|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Лука Мандић

**Наслов**

Дипломски рад

- Основне академске студије -

Нови Сад, 2024.

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА** | Лист: |
| 1/1 |

*(Податке уноси предметни наставник - ментор)*

| Врста студија: | **Основне академске студије** |
| --- | --- |
| Студијски програм: | **Рачунарство и аутоматика** |
| Руководилац студијског програма: | **проф. др Милан Рапаић (РА)** |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | **Лука Мандић** | Број индекса: | **RA 101/2019** |
| Област: | **Електротехничко и рачунарско инжењерство** | | |
| Ментор: | **Др Жељко Вуковић, доцент** | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:**

|  |
| --- |
| **Систем за** |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
| 1. Анализирати стање у области.  2. Израдити спецификацију захтева софтверског решења.  3. Израдити спецификацију дизајна софтверског решења.  4. Имплементирати софтверско решење према израђеној спецификацији.  5. Тестирати имплементирано софтверско решење.  6. Документовати (1), (2), (3), (4) и (5). |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за:- Студента;  - Ментора |

# КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

|  |  |
| --- | --- |
| Редни број, **РБР**: |  |
| Идентификациони број, **ИБР**: |  |
| Тип документације, **ТД**: | монографска публикација |
| Тип записа, **ТЗ**: | текстуални штампани документ |
| Врста рада, **ВР**: | дипломски рад |
| Аутор, **АУ**: | Лука Мандић |
| Ментор, **МН**: | др Жељко Вуковић, доцент |
| Наслов рада, **НР**: | Систем за |
| Језик публикације, **ЈП**: | српски |
| Језик извода, **ЈИ**: | српски / енглески |
| Земља публиковања, **ЗП**: | Србија |
| Уже географско подручје, **УГП**: | Војводина |
| Година, **ГО**: | 2023 |
| Издавач, **ИЗ**: | ауторски репринт |
| Место и адреса, **МА**: | Нови Сад, Факултет техничких наука, Трг Доситеја Обрадовића 6 |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | 7 / 77 / 20 / 6 / 23 / 0 / 19 |
| Научна област, **НО**: | Електротехничко и рачунарско инжењерство |
| Научна дисциплина, **НД**: | Софтверско инжењерство |
| Предметна одредница/Кључне речи, **ПО**: | Спорт |
| **УДК** |  |
| Чува се, **ЧУ**: | Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад |
| Важна напомена, **ВН**: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Извод, ИЗ: | | У оквиру рада се истражује домен превентивног лечења. Главна мотивација је била жеља за унапређењем квалитета живота а самим тим и здравља људи. Ово се постиже кроз софтверски систем који прати четири аспекта здравља пацијента а то су исхрана, физичка активност, витални знаци и посете лекару. Резултат је систем који ове податке обједињује и пацијенту приказује његово тренутно здравствено стање. | |
| Датум прихватања теме, ДП: | |  | |
| Датум одбране, ДО: | |  | |
| Чланови комисије, **КО**: | Председник: | Др , редовни професор |
| Члан: | др, редовно професор  др Име Презиме, звање | |
|  | Члан, ментор: | др Жељко Вуковић, доцент | |
|  |  | Потпис ментора | |

# KEY WORDS DOCUMENTATION

|  |  |  |  |
| --- | --- | --- | --- |
| Accession number, **ANO**: | |  | |
| Identification number, **INO**: | |  | |
| Document type, **DT**: | | monographic publication | |
| Type of record, **TR**: | | textual material | |
| Contents code, **CC**: | | bachelor thesis | |
| Author, **AU**: | | Luka Mandić | |
| Mentor, **MN**: | | Željko Vuković, assistant professor, PhD | |
| Title, **TI**: | | Software for | |
| Language of text, **LT**: | | Serbian | |
| Language of abstract, **LA**: | | Serbian / English | |
| Country of publication, **CP**: | | Serbia | |
| Locality of publication, **LP**: | | Vojvodina | |
| Publication year, **PY**: | | 2023 | |
| Publisher, **PB**: | | author’s reprint | |
| Publication place, **PP**: | | Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6 | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | | 7 / 77 / 20 / 6 / 23 / 0 / 19 | |
| Scientific field, **SF**: | | Electrotechnical and Computer Engineering | |
| Scientific discipline, **SD**: | | Software Engineering | |
| Subject/Key words, **S**/**KW**: | | Sport | |
| **UC** | |  | |
| Holding data, **HD**: | | Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad | |
| Note, **N**: | |  | |
| Abstract, **AB**: | | The paper focuses on the domain of preventive healthcare. The main motivation behind this was the desire to improve the quality of life and, consequently, the health of individuals. This is achieved through a software system that monitors four aspects of a patient's health, namely diet, physical activity, vital signs, and doctor visits. The result is a system that unifies this data and presents the patient with their current health status. | |
| Accepted by the Scientific Board on, **ASB**: | |  | |
| Defended on, **DE**: | |  | |
| Defended Board, **DB**: | President: | PhD, full professor |
| Member: | PhD, full professor  др Име Презиме, звање | |
|  | Member, Mentor: | PhD Željko Vuković, assistant professor | |
|  |  | Mentor’s signature | |

**SADRŽAJ**

[КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА 5](#_Toc164879258)

[KEY WORDS DOCUMENTATION 7](#_Toc164879259)

[1. УВОД 11](#_Toc164879260)

[1.1 Дефиниција проблемa 11](#_Toc164879261)

[1.2 Преглед софтверског решења 12](#_Toc164879262)

[1.3 Јединственост софтверског решења 12](#_Toc164879263)

[1.4 Организација рада 12](#_Toc164879264)

[2. ПРЕГЛЕД СЛИЧНИХ СИСТЕМА 15](#_Toc164879265)

[3. КОРИШЋЕНЕ СОФТВЕРСКЕ ТЕХНОЛОГИЈЕ 17](#_Toc164879266)

[3.1 Серверске технологије 17](#_Toc164879267)

[3.1.1 .NET 6 17](#_Toc164879268)

[3.1.2 MongoDB 18](#_Toc164879269)

[3.1.3 MongoDB Driver 18](#_Toc164879270)

[3.2 Клијентске технологије 19](#_Toc164879271)

[3.2.1 Angular Framework 19](#_Toc164879272)

[3.2.2 SweetAlert2 19](#_Toc164879273)

[4. СПЕЦИФИКАЦИЈА 21](#_Toc164879274)

[4.1 Спецификација захтева 21](#_Toc164879275)

[4.1.1 Функционални захтеви 21](#_Toc164879276)

[4.1.2 Нефункционални захтеви 29](#_Toc164879277)

[4.2 Спецификација система 31](#_Toc164879278)

[4.2.1 Модел података 31](#_Toc164879279)

[4.2.2 Дијаграми секвенци 35](#_Toc164879280)

[4.2.3 Архитектура система 37](#_Toc164879281)

[5. ИМПЛЕМЕНТАЦИЈА 39](#_Toc164879282)

[5.1 Конфигурација система 39](#_Toc164879283)

[5.2 Имплементација функционалних захтева 41](#_Toc164879284)

[5.2.1 Креирање Player класе 42](#_Toc164879285)

[5.3 Добављање састава тимова 44](#_Toc164879286)

[5.3.1 Ажурирање Match класе 46](#_Toc164879287)

[6. ДЕМОНСТРАЦИЈА 49](#_Toc164879288)

[6.1 Модул за преглед профила играча 50](#_Toc164879289)

[6.2 Модул за преглед мечева 51](#_Toc164879290)

[6.3 Модул за преглед статистике 53](#_Toc164879291)

[6.4 Модул за праћење новости 54](#_Toc164879292)

[7. ЗАКЉУЧАК 57](#_Toc164879293)

[ЛИТЕРАТУРА 59](#_Toc164879294)

[БИОГРАФИЈА 61](#_Toc164879295)

# УВОД

У савременом дигиталном окружењу веб сајт представља незаменљиву основу за повезивање фудбалског клуба са својим фановима широм света. Кроз добро дизајниран и редовно ажуриран веб сајт, фудбалски клуб може пренети кључне информације о распореду утакмица, профилима играча, вестима о тиму и детаљима о продаји улазница, негујући осећај транспарентности и доступности. Осим тога, веб сајт представља кључно маркетиншко средство, привлачећи спонзоре и потенцијалне инвеститоре, док истовремено пружа платформу за продају спортске опреме. Сајт делује као виртуални стадион, омогућавајући навијачима широм света да учествују у узбуђењу игре, приступе ексклузивном садржају и буду чланови онлајн заједница.

## Дефиниција проблемa

Овај рад се фокусирао на решавање четири главна проблема, те га сачињавају четири кључна модула:

**Модул за Преглед Профила Играча** пружа корисницима увид у читав састав екипе. Он омогућава једноставан преглед основне биографије сваког појединачног фудбалера, као и његове статистике кроз године и такмичења. Администратору система је омогућено додавање нових, избацивање старих и измена постојећих играча.

**Модул за Преглед Мечева** пружа корисницима увид у распоред предстојећих утакмица, као и преглед резултата тренутне и претходних сезона. За сваку одиграну утакмицу је могуће видети саставе тимова и детаљно описана дешавања током меча. Осим тога корисници имају на располагању увид у стање на табели. Администратор система има слободу да управља табелама, мечевима и детаљима истих.

**Модул за Преглед Статистике** даје могућност корисницима да за сваку одиграну утакмицу прегледају генералну статистику, као и индивидуалну статистику сваког играча који је учествовао на мечу. Администратор система може да управља статистиком.

**Модул за Праћење Новости** олакшава корисницима да остану у току са свим дешавањима у вези са клубом и његовим играчима. У оквиру овог модула је корисницима омогућен приступ артиклима о историји клуба, као и артиклима који садрже битне информације. Администратор система има могућност управљања свим врстама артикала на веб сајту.

## Преглед софтверског решења

Серверска апликација је развијена у програмском језику C# [1]. Централни ентитети овог софтверског система су Меч (енгл. *Match*), Играч (енгл. *Player*) и Статистика (енгл. *Statistics*). Клијентска апликација је развијена користећи Ангулар радни оквир (енгл. *Angular framework*) и пружа корисницима интуитиван кориснички интерфејс који се пролагођава ширини екрана и омогућава фановима једноставно праћење свих новости и резултата клуба.

## Јединственост софтверског решења

Интеграцијом четири претходно наведених кључних модула створено је уникатно софтверско решење које садржи све круцијалне функционалности модерног веб сајта фудбалског клуба. Упркос томе што има много сличности са другим веб сајтовима тог типа, овај софтверски систем се издваја по јединственом дизајну и приступу решењу проблема. Повећавањем обима функционалности и интеграцијом са постојећим агрегаторима статистика овај рад би направио значајан корак ка унапређењу корисничког искуства и генералног квалитета веб сајта.

## Организација рада

Преостали део овог дипломског рада организован је према следећем обрасцу: у другом поглављу су анализирана решења са сличним функционалностима која су инспирисала овај рад. Треће поглавље се фокусира на технологије које су коришћене за развој софтверског система, док се у четвртом поглављу излаже комплетна спецификација решења. Пето поглавље се односи на конкретну имплементацију, док се шесто поглавље бави демонстрацијом решења. На самом крају, седмо поглавље закључује овај рад и нуди смернице за будућа истраживања и даљи развој.

# ПРЕГЛЕД СЛИЧНИХ СИСТЕМА

Приликом развоја новог софтверског система битно је истражити тржиште и анализирати постојеће системе и апликације које се баве сличним циљевима и функционалностима као развијан систем. Ова анализа олакшава детаљну спецификацију функционалности које ће софтверско решење имплементирати и даје увид у то како да његове функционалности буду јединствене и покрију неке случајеве који до сада нису решени. Ова секција има за циљ да пружи детаљнији преглед постојећих решења и да их упореди са системом описаном у овом раду. У наставку ће бити описан пар одабраних веб сајтова фудбалских клубова и анализираће се добре и лоше стране оба решења.

## Arsenal.com

Веб сајт енглеског фудбалског клуба Арсенал [2] представља пример једног од бољих решења за сајт фудбалског клуба. Због својих добрих карактеристика коришћен је делимично као узор за развитак веб сајта описаног у овом дипломском раду.

Веб сајт Арсенала садржи све кључне компоненте за функционисање једног веб сајта фудбалског клуба, као што су преглед новости, играча, утакмица и сл. Поред најосновнијих података веб сајт садржи и исцрпну статистику за сваки одиграни меч и за сваког играча који је наступио. Начин визуелизације ових података је изузетно напредан и омогућава кориснику интуитиван преглед истих. Распоред садржаја на целом веб сајту је врло добро организован, што омогућава корисницима да лако навигују кроз веб сајт и брзо пронађу жељене информације.

Осим доступности информација, омогућена је и интеракција са фановима у виду гласања за играча и гол месеца, у виду квизова о историји клуба и сл. Ови типови интеракције повећавају ангажман са веб сајтом и самим тим расте жеља за поновним посетама.

Ако би се морала издвојити нека мана овог веб сајта, то би онда била навигациона трака и количина опције у истој. Преласком курсора преко дугмића на навигационој траци отвара нове подменије који умеју да заузму до пола странице.

## Partizan.rs

Веб сајта српског фудбалског клуба Партизан [3] је у односу на веб сајт Арсенала за нијансу оскуднији у погледу доступних информација. Кључне компоненте попут резултата, предстојећих утакмица и новости су доступни и лако их је пронаћи. Дизајн и генерални изглед веб сајта је добро одрађен и већину доступних информација није тешко пронаћи. Највећи изузетак је табела са резултатима, која није издвојена у навигационој траци, већ је сакривена на почетној страници иза дугог низа новости и реклама спонзора.

Напоменуто је да веб сајт није толико богат информацијама, као неки други и то се највише огледа у недостатку статистичких података. Детаљнији приказ меча садржи само податке о догађајима на мечу и саставима тимова. Статистику тимова и играча није могуће пронаћи. Осим недостатка статистике играча, о играчима се врло мало може сазнати путем веб сајта. Приказане су само основне информације о играчима које се могу сазнати гледањем утакмице.

Поред ових недостатака, веб сајт функционише релативно добро и омогућава корисницима да се боље упознају са клубом. Фанови могу да прочитају више о историји клуба, омладинским секцијама, стручном штабу и чланству. Веб сајт такође нуди могућност промене језика, што је витално за привлачење иностраних фанова.

