



Portale FIP

Lo scopo di questo documento è quello di descrivere in breve il progetto “Portale FIP”, fatto da me Marco Morini per l’esame di Tecnologie Web della Laurea in Informatica dell’Università degli Studi di Modena, tenuto dalla professoressa Claudia Canali e dal professor Nicola Capodieci.

Verrà proposta prima di tutto una panoramica del documento, per poi descrivere meglio i principali requisiti iniziali pensati da me e rivisti con la professoressa Canali, con anche l’uso di qualche diagramma UML(Unified Modeling Language). In seguito si analizzeranno le principali scelte dietro all’applicazione web, la struttura logica del progetto ed i test eseguiti. Infine una piccola parte sui principali problemi riscontrati ed eventuali scelte iniziali che avrei probabilmente preferito fare. Si ricorda infatti che l’esame ha proprio lo scopo di insegnare agli studenti a mettersi in gioco e creare in completa autonomia un’applicazione Web, proprio per questo mi è talvolta capitato di capire a pieno alcuni concetti chiave solo in un secondo momento.

Home Page:



Indice

1. Introduzione:	3
2. Panoramica generale e tecnologie utilizzate	3
3. Specifica dei Requisiti	4
4.Struttura codice interna:	4
4.1 Class Diagram	5
4.2 Use Case Diagram:	5
5. Immagini Sito:	6
5.1 Home Page:	6
5.2 Login, Registrazione e Logout	7
5.2.1: Login	7
5.2.2: Logout	7
5.2.3: Registrazione	8
5.3 Dettagli Campionato	8
5.4 Ricerca Giocatore	9
5.5 Dettagli Squadra	9
5.6 Dettagli Giocatore:	10
5.7 DashBoard Admin	11
5.7.1 Gestione Partite	11
5.7.2 Gestione Tabelle	11
6 Scelte fatte:	12
7 Principali difficoltà superate	13
8 Tests:	13
8.0 Setup	13
8.1 Test Partite	15
8.2 Controllo commenti, like, dislike	16
8.3 Controllo della dashboard amministratore	16
9 Conclusioni Finali:	17

1. Introduzione:

“Portale FIP”, il nome del mio progetto, è un’applicazione web per la gestione e visione dei risultati delle partite dei principali campionati italiani di pallacanestro; FIP infatti è l’acronimo di Federazione Italiana Pallacanestro. Potrà quindi essere usata dalla Federazione Italiana Pallacanestro per comunicare le partite alle squadre e ai giocatori e allo stesso tempo una qualsiasi persona potrà consultare i risultati delle partite giocate. Attraverso la registrazione al sito, l’utente potrà inserire i tabellini delle partite che è andato a vedere, che quindi conosce il risultato, in modo da rendere il sito aggiornato nel minor tempo possibile.

2. Panoramica generale e tecnologie utilizzate

Il progetto è stato creato tramite l’utilizzo del framework Django, un framework python per la creazione di applicazioni Web dinamiche. L’utilizzo di Django è stato chiaramente cruciale per lo sviluppo dell’applicazione, in quanto il framework si prende carico di una grossa parte dello sviluppo Web per lasciare agli utilizzatori molto più spazio e tempo per l’effettiva implementazione del progetto.

Tutta la parte di gestione dei database e delle tabelle è infatti affidata a Django insieme ad sqlite3. Django fornisce un’interfaccia per comunicare con i database estremamente semplice e intuitiva, senza doversi mai preoccupare dell’effettiva implementazione in linguaggio SQL. Infatti Django è stato pressoché fondamentale, in quanto mi ha permesso di lavorare con molta fluidità senza dover continuamente modificare codice verboso SQL. In particolare l’uso delle reverse query ha avuto un ruolo decisivo, senza cui il progetto avrebbe sicuramente avuto dimensioni ben minori.

Insieme a Django ho deciso di utilizzare ampiamente bootstrap per migliorare di gran lunga l’aspetto delle pagine del sito. Insieme poi a strumenti come JavaScript(Jquery, Ajax, ecc....) ho reso l’applicazione più dinamica e interattiva, con talvolta l’utilizzo di qualche stile CSS particolare.

Un altro ruolo fondamentale, Django l’ha avuto nella presentazione dei template. Infatti grazie a DTL(Django Template Language) è stato molto più semplice “passare” informazioni dalle “views” ai template e riuscire a reperire URL e risorse statiche dal progetto.

L’IDE che ho preferito utilizzare è PyCharm, di JetBrains, insieme a pipenv per creare l’ambiente virtuale attorno al progetto.

