dimestichezza con tutto il progetto

Problemi nella logica del gioco gestita lato frontend



Abbiamo deciso di gestirla lato backend e passare al frontend solo i dati necessari



Grazie per l'attenzione!