# 

# КОРИШЋЕНЕ СОФТВЕРСКЕ ТЕХНОЛОГИЈЕ

У овом поглављу ће бити описане кључне софтверске технологије које су кориштене за имплементацију система. Секција 3.1 ће детаљно описати технологије које су коришћене за имплементацију серверског дела система. Такође, ова секција даје увид у главне коришћене библиотеке. Секција 3.2 ће детаљно описати технологије које су коришћене за имплементацију клијентског дела система. Као и претходна секција, ова секција такође даје увид у главне коришћене библиотеке. Циљ целокупног поглавља је да олакша разумевање основних технологија и њихових библиотека које су употребљене у дипломском раду јер свака од наведених технологија игра кључну улогу у функционисању система.

## Серверске технологије

У наставку ове секције ће бити описане серверске технологије и главни пакети и библиотеке коришћене за развој система а то су: .NET Framework, MongoDb, МоngoDb Driver.

### .NET 6

.NET 6 [4], такође познат као .NET 6.0, представља значајно издање развојне платформе .NET од стране Microsoft-a. Оно означава битну промену у односу на претходне верзије.NET Framework-a i .NET Core, јер обједињује различите варијанте .NET-a у једну платформу која је доступна на различитим платформама, отвореног кода и дизајнирана за различите врсте апликација. .NET 6 нуди побољшане перформансе, проширене могућности и доследан скуп библиотека и алатки на различитим оперативним системима, укључујући Windows, macOS i Linux. Са .NET 6, програмери могу креирати веб апликације, десктоп апликације, микросервисе у облаку и још много тога, све то уз предности модерних и ефикасних карактеристика ове уједињене платформе.

У оквиру дипломског рада .NET 6.0 платформа се користи за развој целокупног серверског система.

### MongoDB

MongoDB [5] је широко распрострањена, NoSQL база података отвореног кода (енгл. *open-source*) која користи документ-оријентисани модел података и складишти информације у BSON формату. MongoDB је постао популаран због своје скалабилности и флексибилности при руковању са великом количином неструктуираних или полуструктуираних података. База података нема шему што омогућава динамичан и агилан приступ развоју структуре података. Великe предности су једноставност употребе, могућност беспрекорне интеграције са многобројним популарним програмским језицима, као и богат језик упита који доприноси високој ефикасности приликом добављања података.

У оквиру овог дипломског рада MongoDB се користи за складиштење свих података у систему.

### MongoDB Driver

MongoDB пружа робустан и ефикасан драјвер (енгл. *driver*) који омогућава глатку интеграцију и интеракцију са MongoDB базама података из C# апликација. Званични MongoDB драјвер [6] је библиотека богата функцијама које омогућавају комуникацију са базама података на природан начин. Са подршком за LINQ упите, програмери могу написати комплексне упите на познатом језику, побољшавајући читљивост и одрживост кода.

У оквиру овог дипломског рада MongoDB Driver је од суштинског значаја за комуникацију са базом података. Користи се за мапирање ентитета система, као што су играчи, утакмице, табеле и други, на одговарајуће документе у бази података. Ово омогућава манипулацију података система путем C# објеката, а MongoDB аутоматски генерише NoSQL упите и управља везама између објеката и колекција у бази података.

## Клијентске технологије

У наставку ове секције ће бити описанеклијентске технологије и главне библиотеке коришћене за развој овог система а то су: Angular Framework, sweetalert2

### Angular Framework

Ангулар [7] је развојни оквир (енгл. *framework*) који се користи за израду клијентских апликација. Овај развојни оквир изграђен је на програмском језику TypeScript [8] и пружа разне могућности за креирање скалабилних веб апликација. Ангулар нуди интегрисане библиотеке које покривају различите аспекте развоја, укључујући рутирање, управљање формама, комуникацију са сервером и многе друге. Такође, Ангулар долази са алатима за развој, изградњу, тестирање и ажурирање кода, што олакшава развојни процес.

У оквиру дипломског рада Ангулар радни оквир је коришћен за креирање целокупне клијентске апликације. Коришћене су компоненте из AngularMaterial библиотеке, као што су mat-button (дугме), mat-dialog (дијалог), mat-table (табела) и друге. У потпуности су искоришћене и ангуларове интегрисане библиотеке које укључују библиотеку за рутирање, библиотеку за комуникацију са сервером као и библиотека за руковање формама. Ангулар ЦЛИ (енгл. *Command Line Interface CLI*) је коришћен за генерисање свих компоненти и сервиса у апликацији, што је олакшало развој и омогућило ефикасно креирање корисничког интерфејса.

### SweetAlert2

SweetAlert2 [9] је популарна JavaScript библиотека која се користи за приказивање интерактивних нотификација које искачу (енгл. *pop-up*) и дијалога у веб апликацијама. Ова библиотека омогућава да се брзо и једноставно прикажу обавештења, упозорења, потврде и друге врсте интерактивних порука корисницима, чиме се побољшава корисничко искуство. Ова библиотека пружа различите опције за прилагођавање изгледа и понашања *pop-up* порука, укључујући могућност додавања дугмади, сличица и анимација.

У оквиру клијентске апликације неке од примена могућности ове библиотеке су: нотификације о успешном ажурирању података, нотификација о успешној пријави на систем, потврде о брисању битних података и друге.

# СПЕЦИФИКАЦИЈА

## Спецификација захтева

У овом поглављу се дефинишу функционални и нефункционални захтеви који су кључни за рад и функционисање система. Функционални захтеви обухватају опис свих функционалности које систем треба да пружи, односно шта систем треба да ради како би задовољио потребе корисника. Нефункционални захтеви дефинишу квалитете система, као што су перформансе, безбедност, употребљивост и друге карактеристике, односно дефинишу како систем треба да ради. Ово поглавље ће описати све аспекте система које је требало имплементирати како би се оствариле функционалности и перформансе система у складу са захтевима корисника без уласка у детаље имплементације.

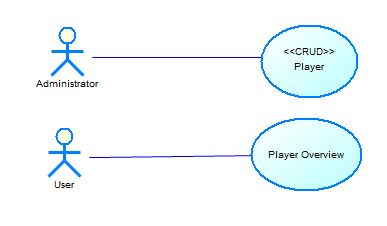
### Функционални захтеви

Функционални захтеви представљају срж система и дефинишу које функционалности систем треба да пружи корисницима. Ови захтеви су произишли из детаљне анализе услуга које пружају софтверска решења која се баве истим доменом.

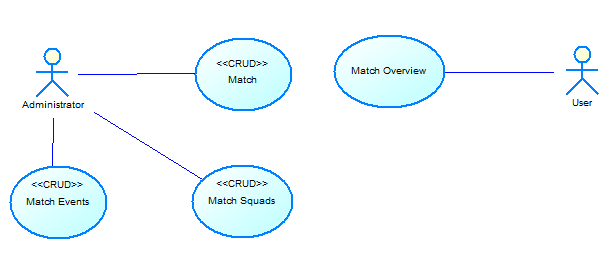
На основу разговора са ментором и детаљне анализе постојећих решења спроведене у претходној секцији дефинисани су главни функционални захтеви. Ови захтеви су представљени као случајеви коришћења [10] (енгл. *Use Cases*) који описују како корисници могу да користе систем како би извршили различите задатке. Сви функционални захтеви су приказани на следећим дијаграмима случајева коришћења и биће детаљно објашњени у наставку.

Случајеви коришћења су подељени на четири групе по једна група за сваки модуо који систем обрађује. Сваки појединачни случај коришћења ће бити детаљно анализиран и обухватати предуслове, основни ток, алтернативне токове, изузетке и последице које оставља у систему. Ова анализа ће бити приказана табеларно.

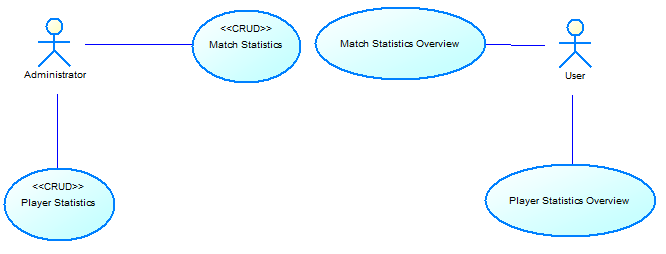
Слика 4.1 илуструје случајеве коришћења модула за преглед профила играча.



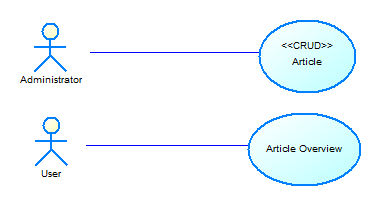
Слика . Случајеви коришћења модула за преглед профила играча

Слика 4.2 илуструје случајеве коришћења модула за преглед мечева. 

Слика 4.2 Случајеви коришћења модула за преглед мечева

Слика 4.3 илуструје случајеве коришћења модула за преглед статистике. 

Слика 4.3 Случајеви коришћења модула за преглед статистике

Слика 4.4 илуструје случајеве коришћења модула за праћење новости.

Слика 4.4 Слувајеви коришћења модула за праћење новости

Случајеви коришћења су подељени на четири групе по једна група за сваки модуо који систем обрађује. Случајеви коришћења ће бити детаљно анализирани и обухватаће предуслове, основни ток, алтернативне токове, изузетке и последице које оставља у систему. Ова анализа ће бити приказана табеларно.

Прва група случајева коришћења која ће бити анализирана је група везана за модуо за преглед профила играча, где су истакнута два случаја коришћења: Додавање, Преглед, Ажурирање и Брисање играча које обавља администратор система, као и приказ играча за обичног корисника.

У табели 4.1 је анализиран случај коришћења „Додавање, Преглед, Ажурирање и Брисање играча“. Подаци које администратор мора да унесе су: име и презиме играча, датум и место рођења, број на дресу, позиција и фотографија. Овај случај коришћења је од изузетне важности, јер омогућава администратору да ажурира сваку промену која се дешава у саставу екипе.

Табела 4.1 Случај коришћења: Додавање, Преглед, Ажурирање и Брисање играча

|  |
| --- |
| **Предуслов:** Корисник мора бити улогован са улогом администратора |
| **Основни ток извршавања:**   1. Администратор бира страницу за додавање/ажурирање играча 2. Администратор попуњава форму одговарајућим подацима 3. Валидност унетих података се утврђује и администратор добија обавештење о успешности ажурирања |
| **Алтернативни ток 3А: Унет је број на дресу који је заузет**   1. Администратор добија обавештење да мора променити унети број на дресу како би додао/ажурирао играча |
| **Последице:** Додати играч је запамћен у систему, односно ажурирани подаци о играчу су запамћени |

Други случај коришћења приказа играча неће бити детаљно анализиран због своје једноставности. Кориснику је омогућен преглед свих играча, подељених на основу позиције, као и индивидуалан приказ одабраног играча.

Друга група случајева коришћења која ће бити анализирана је група везана за модуо за преглед мечева, где су истакнута четири случаја коришћења: Додавање, Преглед, Ажурирање и Брисање мечева, Додавање, Преглед, Ажурирање и Брисање састава екипа за меч, Додавање, Преглед, Ажурирање и Брисање истакнутих догађаја на мечу и приказ мечева.

Први случај коришћења „Додавање, Преглед, Ажурирање и Брисање мечева“ неће бити детаљније анализиран због своје једноставности и сличности са претходно анализираним случајем коришћења „Додавање, Преглед, Ажурирање и Брисање играча“. Треба истаћи да приликом додавања/ажурирања меча не сме доћи до преклапања датума два меча.

У табели 4.2 је анализиран случај коришћења „Додавање, Преглед, Ажурирање и Брисање састава екипа за меч“. Подаци о играчима које администратор мора да унесе су: име и презиме играча и број на дресу. Састави екипа су једни од основних информација у вези са једним фудбалским мечем, стога је могућност ажурирања састава кључна за веб сајт фудбалског клуба.

Табела 4.2 Случај коришћења: Додавање, Преглед, Ажурирање и Брисање састава екипа за меч

|  |
| --- |
| **Предуслов:** Корисник мора бити улогован са улогом администратора |
| **Основни ток извршавања:**   1. Администратор бира меч 2. Администратор бира страницу за ажурирање састава екипа 3. Администратор превлачењем играча мења прву поставу и играче који се налазе на клупи 4. Администратор попуњавањем форме додаје играче противничке екипе у прву поставу или на клупу противника 5. Након уноса података извршене промене се памте |
| **Алтернативни ток 3А: Више играча се помера одједном**   1. Администратор има могућност да селектује више играча и да их истовремено пребаци у прву поставу или на клупу |
| **Алтернативни ток 5А: Погрешан број играча је унет**   1. Администратор добија обавештење да није унет адекватан број играча у првој постави или на клупи једна од екипа |
| **Последице:** Састави екипа за изабрани меч су ажурирани и запамћени у систему |

У табели 4.3 је анализиран случај коришћења „Додавање, Преглед, Ажурирање и Брисање истакнутих догађаја на мечу“. Подаци које администратор мора да унесе за сваки догађај јесу опис догађаја и минут у којем се он одиграо. Опис догађаја треба да почне са кључном речју која описује тип догађаја.

Табела 4.3 Случај коришћења: Додавање, Преглед, Ажурирање и Брисање истакнутих догађаја на мечу

|  |
| --- |
| **Предуслов:** Корисник мора бити улогован са улогом администратора |
| **Основни ток извршавања:**   1. Администратор бира меч 2. Администратор бира страницу за ажурирање истакнутих догађаја 3. Администратор попуњавањем форме додаје догађаје 4. Након уноса података извршене промене се памте |
| **Алтернативни ток 3А: Опис догађаја не садржи кључну реч**   1. Администратор има могућност да обрише погрешно унети догађај којем ће недостајати адекватна сличица |
| **Последице:** Истакнути догађаји за изабрани меч су ажурирани и запамћени у систему |

Последњи случај коришћења везан за модуо за преглед мечева неће бити детањније анализиран збој своје једноставности и интуитивности. Корисник може да прегледа резултате,распоред предстојећих мечева и да филтрира мечеве на основу такмичења. Такође је омогућено и филтрирање резултата по сезонама.