3. Specifica dei Requisiti

I requisiti principali del progetto sono riportati qui sotto:

- Pagina home di benvenuto al sito per navigare tra le diverse pagine.
- Possibilità di avere dettagli su squadre e giocatori, cercandoli per nome.
- Possibilità di lettura, inserimento e modifica di tutte le informazioni che circondano una partita di pallacanestro(come luogo, data, punteggio, statistiche dei giocatori, ecc....)
- Tutta la logica di gestione di una stagione, con tanto di classifica costantemente aggiornata in base al risultato delle partite
- Calcolo di statistiche della stagione di un giocatore, come punti/rimbalzi/assist totali e a partita e visualizzazione di questi dati attraverso tabelle e grafici
- Possibilità da parte di utenti registrati di lasciare commenti sotto le partite e inserire like/dislike ai commenti.
- Possibilità di inserire immagini profilo di squadra e del singolo giocatore.*
- Dashboard di amministrazione per rendere la gestione del database più facile ed intuitiva.*

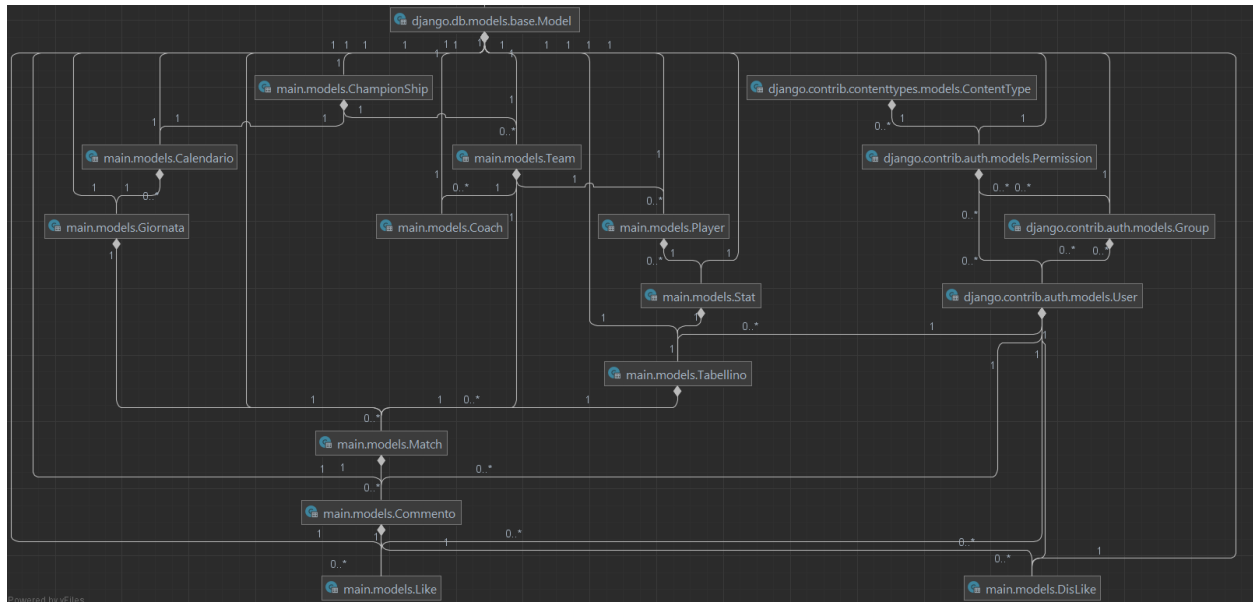
(Le funzionalità descritte nei requisiti che finiscono per * sono disponibili solo agli utenti amministratori)

4.Struttura codice interna:

Iniziamo ora a vedere gli aspetti interni del progetto. Si mostrano qui un paio di diagrammi UML per facilitare la comprensione della struttura interna del progetto e delle funzionalità che esso offre all'utente.

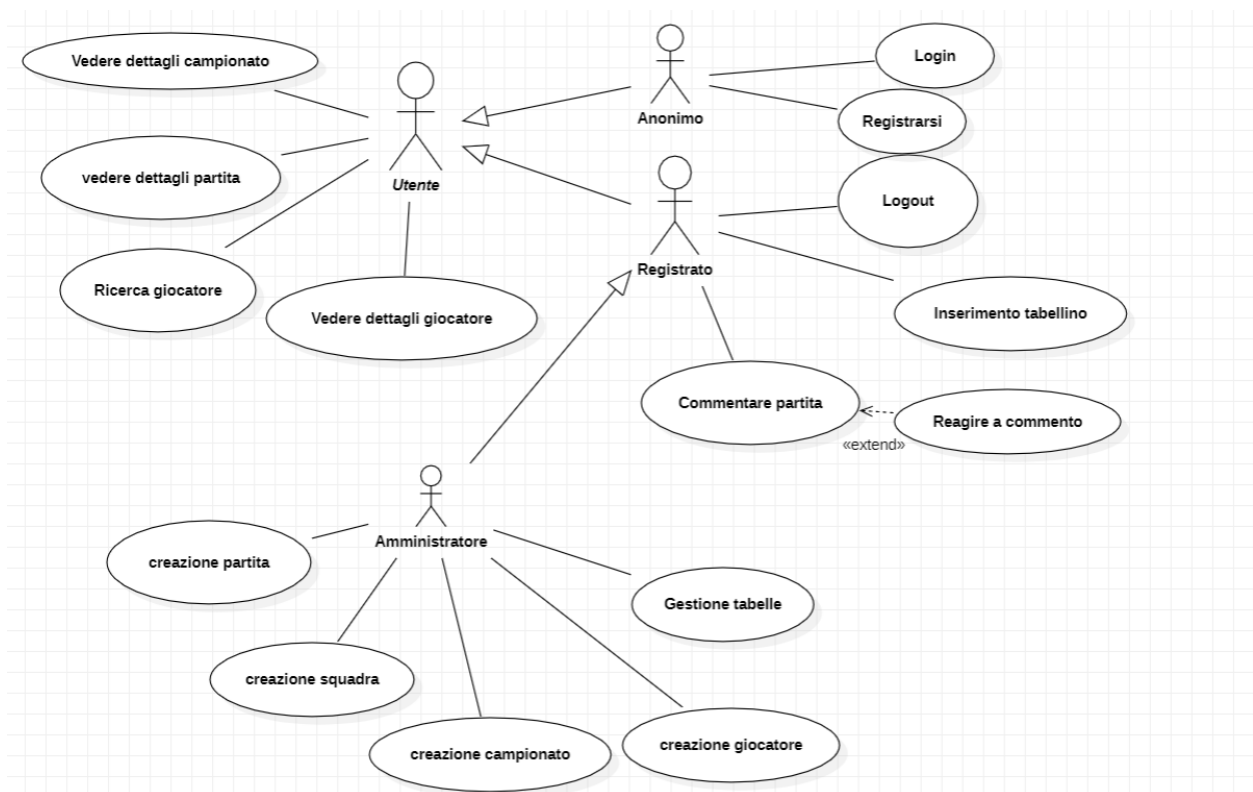
Qui segue il diagramma delle classi, o **Class Diagram**, dei modelli dell'applicazione:

4.1 Class Diagram



4.2 Use Case Diagram:

(l'attore Utente è astratto, è infatti scritto in italico ma è difficile notarlo)



5. Immagini Sito:

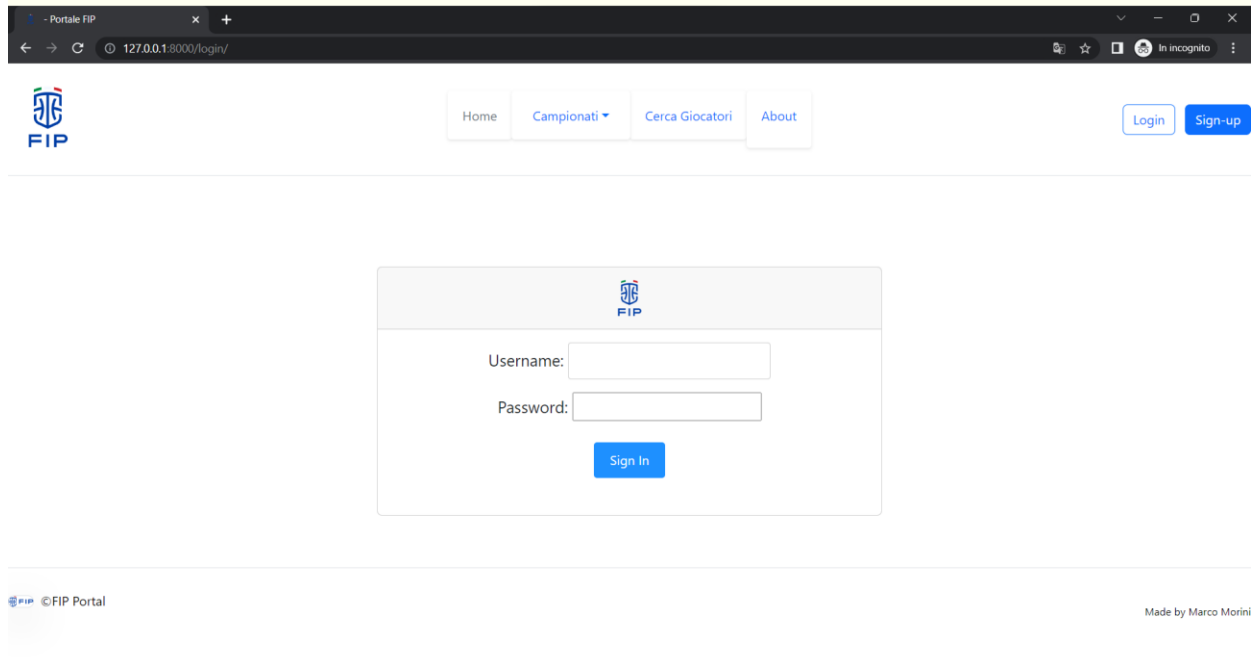
Si riportano di seguito le principali pagine consultabili sul sito.