Трећа група случајева коришћења које ће се анализирати је група везана за модуо за преглед статистике, где су истакнута четири случаја коришћења: Додавање, Преглед, Ажурирање и Брисање статистике меча, Додавање, Преглед, Ажурирање и Брисање статистике играча, приказ статистике играча и приказ статистике меча.

Прва два случаја коришћења, „Додавање, Преглед, Ажурирање и Брисање статистике меча и играча“, неће бити детаљније анализирани због своје једноставности. Ови случајеви коришћења се своде на попуњавање форми са адекватним многобројним статистичким подацима као што су: број шутева,број додавања, број корнера, број жутих и црвених картона и сл.

У табели 4.4 је анализиран случај коришћења „Приказ статистике играча“. Овај случај коришћења подразумева преглед статистике изабраног играча за један меч, као и приказ целокупне статистика тог играча. Детаљно ће се анализирати приказ целокупне статистике изабраног играча.

Табела 4.4 Случај коришћења: Приказ статистике играча

|  |
| --- |
| **Предуслов:** Увек се извршава |
| **Основни ток извршавања:**   1. Корисник прелази на страницу за преглед свих играча 2. Корисник бира играча 3. Аутоматски се добавља и приказује целокупна статистика изабраног играча за текућу сезону у свим такмичењима |
| **Алтернативни ток 3А: Филтрирање статистике**   1. Корисник бира сезону и/или такмичење 2. Приказ се ажурира у складу са изабраним филтером |
| **Последице:** Корисник има увид у статистику изабраног играча по сезони и такмичењу |

Последњи случај коришћења везан за модуо за преглед статистика неће бити детаљно анализиран због своје једноставности. Кориснику је омогућен генерални приказ статистике, подељене по категоријама, за изабрани меч.

Четврта и последња група случајева коришћења која ће се анализирати је група везана за модуо за праћење новости, где су истакнута два случаја коришћења: Додавање, Преглед, Ажурирање и Брисање чланака и приказ чланака.

У табели 4.5 је анализиран случај коришћења „Додавање, Преглед, Ажурирање и Брисање чланака“. Подаци које администратор мора да унесе су: тип чланка, наслов, време објављивања чланка, као и информацију о томе да ли је чланак у уској вези са неким мечем и/или играчем. Фреквентно објављивање новости у вези са клубом је круцијално за одржавање везе између клуба и његових фанова, стога је овај случај коришћења од изузетне важности.

Табела 4.5 Случај коришћења: Додавање, Преглед, Ажурирање и Брисање чланака

|  |
| --- |
| **Предуслов:** Корисник мора бити улогован са улогом администратора |
| **Основни ток извршавања:**   1. Администратор прелази на страницу за додавање/ажурирање чланака 2. Администратор попуњава форму основним подацима о чланку 3. Администратор додаје параграфе,слике и видео снимке у чланак 4. Чланак се објављује на веб сајту у време кад је администратор специфицирао |
| **Последице:** Чланак је креиран, односно ажуриран, објављен и меморисан у систему |

Други случај коришћења везан за модуо за праћење новости неће бити детаљно анализиран због своје једноставности. Корисник има увид у све новости сортиране од најновијих ка најстаријим. Корисник такође може на одвојеним страницама да прочита чланке о историји клуба и о важним информацијама у вези са клубом и мечевима.

### Нефункционални захтеви

Нефункционални захтеви представљају оне захтеве који описују које карактеристике и особине система треба да поседује, односно на који начин систем треба да ради. Ови захтеви су произишли из договора са ментором и пажљивом анализом постојећих система који се баве истим или сличним доменом. Фокус нефункционалних захтева је био на корисничком искуству и интуитивном корисничком интерфејсу који омогућава једноставну навигацију кроз систем.

Имајући у виду да веб сајт фудбалског клуба мора да рукује са великом количином података, првенствено у виду статистике, приказ и унос ових података морају бити обезбеђени на практичан и ефикасан начин. Приликом приказа статистике подаци су подељени по одговарајућим категорија са циљем да кориснику омогуће пријатније искуство приликом прегледа података. Кад је у питању унос статистике, администратору су поједностављене и прилагођене форме на такав начин да се минимизује вероватноћа за прављење грешака приликом уноса података. Уколико, ипак, дође до грешке, администратор одмах добија обавештење о грешки и грешка не оставља трајне последице на систем.

Интуитивна навигација кроз систем је критична за веб сајт фудбалског клуба, са тим на уму, кориснички интерфејс је дизајниран на такав начин да корисник може брзо и без напора да пронађе било коју информацију од интереса. Осим тога, кориснички интерфејс се прилагођава величини екрана уређаја што побољшава корисничко искуство.

Последњи кључни нефункционални захтев јесте тај да се осигура безбедност података у систему. Веб сајт фудбалског клуба типично не садржи сензитивне податке који морају остати скривени, али малверзација постојећих података би нарушила кредибилитет веб сајта као поузданог извора информација. Са тим на уму су заштићени, како серверска апликација, тако и клијентска апликација. Свака акција која може да измени податке у систему је омогућена искључиво администратору система.

## Спецификација система

У овом поглављу ће се изложити спецификација система. Спецификација система описује структуру, понашање и архитектуру система, а детаљна анализа је представљена кроз следеће секције:

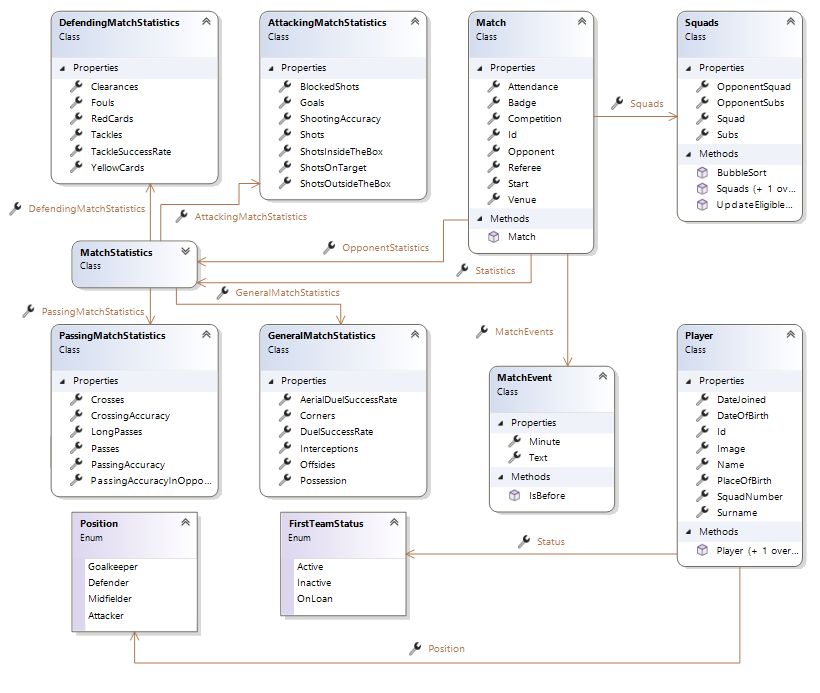
Секција 4.2.1 овог поглавља ће се фокусирати на модел података система који је приказан у виду дијаграма класа. Биће анализирани ентитети, њихови атрибути и међусобни односи ентитета. Овај модел ће омогућити јасно разумевање структуре података које систем користи за своје функционалности.

У секцији 4.2.2 овог поглавља ће бити приказани дијаграми секвенци система. Дијаграм секвенцe је корисни алат за визуелизацију интеракција између различитих компонената система током извршавања одређених сценарија. Кроз ове дијаграме, биће приказани процеси и токови информација унутар система. Ови дијаграми су прављени у алату PowerDesigner [11].

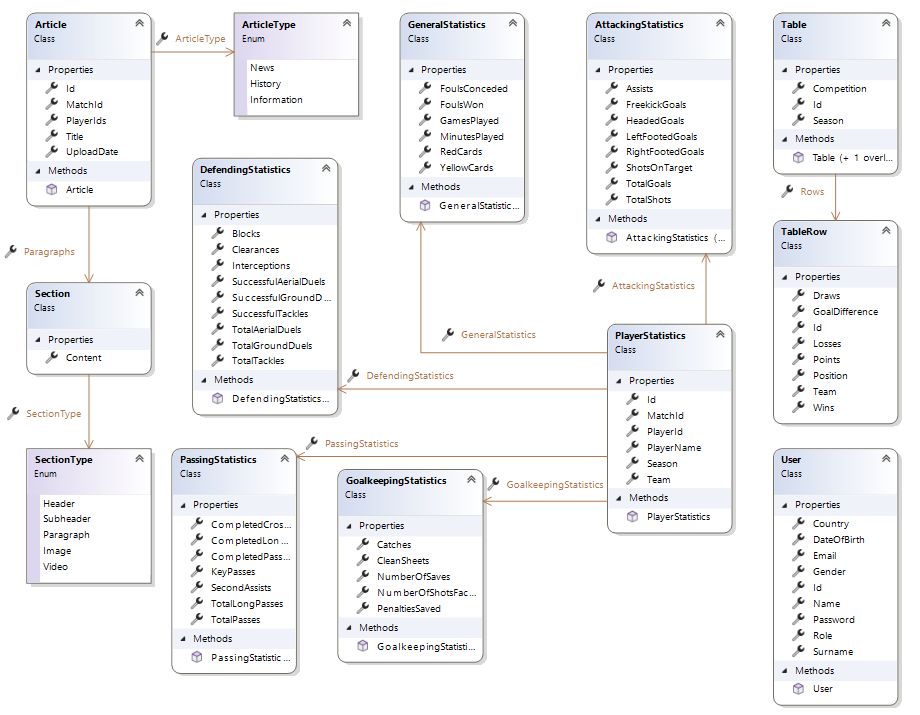
Секција 4.2.3 овог поглавља се бави архитектуром система. Архитектура система дефинише основне компоненте система, њихове функционалности, као и међусобну интеракцију и комуникацију компоненти. Архитектура система пружа апстрактни приказ система, што омогућава разумевање његових главних карактеристика без уласка у имплементационе детаље.

### Модел података

У оквиру овог дипломског рада, посебан нагласак ставља се на креирање и разумевање модела података. Модел података представља апстрактан приказ података и њихових међусобних односа који се користе у информационом систему. Модел података је од суштинског значаја током анализе, дизајна и имплементације базе података, као и самих функционалности софтвера. Пружа дубље разумевање структуре и сврхе података и омогућава проверу њихове конзистентности и интегритета. Такође, има улогу у документовању и одржавању система. На слици 4.5 је приказана лева страна дијаграма класа, а на слици 4.6 је приказана десна страна. У наставку секције ће се анализирати сви ентитети модела.



Слика 4.5 Лева страна дијаграма класа



Слика 4.6 Десна страна дијаграма класа

Класа Player је централни ентитет модула за преглед профила играча и поред основних биографских података садржи следећа поља: DateJoined (датум доласка у клуб), Image (слика), SquadNumber (број на дресу), Status (статус) и Position (позиција на терену). Статус играча може бити Active (активан), Inactive (неактиван) или OnLoan (на позајмици), док позиција на терену може бити Goalkeeper (голман), Defender (дефанзивац), Midfielder (везиста) или Attacker (нападач). Раније је истакнуто да број на дресу мора бити уникатан међу активним играчима.

Класа Match је централни ентитет модула за преглед мечева и могла би се посматрати и као централни ентитет читавог система. Ова класа садржи поља која описују основне информације о мечу као што су: Opponent (противник), Badge (грб противника), Competition (такмичење), Start (време почетка меча), Venue (стадион), Referee (судија) и Attendance (број гледалаца). Осим ових поља, ова класа садржи и поља Statistics (статистика) и OpponentStatistics (статистика противника), као и поља MatchEvents (догађаји на мечу) и Squads (састави екипа). Поље MatchEvents садржи низ објеката класе MatchEvent која има поља Minute (минут) и Text(текст), као и методу IsBefore() која помаже при хронолошком сортирању догађаја. Класа Squads садржи поља која садрже информацију о стартној постави и клупи оба тима. У оквир овог модула улази и класа Table која садржи поља Competition (такмичење) и Season (сезона), као и поље Rows (редови). Rows поље садржи низ објеката класе TableRow (ред у табели) која садржи поља у којима се бележи име тима, број бодова, број победа и сл.

Централни ентитети модула за преглед статистике су MatchStatistics (статистика меча) и PlayerStatistics (статистика играча). Ове две класе садрже нека врло слична поља која представљају различите врсте статистика. MatchStatistics класа садржи поља: GeneralMatchStatistics (генерална статистика меча), PassingMatchStatistics (статистика додавања), AttackingMatchStatistics (нападачка статистика) и DefendingMatchStatistics (дефанзивна статистика). PlayerStatistics класа садржи поља истих назива без кључне речи Match уз додатак поља GoalkeepingStatistics (голманска статистика). Класа PlayerStatistics још садржи и поља која описују за ког играча, меч и сезону је статистика везана. Многобројне статистичке класе неће бити темељно анализиране због своје једноставност и великог број тривијалних поља.

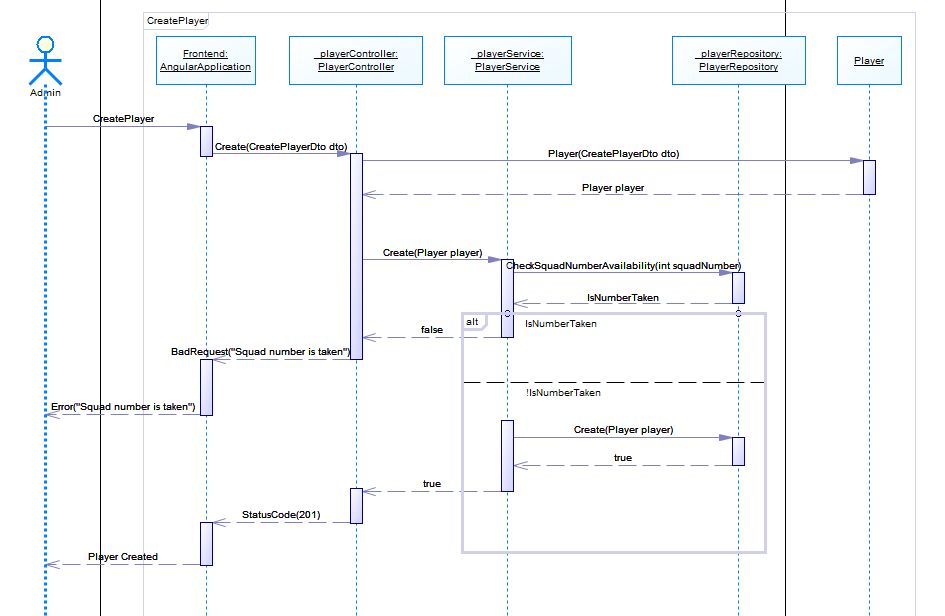
Класа Article је централни ентитет модула за праћење новости и садржи поља: Title (наслов), UploadDate (датум објављивања), ArticleType (тип чланка), Paragraphs (параграфи), MatchId (идентификатор меча) и PlayerIds (идентификатори играча). Поље Paragraphs садржи низ објеката класе Section (одељак) која садржи поља Content (садржај) и SectionType (тип одељка). Поља MatchId и PlayerIds служе да би се успоставила веза између чланка и меча, односно играча, на које се чланак односи.

Класа User није у уској вези ни са једним од четири модула и она садржи поља са основним подацима о кориснику, укључујући податке о његовом налогу, као што су имејл-адреса, лозинка и улога у систему.

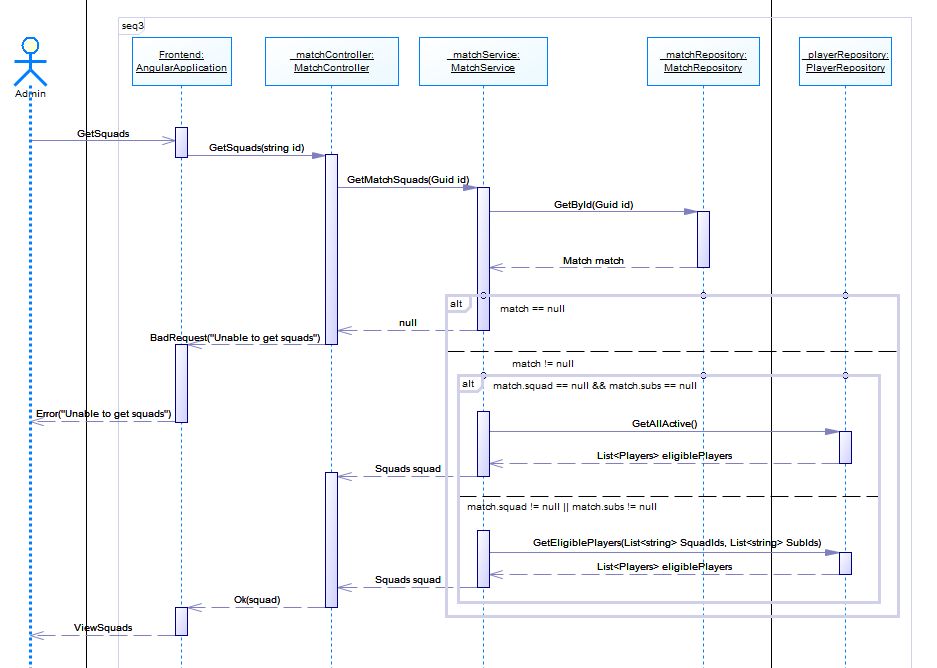
### Дијаграми секвенци

Дијаграми секвенци су графички прикази који омогућавају разумевање редоследа догађаја и комуникацију између различитих компонената система током извршавања одређених сценарија. Ови дијаграми представљају алат за идентификацију и анализу тока података унутар функционалности система.

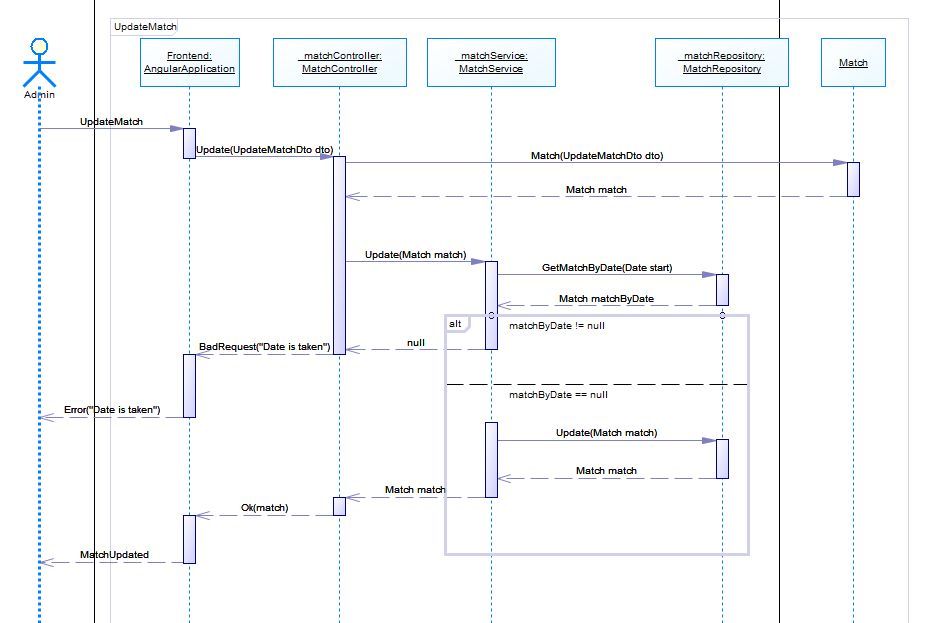
У наставку су представљена три дијаграма секвенци који илуструју интеракције у систему. На слици 4.7 ће бити илустрован дијаграм за креирање новог играча. На слици 4.8 ће бити илустрован дијаграм за добављање састава тимова за неки меч. На слици 4.9 ће бити илустрован дијаграм за ажурирање основних информација о мечу.



Слика 4.7 Дијаграм секвенце: Креирање Player класе



Слика 4.8 Дијаграм секвенце: Добављање састава тимова



Слика 4.9 Дијаграм секвенце: Ажурирање Match класе

### Архитектура система

Архитектура система представља кључни стуб сваког софтверског решења, јер она дефинише основне компоненте система, њихове улоге и карактеристике, као и начин њихове међусобне комуникације. У наставку ове секције биће описан апстрактан приказ система.

Архитектура система је базирана на Onion [13] архитектури, чији је главни принцип разлагање система у слојеве, тако да сваки слој има одређену улогу и може да комуницира само са слојевима са којима је у директном контакту. Централни слој се назива доменски (енгл. *Domain*) слој и у њему се налазе ентитети, репозиторијуми и сервиси система. Овај слој представља срж апликације и он садржи пословну логику и правила система. Такође, овај слој би требало да буде независан од других слојева и технологија у којим је развијен. Следећи слој је апликациони (енгл. *Application*) слој који служи за координацију доменског слоја са спољашњим светом. У овом слоју се налазе сервиси система који се мапирају на функционалне захтеве система, једноставни су и углавном представљају CRUD [12] операције. Следећи слој се састоји из два дела: инфраструктурног (енгл. *Infrastrucure*) слоја и презентационог (енгл. *Presentation*) слоја. Инфраструктурни слој служи за обезбеђивање комуникације са екстерним системима, док презентациони слој представља улазну тачку у систем уз помоћу које корисници интерагују са системом. У овом систему презентациони слој је у облику Web API-ја и чине га различити контролери помоћу којих је корисницима омогућен приступ функционалностима система.

# ИМПЛЕМЕНТАЦИЈА

У овом поглављу ће бити представљена конкретна имплементација софтверског решења веб сајта фудбалског клуба. У првој секцији овог поглавља биће описана генерална конфигурација серверског и клијентског система, док ће у другој секцији фокус бити на конкретним функционалним захтевима и на њиховој имплементацији. Функционалности које ће бити описане су оне чије су дијаграми секвенци приказани у претходном поглављу.

## Конфигурација система

Ова секција пружа детаљну документацију Program.cs и Helper.cs фајлова који представљају главне конфигурационе класе у оквиру ASP.NET Web API апликације “FootballClubBackend”. Ови фајлови садрже логику која је кључна за подизање веб абликације, конфигурацију различитих сервиса, аутентификацију, ауторизацију и друге компоненте компоненте које су неопходне за функционалност целикупног система.

У листингу 5.1 приказано је како се гради и покреће веб апликација у Program.cs фајлу.

|  |
| --- |
| var builder = WebApplication.CreateBuilder(args);  //...  var app = builder.Build();  //...  app.Run(); |

Листинг 5.1 Грађење и покретање веб апликације

У Program.cs фајлу је извршена конфигурација HTTP захтева и одговора, додавање средњег слоја за аутентификацију и ауторизацију, и мапирање контролера. У листингу 5.2 је приказана импементација претходно наведених ставки.

|  |
| --- |
| builder.Services.**AddSwaggerGen**();  builder.Services.**AddAuthentication**(JwtBearerDefaults.AuthenticationScheme)  .AddJwtBearer(options =>  {  //…  });  app.**UseAuthorization**();  app.**UseCors**(x => x  .AllowAnyOrigin()  .AllowAnyMethod()  .AllowAnyHeader()  );  app.**MapControllers**(); |

Листинг 5.2 Конфигурације у Program.cs фајлу

AddSwaggerGen метода се користи за конфигурацију Swagger [14] документације, што омогућава генерисање и преглед API спецификације. AddAuthentication и UseAuthorization методе се користе за конфигурацију JWT аутентификације и ауторизације. Ту су дефинисани параметри за валидацију JWT токена, као што су кључеви и издавач токена. AddCors метода се користи како би се обезбедио *Cross-Origin Resource Sharing* (CORS), што омогућава захтевима различитих извора, метода и заглавља приступ ресурсима које нуди API. Метода MapControllers служи за мапирање улазних тачака контролера.

У Helper.cs фајлу је дефинисана метода која служи за регистрацију сервиса и репозиторијума који су неопходни за функционисање система. Осим тога, наведене су и метода за генерисање JWT токена и метода за хеширање (енгл. *Hashing*) и верификацију лозинки. У листингу 5.3 су приказане ове методе.

|  |
| --- |
| public static void AddScoped(WebApplicationBuilder builder)  {  builder.Services.AddScoped<PlayerService>();  builder.Services.AddScoped<PlayerRepository>(); builder.Services.AddScoped<PlayerStatisticService>(); builder.Services.AddScoped<PlayerStatisticRepository>();  builder.Services.AddScoped<MatchService>();  builder.Services.AddScoped<MatchRepository>();  builder.Services.AddScoped<UserService>();  builder.Services.AddScoped<UserRepository>();  builder.Services.AddScoped<ArticleService>();  builder.Services.AddScoped<ArticleRepository>();  builder.Services.AddScoped<TableService>();  builder.Services.AddScoped<TableRepository>();  }  public static string GenerateToken(string username, string role)  {  var tokenHandler = new JwtSecurityTokenHandler();  // ...  var descriptor = new SecurityTokenDescriptor  {  // ...  };  var token = tokenHandler.CreateToken(descriptor);  return tokenHandler.WriteToken(token);  }  public static string HashPassword(string plainPassword)  {  return BCrypt.Net.BCrypt.HashPassword(plainPassword, BCrypt.Net.BCrypt.GenerateSalt());  }  public static bool VerifyPassword(string plainPassword, string hashedPassword)  {  return BCrypt.Net.BCrypt.Verify(plainPassword, hashedPassword);  } |

Листинг 5.3 Методе у Helper.cs фајлу

## Имплементација функционалних захтева

У оквиру ове секције је приказана имплементација функционалних захтева система. Захтеви који су били приказани у претходном поглављу кроз дијаграме секвенци биће детаљно анализирани кроз кôд у наставку. Ова секција је подељена на 3 дела, сваки за по један функционални захтев. Подсекција 5.2.1 се односи на креирање Player класе. Подсекција 5.2.2 анализира добављање састава тимова. Подсекција 5.2.3 анализира имплементацију ажурирања Match класе.

### Креирање Player класе

Почетна тачка овог функционалног захтева јесте креирање и слање HTTP Post захтева са клијентске на серверску апликацију. Тело захтева представља JSON објекат и пример захтева је приказан у листингу 5.4.

|  |
| --- |
| {  "name": "string",  "surname": "string",  "dateOfBirth": "2024-04-15T12:03:27.877Z",  "placeOfBirth": "string",  "position": 0,  "squadNumber": 99,  "image": "string"  } |

Листинг 5.4 Тело CreatePlayer захтева

Овај захтев прихвата PlayerController класа, која наслеђује ControllerBase класу из Microsoft.AspNetCore.Mvc пакета. Приступ улазној тачки за креирање играча имају само захтеви које је послао корисник са улогом администратора. Позива се метода Create у PlayerService класи и на основу резултата тог позива се формира HTTP одговор. Претходно описана метода приказана је у листингу 5.5.

|  |
| --- |
| [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Roles = "Admin")]  [HttpPost]  public ActionResult Create(CreatePlayer dto)  {  return \_playerService.Create(new Player(dto)) ? Ok(new { message = "Player created successfully" }) : BadRequest(new { message = "Squad number is taken" });  } |

Листинг 5.5 Улазна тачка креирања Player класе

У листингу 5.6 приказана је Create метода унутар PlayerService класе. Ова метода прво проверава, помоћу CheckSquadNumberAvailability методе из PlayerRepository класе, да ли је број на дресу новог играча већ заузет. Уколико није, позива се метода Create из PlayerRepository класе.

|  |
| --- |
| public bool Create(Player player)  {  if (\_playerRepository.CheckSquadNumberAvailability(player.SquadNumber))  {  return false;  }  \_playerRepository.Create(player);  return true;  } |

Листинг 5.6 Метода Create у PlayerService класи

Метода CheckSquadNumberAvailability је приказана у листингу 5.7 , док је у листингу 5.8 приказана Create метода за креирање новог играча. Ове методе се налазе у PlayerRepository класи.

|  |
| --- |
| public bool CheckSquadNumberAvailability(int squadNumber)  {  var filter = Builders<Player>.Filter.And(  Builders<Player>.Filter.Eq(p => p.SquadNumber, squadNumber),  Builders<Player>.Filter.Ne(p => (int)p.Status, 1)  );  return collection.Find(filter).Any();  } |

Листинг 5.7 Метода за проверу доступности броја на дресу

|  |
| --- |
| public void Create(Player player)  {  player.Id = Guid.NewGuid();  collection.InsertOne(player);  return;  } |