5.1 Home Page:



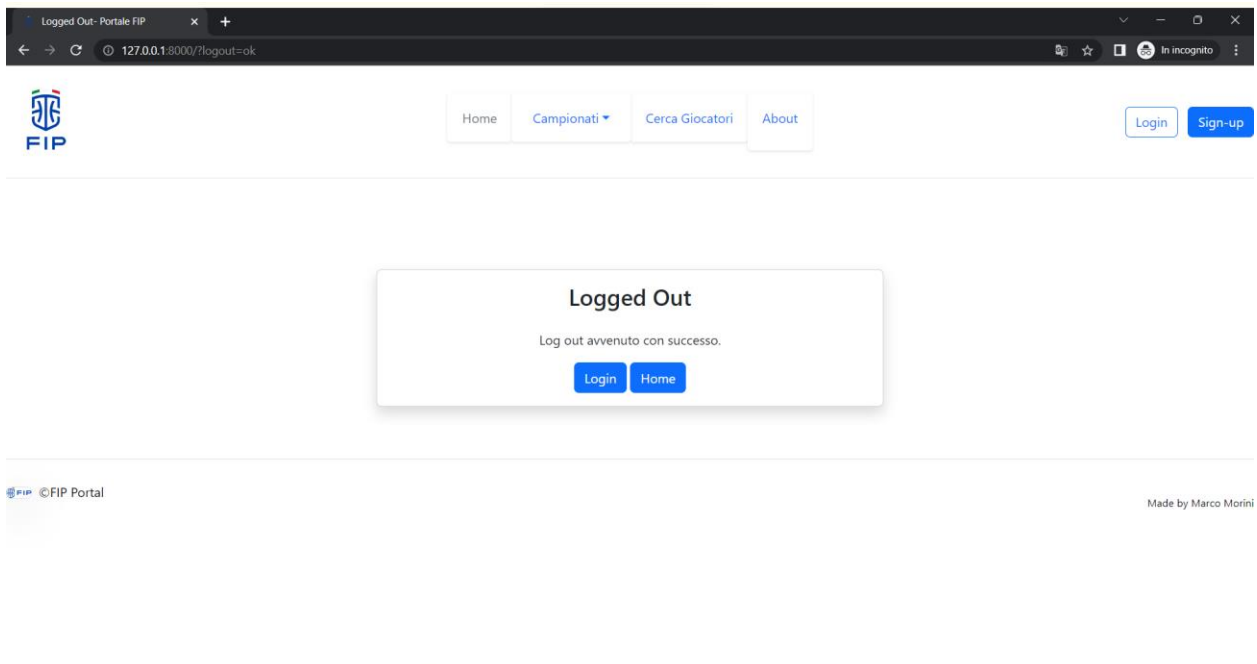
5.2 Login, Registrazione e Logout

5.2.1: Login



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/login/". The page features the FIP logo on the left and a navigation menu with "Home", "Campionati", "Cerca Giocatori", and "About" in the center. On the right, there are "Login" and "Sign-up" buttons. The main content area contains a login form with a header "FIP" and fields for "Username:" and "Password:". Below these fields is a blue "Sign In" button. The footer includes the text "©FIP Portal" on the left and "Made by Marco Morini" on the right.

5.2.2: Logout



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/?logout=ok". The page layout is consistent with the login page, featuring the FIP logo, navigation menu, and login/sign-up buttons. The main content area displays a "Logged Out" message with the text "Log out avvenuto con successo." and two buttons: "Login" and "Home". The footer remains the same, with "©FIP Portal" on the left and "Made by Marco Morini" on the right.

5.2.3: Registrazione

Portale FIP

Home Campionati Cerca Giocatori About Login Sign-up

Sign Up

Username: Required. 150 characters or fewer. Letters, digits and @/+/-/ only.

Password:

Your password can't be too similar to your other personal information.

Your password must contain at least 8 characters.

Your password can't be a commonly used password.

Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

Sign Up

5.3 Dettagli Campionato

Serie C - Portale FIP

Serie C

Classifica

Pos	Squadra	Vittorie	Sconfitte	Punti
1	Pallacanestro Novellara	3	2	6
2	CVD Casalecchio	1	0	2
3	Veni Basket	1	0	2
4	4 Torri Ferrara	0	1	0
5	Angels Santarcangelo	0	1	0
6	BSL San Lazzaro	0	1	0
7	Basket Lugo	0	0	0
8	Gaetano Scirea	0	1	0
9	Grifo Imola	0	1	0
10	Magik Parma	0	1	0
11	Omega Basket	0	1	0
12	Pallacanestro Correggio	0	2	0
13	Pallacanestro Scandiano	0	0	0
14	ReBasket	0	1	0
15	Scuola Basket Ferrara	0	1	0
16	Virtus Medicina	0	1	0

1ª Giornata

Grifo Imola di Imola - Grifo Oro 0 - 0 Angels Santarcangelo di Santarcangelo - Angels

Vedi dettagli

5.4 Ricerca Giocatore

Portale FIP

127.0.0.1:8000/players/search/

YouTube traduttore WhatsApp Netflix adsettings I Semestre II Semestre GitHub Gmail Traduci Make a PNG Transp... Python Tutor - Visu... Disney

FIP

Home Campionati Cerca Giocatori About

Login Sign-up

Ricerca Giocatore

Cognome: Nome: Campionato: Tutti i Campionati Squadra: Tutte le Squadre

Cerca

Giocatori Trovati

[Amadio Riccardo #16-ReBasket](#)
[Amadori Alessandro #10-Veni Basket](#)
[Astani Alessandro #1-4 Torri Ferrara](#)
[Astolfi Massimo #23-CVD Casalecchio](#)
[Bertolini Riccardo #17-ReBasket](#)
[Bovio bho #18-ReBasket](#)
[Cevoli Alessandro #8-Gaetano Scirea](#)
[Cinciarini Alessandro #3-OlimpiaMilano](#)
[Cinciarini Alessandro #12-Pallacanestro Reggiana](#)


5.5 Dettagli Squadra

Dettagli - Portale FIP

Morini Marco #6 - Portale FIP

127.0.0.1:8000/teams/103/

YouTube traduttore WhatsApp Netflix adsettings I Semestre II Semestre GitHub Gmail Traduci Make a PNG Transp... Python Tutor - Visu... Disney



Pallacanestro Novellara
Città: Novellara
Sponsor: Max Devil
Staff tecnico: Guido Boni; Giancarlo Mantovani; Paolo Nallin;
Campionato: [Serie C](#)
Giocatori:
[Ferrari Nicolo #4](#)
[Ferrari Tommaso #11](#)
[Folloni Nicola #7](#)