Листинг 5.8 Креирање новог играча

## Добављање састава тимова

Почетна тачка овог функционалног захтева јесте креирање и слање HTTP Get захтева са клијентске на серверску апликацију. Овај захтев прихвата MatchController класа, која такође наслеђује ControllerBase класу. Приступ овој улазној тачки имају сви корисници, укључујући неулоговане кориснике. Најпре се извршава конверзија идентификатора меча из string облика Guid објекат. Након тога се позива метода GetMatchSquads из MatchService класе, на основу чијег резултата се формира одговарајући HTTP одговор. Претходно описана метода приказана је у листингу 5.9.

|  |
| --- |
| [HttpGet("squads/{id}")]  public ActionResult GetSquads(string id)  {  var guid = Guid.Parse(id);  Squads squads = \_matchService.GetMatchSquads(guid);  if (squads != null)  return Ok(\_matchService.GetMatchSquads(guid));  return BadRequest(new { message = "Unable to get squads" });  } |

Листинг 5.9 Улазна тачка добављања састава тимова

У листингу 5.10 приказана је GetMatchSquads метода унутар MatchService класе. Најпре се позива GetById метода из MatchRepository класе како би се добио приступ Match објекту. Будући да Match класа садржи само идентификаторе играча, потребно је позвати и методу GetInMatchSquad из PlayerRepository класе, како би се добавили подаци о играчима. После тога се добављају сви преостали играчи у клубу позивом GetEligibleForMatch или GetAllActive методе из PlayerRepository класе. На крају се врши конверзија у одговарајући облик да би се послала назад клијентској апликацији.

|  |
| --- |
| public Squads? GetMatchSquads(Guid id)  {  Match match = \_matchRepository.GetById(id);  if (match == null)  return null;  var squadIds = match.SquadIds;  var subsIds = match.SubsIds;  var opponentSquad = match.OpponentSquad;  var opponentSubs = match.OpponentSubs;  var squad = \_playerRepository.  GetInMatchSquad(squadIds);  var subs = \_playerRepository.  GetInMatchSquad(subsIds);  ICollection<Player> eligiblePlayers;  if (squadIds != null || subsIds != null)  {  eligiblePlayers = \_playerRepository.  GetEligibleForMatch(squadIds,subsIds);  }  else  {  eligiblePlayers = \_playerRepository.  GetAllActive();  }  return new Squads(squad,eligiblePlayers,  opponentSquad,subs,opponentSubs);  } |

Листинг 5.10 Метода GetMatchSquads у MatchService класи

У листингу 5.11 је приказана метода GetInMatchSquad, док је у листингу 5.12 приказана метода GetEligibleForMatch. Ове методе се налазе у PlayerRepository класи.

|  |
| --- |
| public ICollection<Player> GetInMatchSquad(ICollection<Guid>? ids)  {  if (ids == null)  {  return null;  }  var filter =Builders<Player>.Filter.In(p => p.Id, ids);  return collection.Find(filter).ToList();  } |

Листинг 5.11 Метода GetInMatchSquad у PlayerRepository класи

|  |
| --- |
| public ICollection<Player> GetEligibleForMatch(ICollection<Guid>? squadIds, ICollection<Guid>? subsIds)  {  var filter = Builders<Player>.Filter.And(  Builders<Player>.Filter.Eq(p => p.Status,FirstTeamStatus.Active),  Builders<Player>.Filter.Nin(p => p.Id, squadIds),  Builders<Player>.Filter.Nin(p => p.Id, subsIds));  return collection.Find(filter).ToList();  } |

Листинг 5.12 Метода GetEligibleForMatch у PlayerRepository класи

### Ажурирање Match класе

Почетна тачка овог функционалног захтева јесте креирање и слање HTTP Put захтева са клијентске на серверску апликацију. Тело захтева представља JSON објекат и пример захтева је приказан у листингу 5.13.

|  |
| --- |
| {  "id": "string",  "opponent": "string",  "venue": "string",  "referee": "string",  "competition": "string",  "start": "2024-04-15T17:21:51.234Z",  "badge": "string",  "attendance": 0  } |

Листинг 5.13 Тело UpdateMatch захтева

Овај захтев прихвата PlayerController класа и приступ овој улазној тачки само захтеви које је послао корисник са улогом администратора. Позива се метода Update у MatchService класи и на основу резултата тог позива се формира HTTP одговор. Претходно описана метода приказана је у листингу 5.14.

|  |
| --- |
| [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Roles = "Admin")]  [HttpPut("update/match")]  public ActionResult Update(UpdateMatchDto dto)  {  Match? match = \_matchService.  Update(new Match(dto));  if (match != null)  return Ok(match);  return BadRequest(new { message = "Date is taken" });  } |

Листинг 5.14 Улазна тачка за ажурирање Match класе

У листингу 5.15 приказана је Update метода у MatchService класи. Најпре се позивом GetByDate методе из MatchRepository класе проверава да ли је заузет новоизабрани датум. Уколико је датум слободан, позива се метода Update из MatchRepository класе.

|  |
| --- |
| public Match? Update(Match match)  {  if (\_matchRepository.GetByDate(match.Start) == null)  {  return \_matchRepository.Update(match);  }  return null;  } |

Листинг 5.15. Метода Update у MatchService класи

У листингу 5.16 је приказана метода GetByDate, док је у листингу 5.17 приказана метода Update. Ове методе се налазе у MatchRepository класи.

|  |
| --- |
| public Match? GetByDate(DateTime start)  {  var helperDateFrom = start.Date;  var helperDateTo = helperDateFrom.AddDays(1);  var matchFilter = Builders<Match>.Filter.And(  Builders<Match>.Filter.Gte  (m => m.Start, helperDateFrom),  Builders<Match>.Filter.Lte  (m => m.Start, helperDateTo)  );  return collection.Find(matchFilter).  FirstOrDefault();  } |

Листинг 5.16 Метода GetByDate у MatchRepository класи

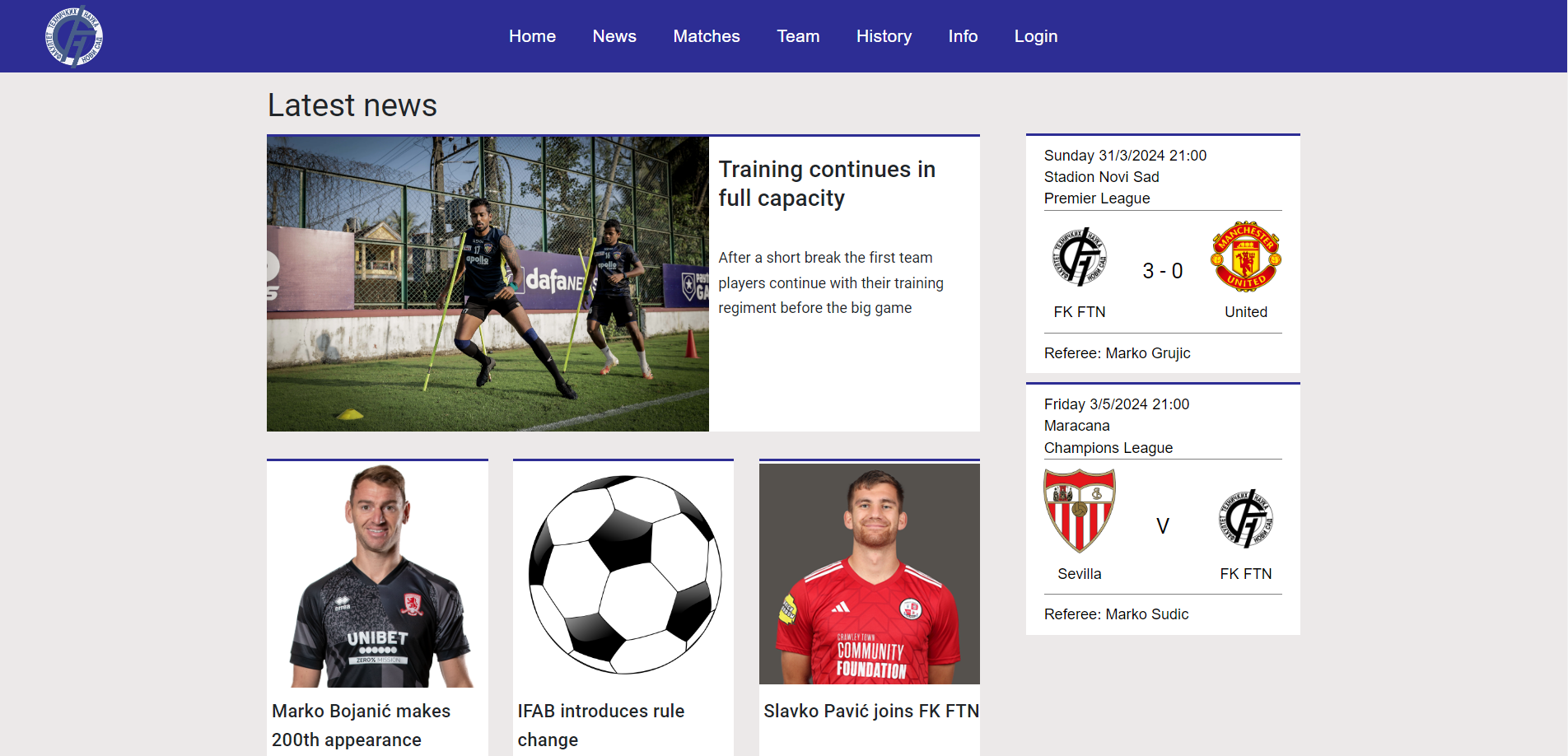
|  |
| --- |
| public Match Update(Match match)  {  var filter = Builders<Match>.Filter.Eq(m => m.Id, match.Id);  var update = Builders<Match>.Update.  Set(m => m.Attendance, match.Attendance).  Set(m => m.Venue, match.Venue).  Set(m => m.Opponent, match.Opponent).  Set(m => m.Badge, match.Badge).  Set(m => m.Referee, match.Referee).  Set(m => m.Start, match.Start).  Set(m => m.Competition, match.Competition);  var options = new FindOneAndUpdateOptions<Match>  { ReturnDocument = ReturnDocument.After };  return collection.FindOneAndUpdate(filter, update, options);  } |

Листинг 5.17 Метода Update у MatchRepository класи

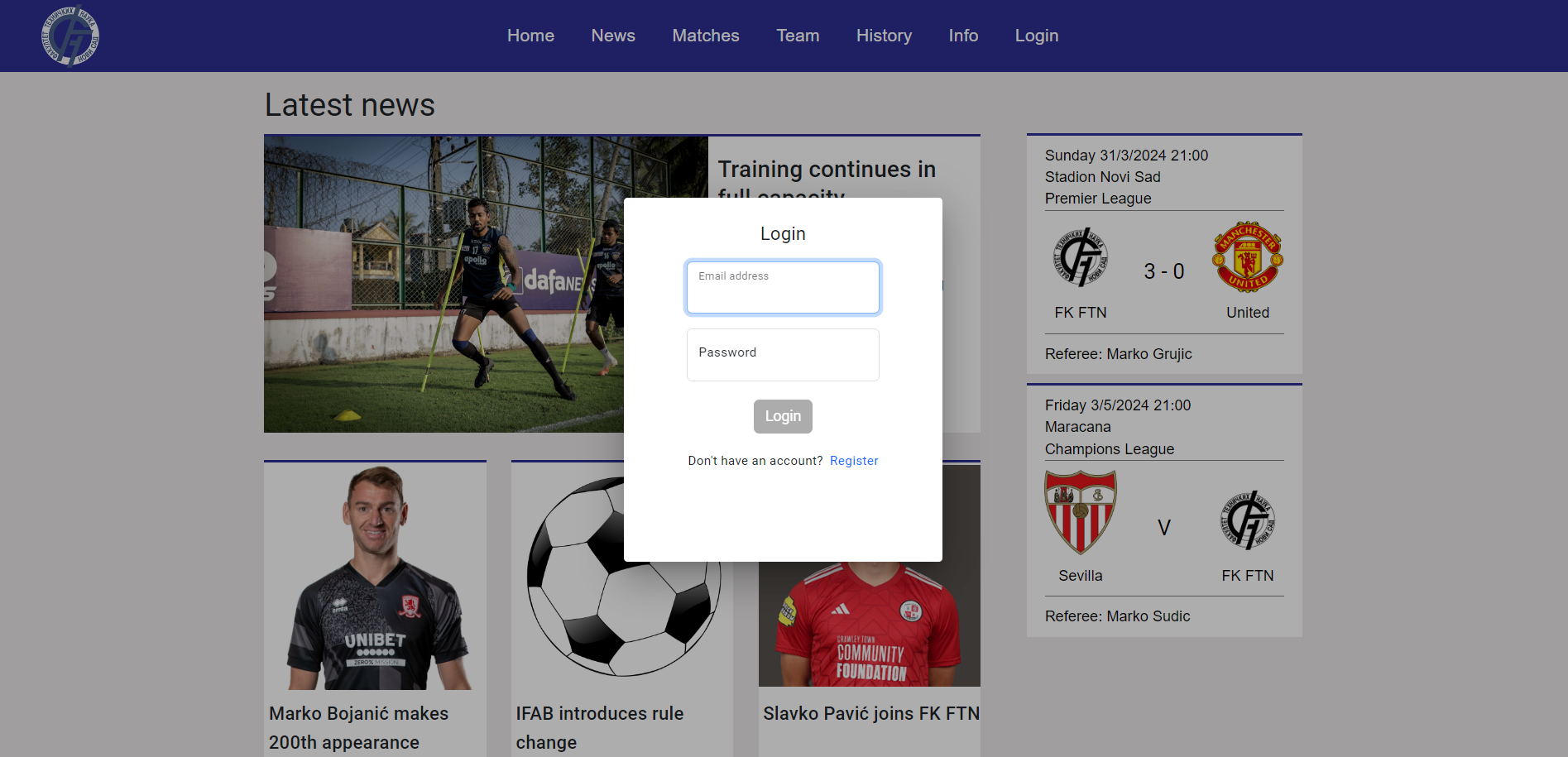
# ДЕМОНСТРАЦИЈА

У оквиру овог поглавља је приказан кориснички интерфејс (енгл. *Interface*) за веб сајт фудбалског клуба који је развијен за подршку функционалности дипломског рада. Сврха овог поглавља јесте да се истакну најбитније функционалности система и може послужити као упутство за коришћење клијентске апликације.

Приликом уласка на веб сајт корисник долази на почетну страницу на којој су истакнуте најновије вести, као и информације о актуелним мечевима. Са ове странице корисник може помоћу навигационе траке да се пребаци на страницу о мечевима, играчима, информацијама или историји клуба. Слика 6.1 илуструје почетну страницу.



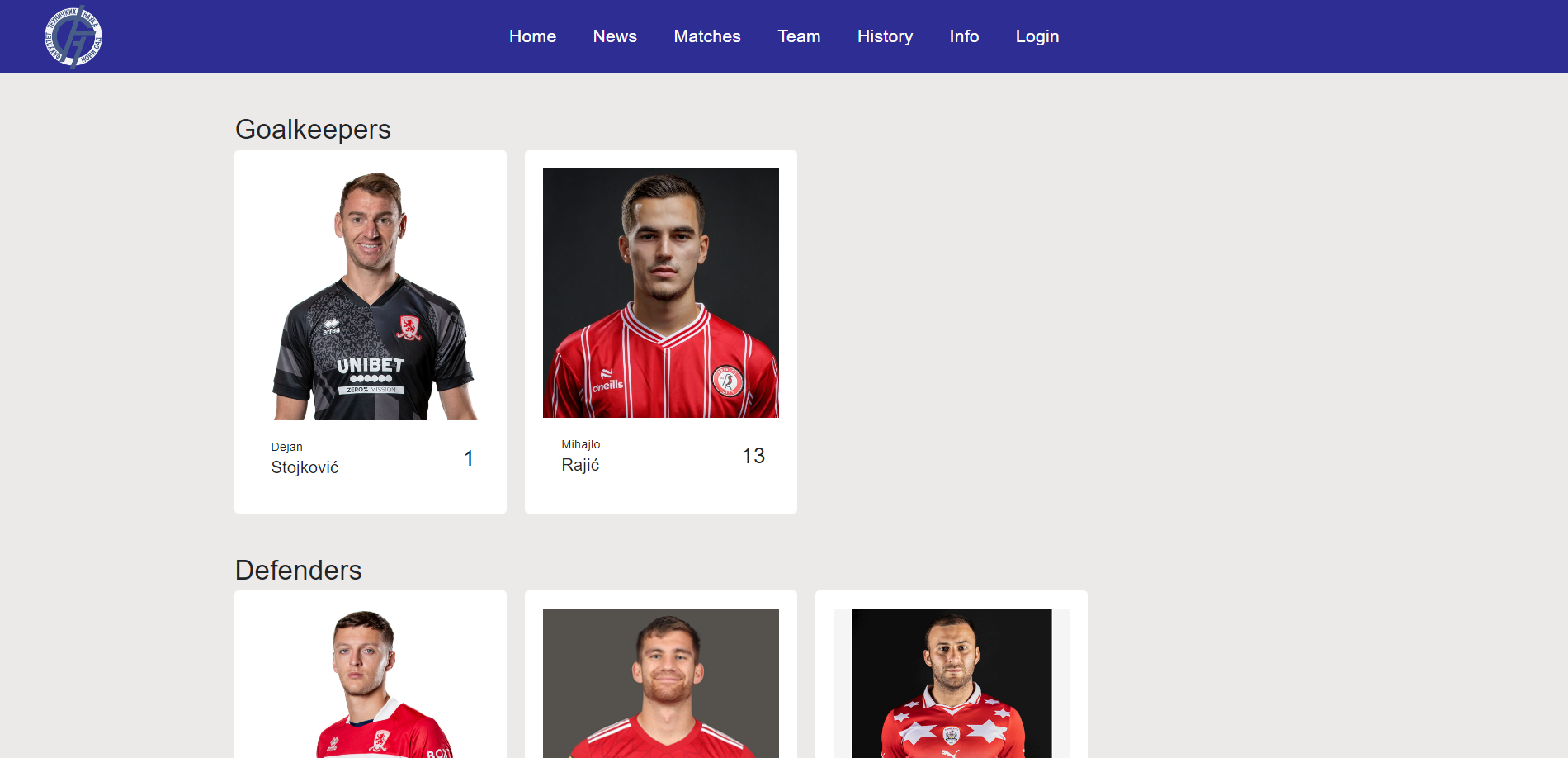
Слика 6.1 Приказ почетне странице

У навигационој траци се такође налази дугме за пријаву на систем. Кликом на ово дугме отвара се дијалог за пријаву са којег се може прећи и на страницу за регистрацију у случају да корисник још увек није креирао налог. На слици 6.2 приказан је дијалог за пријаву. 

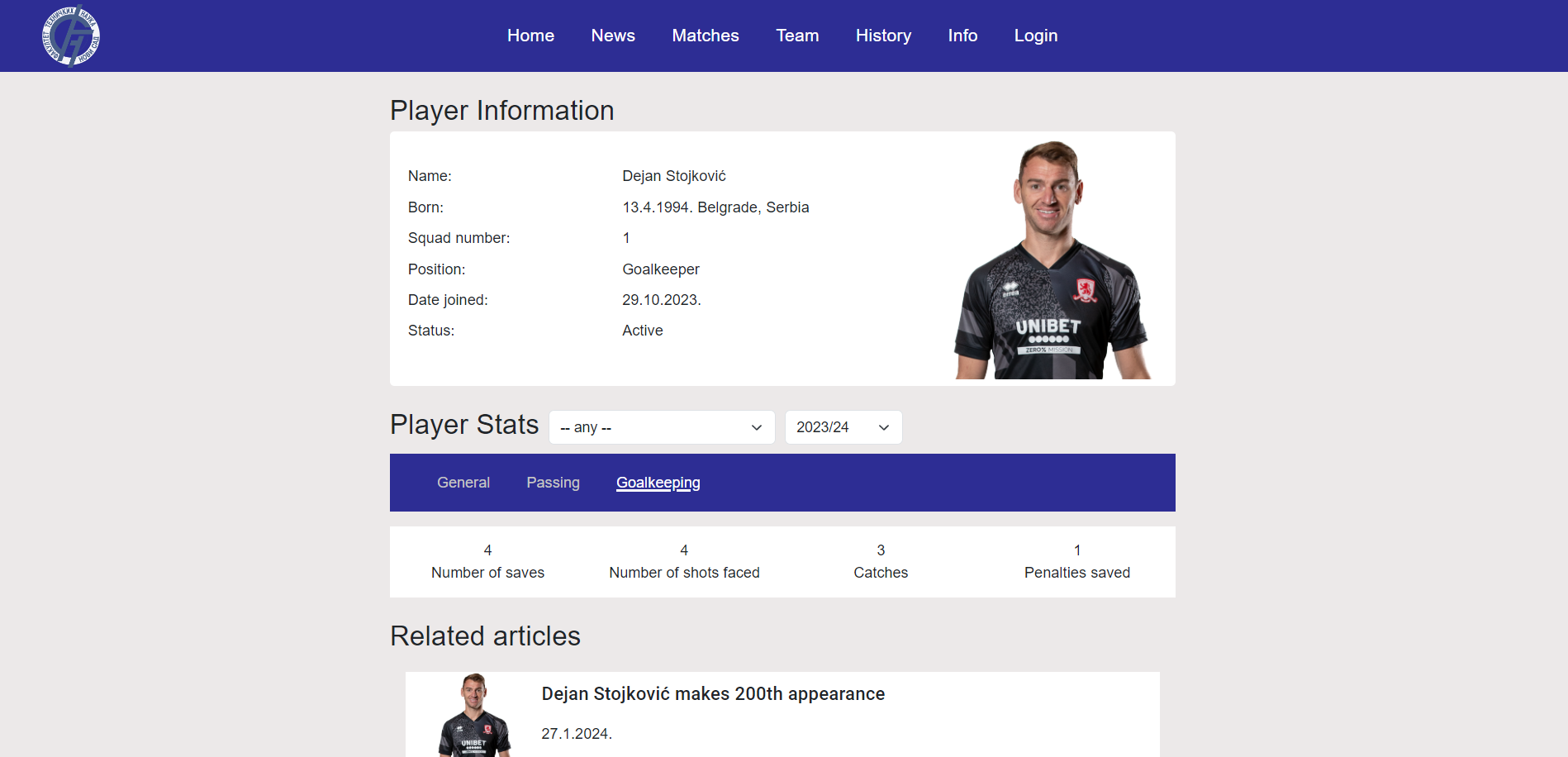
Слика 6.2 Приказ дијалога за пријаву на систем

## Модул за преглед профила играча

Кликом на Team дугме у оквиру навигационе траке, корисник прелази на страницу за преглед свих играча, сортираних на основу позиције на терену. Кликом на једног од играча прелази се на страницу на којој су приказани биографски подаци, статистика и вести у вези са изабраним играчем. На слици 6.3 илустрован је приказ свих играча, док је на слици 6.4 илустрован индивидуални преглед профила играча.