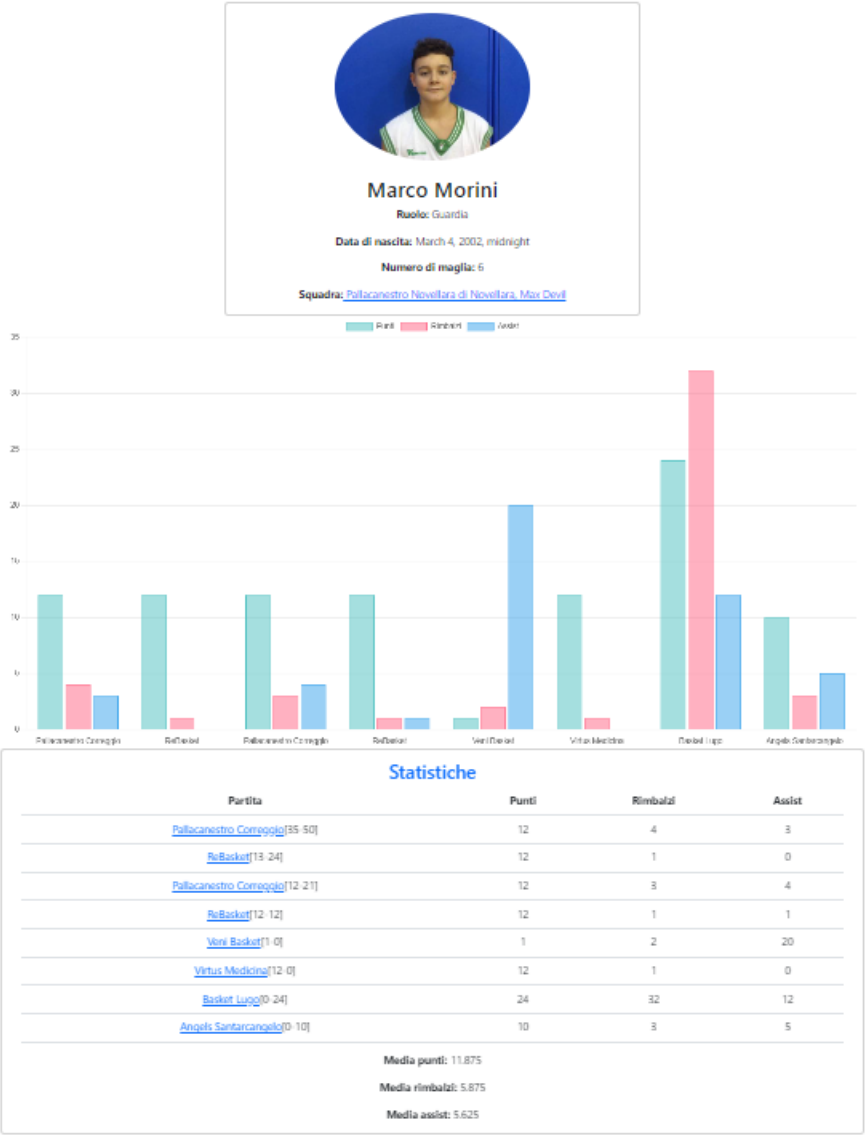
Partite della stagione:

Data	Avversario	Risultato	Dettagli Partita
03/10/2022	Veni Basket L	26-43	i
16/10/2022	Gaetano Scirea W	50-31	i
16/10/2022	Virtus Medicina -	70-0	i
22/10/2022	ReBasket W	81-35	i
30/10/2022	Pallacanestro Correggio -	2-0	i
06/11/2022	Magik Parma W	23-22	i
13/11/2022	CVD Casalecchio L	4-15	i
19/11/2022	Angels Santarcangelo -	25-0	i
21/11/2022	4 Torri Ferrara -	10-0	i
27/11/2022	Grifo Imola -	11-0	i
04/12/2022	Omega Basket -	14-0	i
10/12/2022	BSL San Lazzaro -	20-0	i
18/12/2022	Scuola Basket Ferrara -	12-0	i

Statistiche squadra:

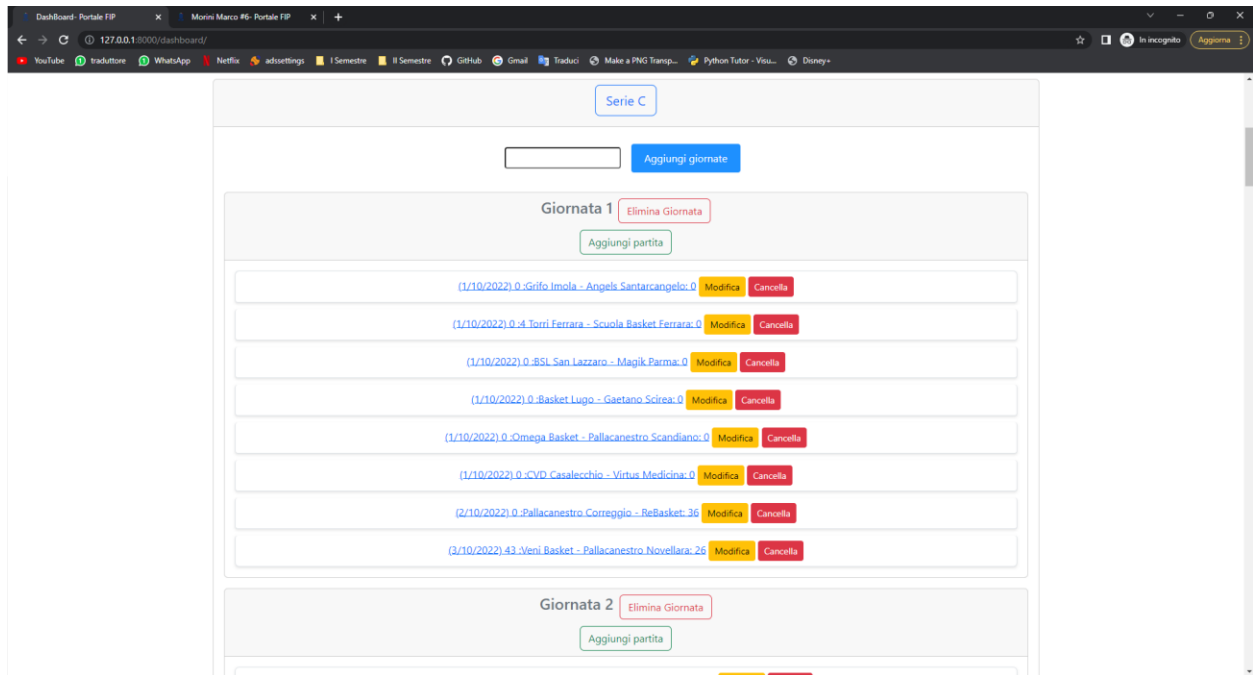
Media punti segnati: 26.769
Media punti subiti: 11.23
Miglior marcatore: Morini Marco #6
Miglior rimbalzista: Morini Marco #6

5.6 Dettagli Giocatore:

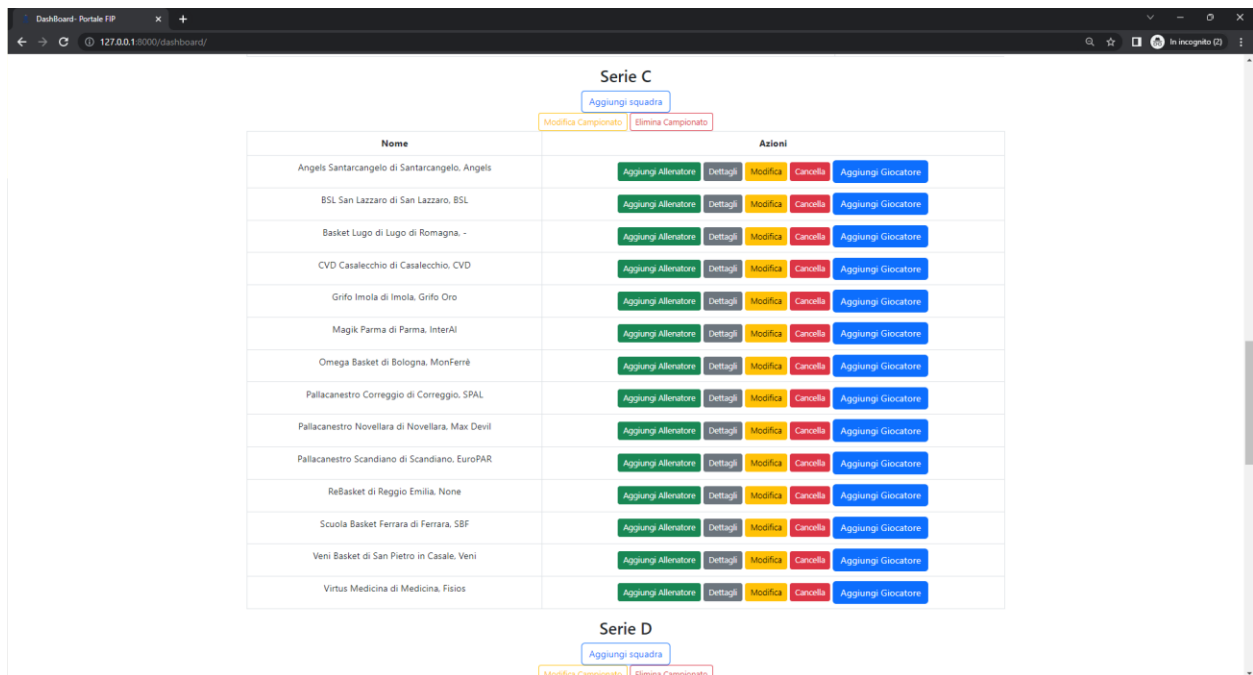


5.7 DashBoard Admin

5.7.1 Gestione Partite



5.7.2 Gestione Tabelle



6 Scelte fatte:

Vista la numerosa quantità di tabelle e quindi di entries nei database si è spesso deciso di mostrare all'utente soluzioni compatte espandibili attraverso filtri inseriti dall'utente stesso. Un perfetto esempio è la ricerca dei giocatori, dove grazie ad uno script JavaScript una volta scelto il campionato di appartenenza vengono mostrati solo le squadre appartenenti a quel campionato. Un altro esempio è l'uso della classe collapse nella dashboard per poter accedere a una qualsiasi partita molto più rapidamente. In ogni caso in ogni punto del sito se si ha davanti una lista di oggetti che siano partite, squadre, giocatori, ecc.... saranno sempre in ordine o alfabetico o cronologico se è un evento con un attributo data. Si è inoltre cercato di non appesantire troppo i controlli sull'input per tutto quello che riguarda la parte amministratore, in quanto sono pagine non destinate ad un utente qualunque.