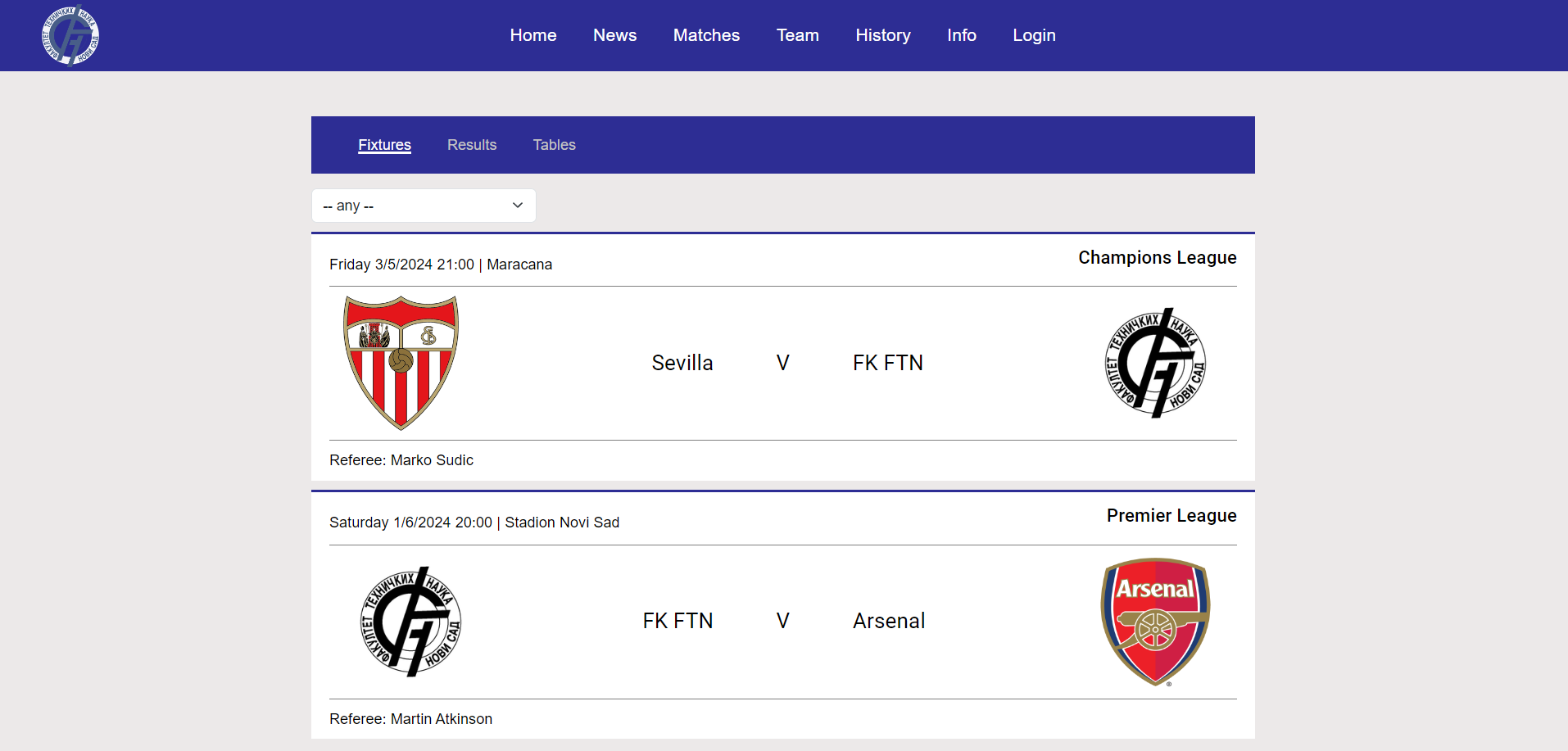
Слика 6.3 Приказ свих играча



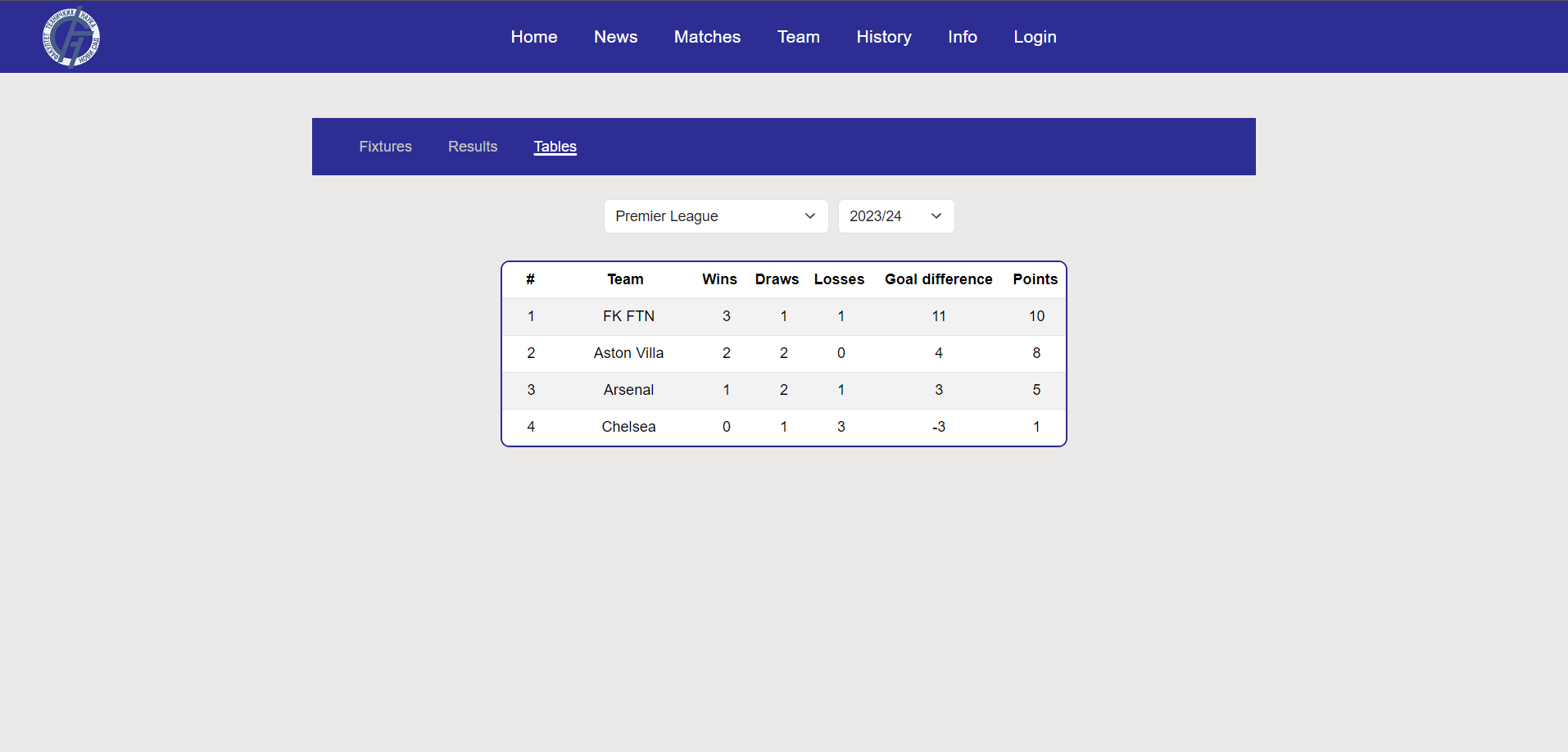
Слика 6.4 Приказ индивидуалног профила играча

## Модул за преглед мечева

Кликом на Matches дугме у оквиру навигационе траке, корисник прелази на страницу за преглед предстојећих мечева. Корисник може кликом на дугме Results да види резултате утакмица сортиране од најскоријих ка најстаријим, а такође може да види резултате претходних сезона. Осим тога, кликом на Tables дугме корисник може да прегледа стање на табели за сва такмичења. На сликама 6.5 и 6.6 приказани су изгледи странице за преглед предстојећих мечева и странице за преглед табела, редом.

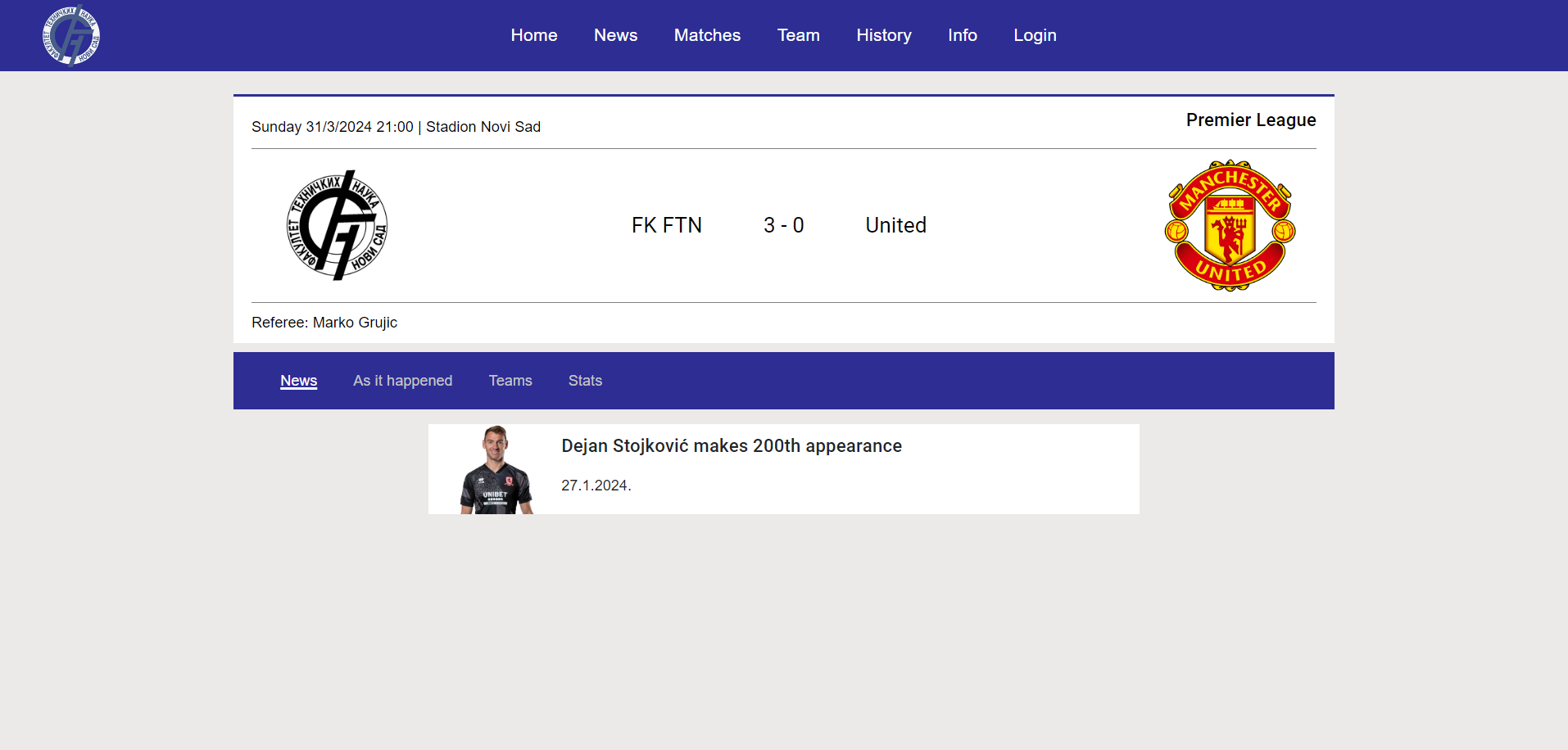


Слика 6.5 Приказ прегледа предстојећих мечева



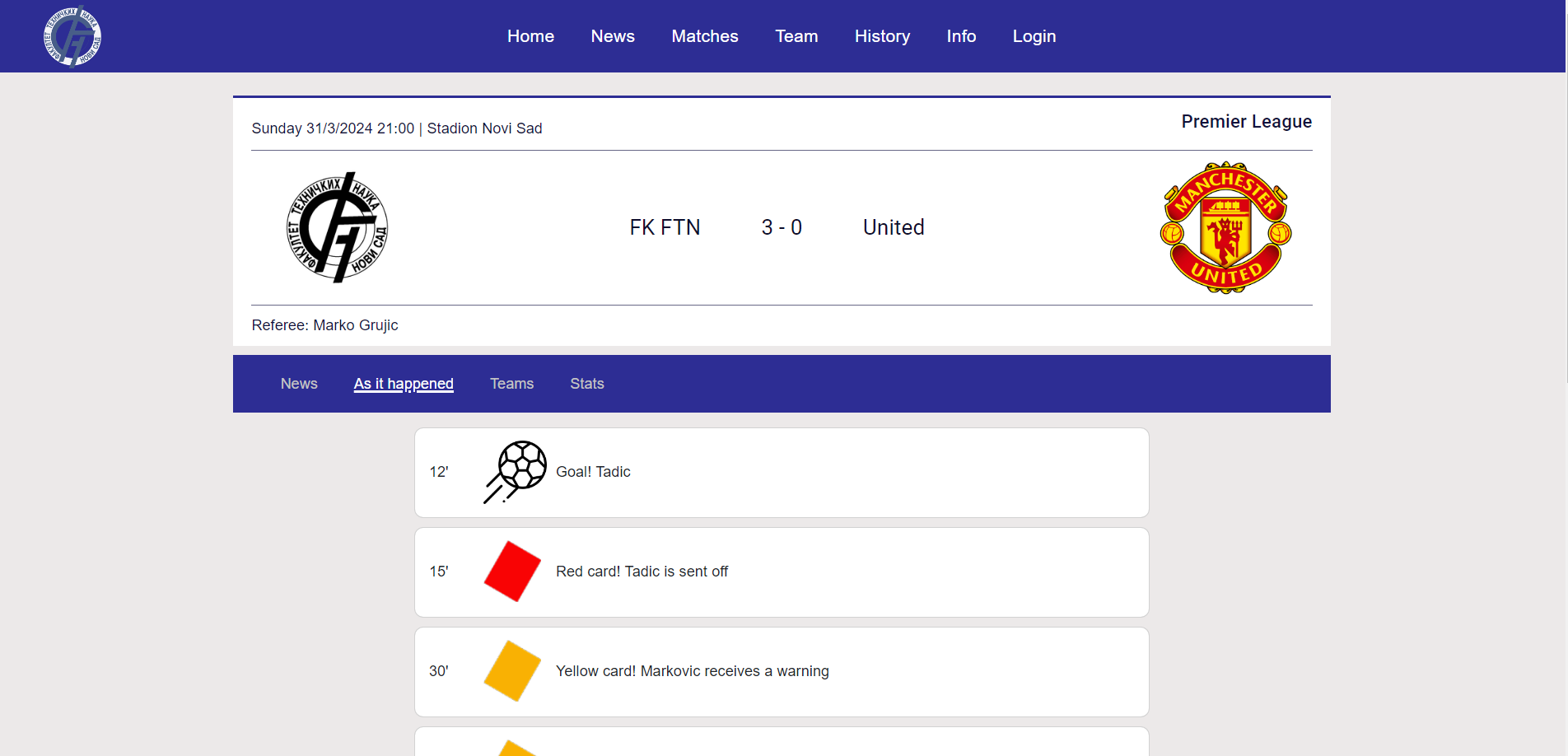
Слика 6.6 Приказ прегледа табела

Кликом на меч, корисник прелази на страницу на којој су приказане додатне информације у вези са мечом, као што су новости, догађаји на мечу, састави тимова и статистика. На слици 6.7 приказан је изглед претходно описане странице.

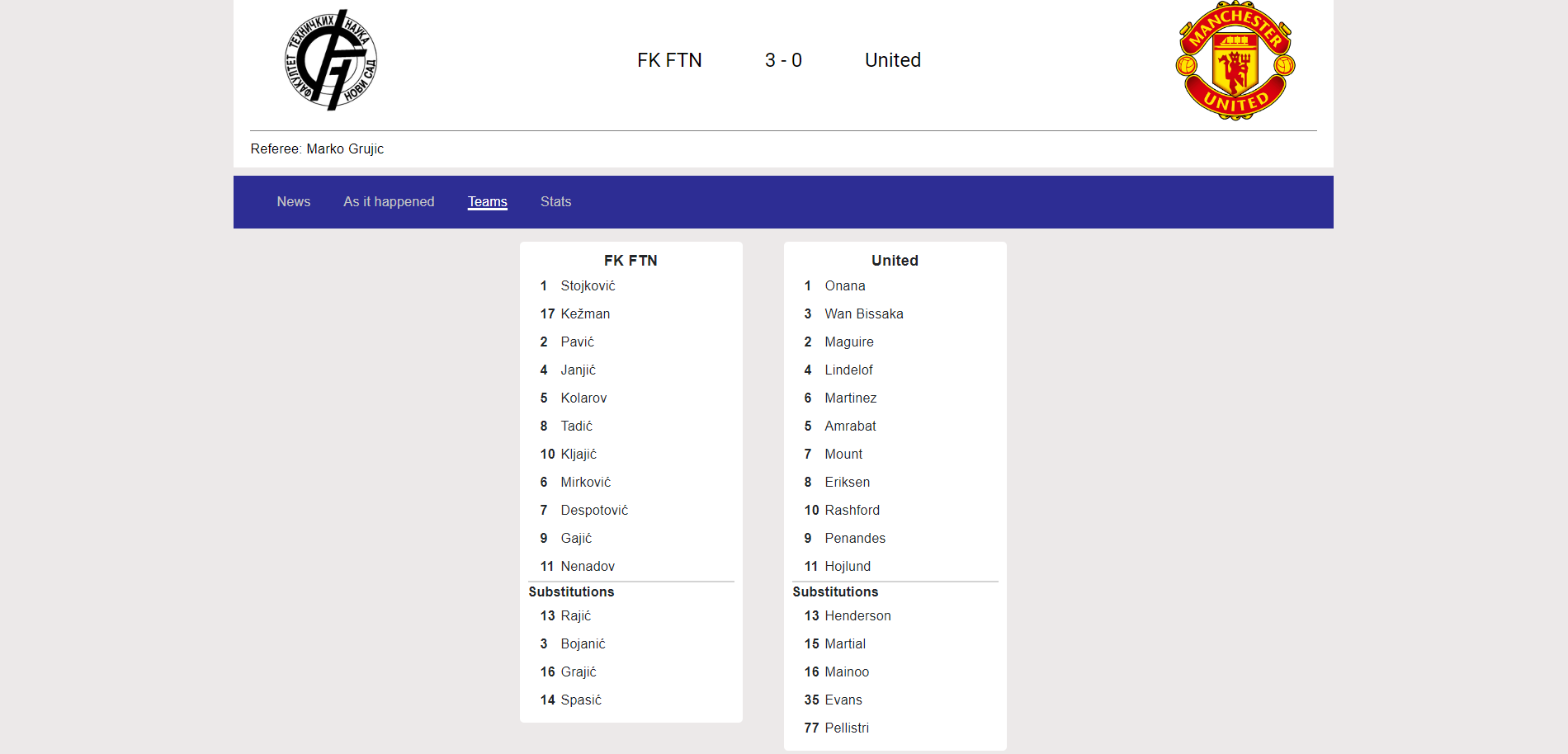


Слика 6.7 Детаљан приказ меча

На сликама 6.8 и 6.9 илустрован је приказ догађаја на мечу и приказ састава тимова за меч, редом. На ове странице стиже се кликом на дугмиће As it happened и Teams.