Infine un'ultima scelta fatta in fase di progetto è quella di permettere agli utenti di inserire un tabellino. Su questo ho dovuto pensare a fondo, in quanto se un qualsiasi utente registrato potesse inserire tabellini ovunque potrebbe potenzialmente rovinare l'intero database di un campionato. Si è infatti deciso di poter inserire un tabellino solo quando non è già presente un tabellino creato da un altro utente, in modo che l'utente "malintenzionato" non possa modificare statistiche vecchie. Quindi solo un utente amministratore potrà cambiare/eliminare un tabellino di una partita.

Ho inoltre deciso di utilizzare solo due applicazioni all'interno del progetto: "main" e "users". Il principale motivo di questa scelta è che tutti i modelli sono a strettissimo contatto tra loro, si hanno costantemente dipendenze "uscenti" verso altri modelli. Per questo ho deciso di tenere tutto molto compatto nonostante le dimensioni siano cresciute, dando priorità alla coesione delle componenti interne, piuttosto che separarli senza un valido motivo. Main è dove la maggior parte dell'applicazione si concentra, dove tutti i modelli e le views che ne fanno riferimento stanno. L'applicazione users è invece una piccola applicazione per la gestione degli utenti nel sistema; piccola in quanto Django tra le infinite cose che ci dà già fatte, ha anche tutta la logica di gestione autenticazione già implementata.

7 Principali difficoltà superate

Come anticipato all'inizio, il progetto è il fulcro dell'esame, lo scopo dell'esame è infatti quello di implementare un'applicazione Web dinamica. Ritengo infatti di aver imparato parecchio da questo progetto, mi ha dato la possibilità di mettermi veramente alla prova. Ho capito meglio cosa vuol dire gestire un progetto di dimensioni un minimo importanti dove più componenti devono costantemente interagire tra loro perché il tutto funzioni.

Le parti più difficili penso siano state in prima battuta la confidenza con i linguaggi come DTL, html e javascript che erano a me pressoché nuovi. Successivamente poi una delle difficoltà maggiori secondo me è stato far interagire agilmente tutti i diversi modelli tra loro. Ci sono template che racchiudono informazioni provenienti da più di una decina di modelli per presentare al meglio il risultato.

8 Tests:

Per assicurarmi di svolgere bene quello che volevo implementare ho inizialmente inserito qualche test per verificare ciò che avrei fatto "in completa autonomia", ovvero quando non usavo strumenti (views) già forniti da Django, quindi dove ero io creare tutto da zero. Se ne riportano un paio di seguito:

8.0 Setup

Per ogni test ho inserito una piccola parte di setUp per avere gli oggetti pronti per essere usati nel test; in particolare creo un utente, due squadre, qualche giocatore per squadra e una partita tra le due squadre.

```
def setUp(self):
    self.user = User.objects.create_user(username='test_1',
password='test_1')

    self.client = Client()

    self.campionato = Championship.objects.create(name='Serie C',
location='Italia', year=2023)
    self.calendario = Calendario.objects.create(championship=self.campionato)
    self.team_novellara = Team.objects.create(name='Pallacanestro Novellara',
main_sponsor='Max Devil',
city='Novellara',
championships=self.campionato, img=File(
open('static/img/team_000296_2023_CS_PallacanestroNovellara.jpg',
'rb')))
    self.team_rebasket = Team.objects.create(name='ReBasket',
main_sponsor='MLSerramenti',
city='Reggio Emilia',
championships=self.campionato, img=File(
```

```

open('static/img/team_022281_2023_CS_RebasketCastelnovoSotto.jpg', 'rb'))

    self.giocatore1 = Player.objects.create(
        name='Marco', last_name='Morini',
        birth_date=timezone.make_aware(timezone.datetime(2002, 3, 4),
timezone.get_default_timezone(), is_dst=None),
        number=6, team=self.team_novellara
    )

    self.giocatore2 = Player.objects.create(
        name='Matteo', last_name='Frediani',
        birth_date=timezone.make_aware(timezone.datetime(2004, 12, 24),
timezone.get_default_timezone(),
                                is_dst=None),
        number=5, team=self.team_novellara
    )

    self.giocatore3 = Player.objects.create(
        name='Riccardo', last_name='Amadio',
        birth_date=timezone.make_aware(timezone.datetime(1999, 11, 11),
timezone.get_default_timezone(),
                                is_dst=None),
        number=11, team=self.team_rebasket
    )

    self.giocatore4 = Player.objects.create(
        name='Gianmarco', last_name='Bertolini',
        birth_date=timezone.make_aware(timezone.datetime(1992, 1, 21),
timezone.get_default_timezone(),
                                is_dst=None),
        number=17, team=self.team_rebasket
    )

    self.giornata1 = Giornata.objects.create(num=1,
calendario=self.calendario)
    self.partita = Match.objects.create(date=timezone.now(),
teamA=self.team_novellara, teamB=self.team_rebasket,
                                location='Novellara',
giornata=self.giornata1)

```

8.1 Test Partite

In questo primo test verifico che l'inserimento dei tabellini nel sistema funzioni come si deve, andando a controllare il codice di risposta della richiesta di inserimento di un tabellino e l'effettivo inserimento, con tanto di somma corretta per il calcolo del punteggio finale

```
def test_tabellinoA(self):
    # Effettua il login come utente
    self.client.login(username='test_1', password='test_1')

    # Simula l'azione dell'utente che crea un tabellino per la partita
    tabellino1 = {
        'player1': self.giocatore1.pk, 'points1': 10, 'rebounds1': 12,
        'blocks1': 5,
        'player2': self.giocatore2.pk, 'points2': 12, 'rebounds2': 3,
        'blocks2': 10,
    }
    for i in range(3, 13):
        tabellino1['player' + str(i)] = ''
        tabellino1['points' + str(i)] = ''
        tabellino1['rebounds' + str(i)] = ''
        tabellino1['blocks' + str(i)] = ''

    response = self.client.post(reverse('main:create-tabellinoA',
args=[self.partita.id]), tabellino1)
    # Verifica se la risposta HTTP è corretta (ad esempio, reindirizzamento a
una pagina successiva)
    self.assertEqual(response.status_code, 302)

    partita = Match.objects.get(id=self.partita.id)
    tabellino = Tabellino.objects.get(partitaA=self.partita)

    tabellino_points_sum = sum([stat.points if stat else 0 for stat in
tabellino.get_stats()])

    # Verifico se la somma dei punti nel tabellino è uguale al valore della
variabile points della partita
    self.assertEqual(tabellino_points_sum, partita.pointsA)
    self.assertEqual(22, partita.pointsA)
    # Giocatore 1
    self.assertEqual(tabellino.stat1.player.id, self.giocatore1.id)
    self.assertEqual(tabellino.stat1.points, 10)
    self.assertEqual(tabellino.stat1.rebounds, 12)
    self.assertEqual(tabellino.stat1.blocks, 5)
    # Giocatore 2
    self.assertEqual(tabellino.stat2.player.id, self.giocatore2.id)
    self.assertEqual(tabellino.stat2.points, 12)
    self.assertEqual(tabellino.stat2.rebounds, 3)
    self.assertEqual(tabellino.stat2.blocks, 10)
```

8.2 Controllo commenti, like, dislike

Qui si controlla invece che i commenti vengano inseriti correttamente e che il sistema di like e dislike funzioni correttamente

```
def test_commento_e_like_dislike(self):
    self.client.login(username='test_1', password='test_1')

    response = self.client.post(reverse('main:add-comment',
args=[self.partita.id]), {'content': 'Test Commenti'})

    # Verifica se la risposta HTTP è corretta (ad esempio, reindirizzamento a
una pagina successiva)
    self.assertEqual(response.status_code, 302)

    comment = Commento.objects.get(created_by=self.user.id,
match_id=self.partita.id)

    self.assertEqual('Test Commenti', comment.comment)

    response = self.client.post(reverse('main:like-comment'), {'comment_id':
comment.id})
    self.assertEqual(response.status_code, 200)
    self.assertEqual(comment.likes.count(), 1)

    response = self.client.post(reverse('main:dislike-comment'),
{'comment_id': comment.id})
    self.assertEqual(response.status_code, 200)
    self.assertEqual(comment.likes.count(), 0)
    self.assertEqual(comment.dislikes.count(), 1)
```

8.3 Controllo della dashboard amministratore

In questi ultimi test invece si controlla il corretto inserimento delle giornate in ordine, e che la cancellazione di un tabellino azzeri il punteggio di una partita correttamente

```
def test_creazione_giornata(self):
    self.client.login(username='admin', password='admin')

    # Creo altre due giornate (la 2 e la 3)
    response = self.client.post(reverse('main:create-giornata',
args=[self.campionato.id])
    self.assertEqual(response.status_code, 302)
    response = self.client.post(reverse('main:create-giornata',
args=[self.campionato.id])
    self.assertEqual(response.status_code, 302)
```



```

giornata2 = Giornata.objects.get(calendario_id=self.calendario.id, num=2)
self.assertIsNotNone(giornata2)
giornata3 = Giornata.objects.get(calendario_id=self.calendario.id, num=3)
self.assertIsNotNone(giornata3)

# Cancello la prima
giornata2.delete()
response = self.client.post(reverse('main:create-giornata',
args=[self.campionato.id]))
self.assertEqual(response.status_code, 302)
giornata2 = Giornata.objects.get(calendario_id=self.calendario.id, num=2)
self.assertIsNotNone(giornata2)

self.assertFalse(Giornata.objects.filter(calendario_id=self.calendario.id,
num=4).exists())

def test_cancellazione_tabellino(self):
    self.client.login(username='admin', password='admin')

    response = self.client.post(reverse('main:delete-tabellino',
args=[self.tabellinoA.id]))
    self.assertEqual(response.status_code, 302)

    self.assertEqual(self.partita.pointsA, 0)
    self.assertIsNotNone(self.tabellinoA)

```

9 Conclusioni Finali:

Per maggiori dettagli, il repository GitHub del progetto è recuperabile al seguente link:

<https://github.com/Morri02/basketBallWebSite.git>, con anche qualche istruzione per l'installazione se si volesse effettivamente provare il sito.