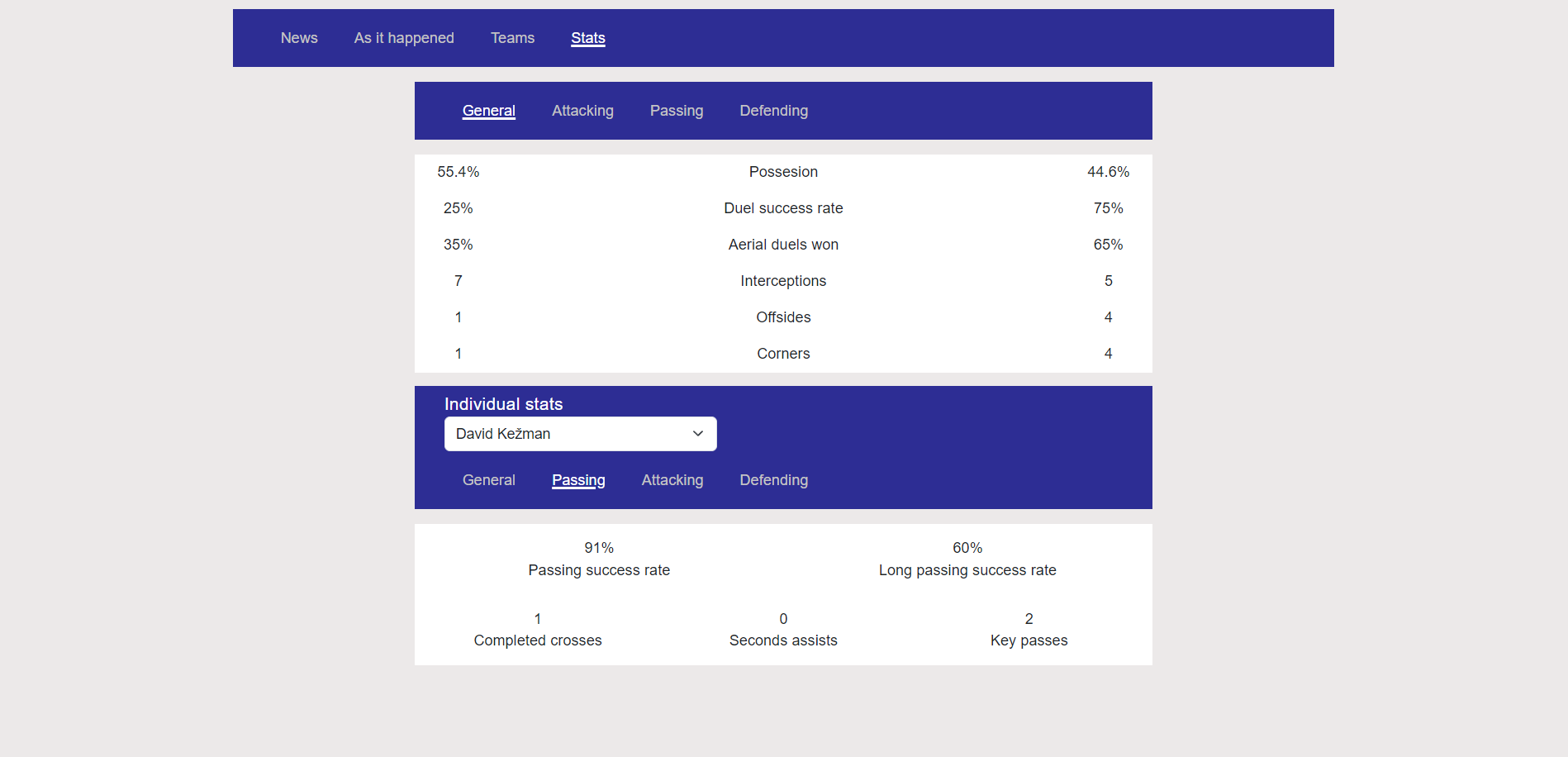
Слика 6.8 Приказ догађаја на мечу



Слика 6.9 Приказ састава тимова за меч

## Модул за преглед статистике

Избором меча и кликом на Statistics дугме на страници за детаљан преглед меча, кориснику се приказују статистике оба тима и индивидуална статистика играча подељене по категоријама ради боље прегледности. На слици 6.10 илустрован је изглед ове странице, док је преглед индивидуалне статистике играча током целе сезоне демонстриран у секцији везаној за модул за преглед профила играча.

 Слика 6.10 Приказ статистике меча

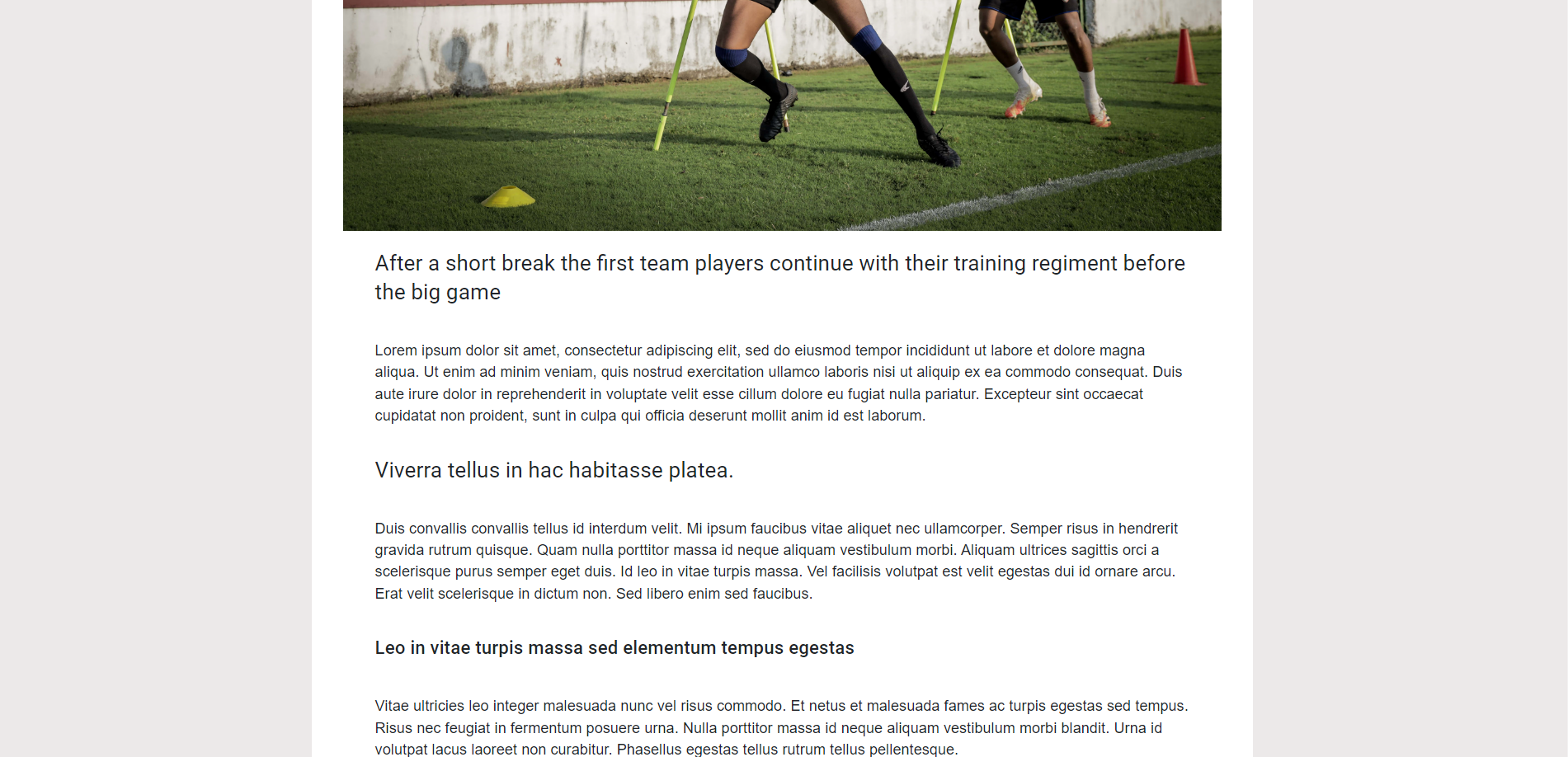
Администратор система има приступ страницама којим обични корисници немају. На слици 6.11 приказан је изглед странице за унос, односно измену, статистике меча.



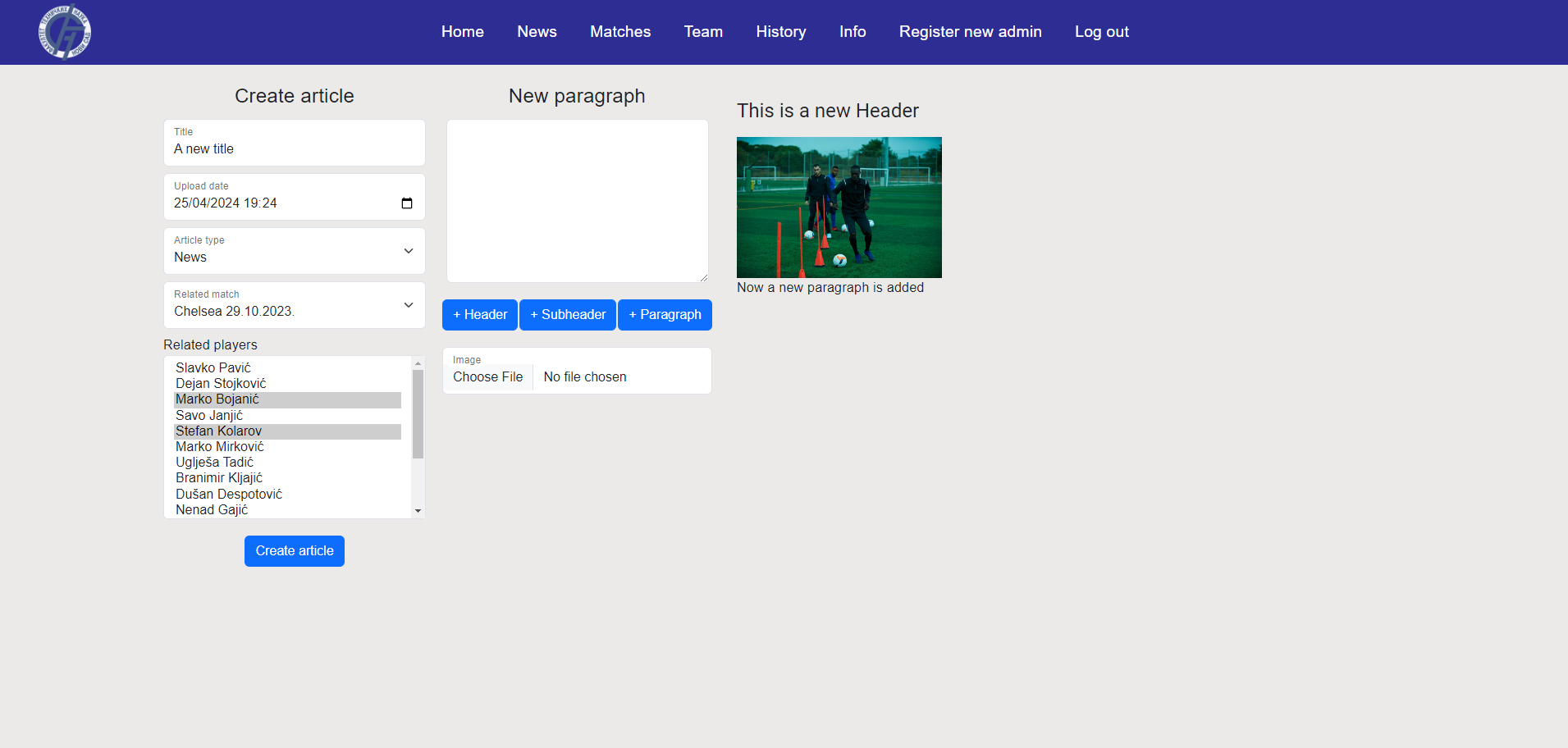
Слика 6.11 Приказ странице за измену статистике меча

## Модул за праћење новости

Кликом на дугмиће News, History и Info у оквиру навигационе траке, корисник на прелази на страницу са чланцима изабране категорије. Кликом на неки од чланака корисник прелази на страницу на којој је приказан чланак у целости. На слици 6.12 приказан је пример изгледа једног чланка.

Слика 6.12 Приказ изгледа чланка у целости

Администратор има могућност додавања нових чланака у систем. На слици 6.13 илустрован је изглед странице за креирање нових чланака.



Слика 6.13 Приказ странице за креирање чланака

# ЗАКЉУЧАК

Први и основни проблем који се решава у оквиру овог рада јесте креирање свеобухватног софтверског система за праћење резултата, статистика и свих других новости у вези са једним фудбалским клубом.

Ово је постигнуто на следећи начин. На серверској страни развијен је пројекат са називом FootballClubBackend који садржи приступне тачке ка спољашњем свету и омогућава комуникацију са клијентском апликацијом. Он такође садржи модел система са ентитетима и репозиторијуме помоћу којих се координише рад система и складиштење података. Осим тога садржи и апликативне сервисе који извршавају кључне операције везане за бизнис логику. Овим приступом постиже се јасна организација система и раздвајање одговорности између компоненти.

Детаљном анализом сличних система у другом поглављу рада постигнуто је боље схватање аспеката који чине веб сајт фудбалског клуба не само корисним и ефикасним, већ и јединственим. Ова имплементација се посебно истиче у организацији и структури система, што омогућава лаку скалабилност и одржавање. Сваки од модула се може развијати независно, могу се додавати нове функционалности или унапредити постојеће.

Овај систем је могуће унапредити даље проширавањем постојећих и имплементацијом нових функционалности. Једна од најкориснијих нових функционалности би била додавање могућности фановима да интерагују са клубом путем јавних гласања, коментара, постављањем питања и сл. Овај вид интеракције побољшава однос обожавалаца са клуб и повећава вероватноћу поновних посета веб сајту. Сем тога, било би добро подржати могућност промене језика и имплементацију функционалности за претрагу. Ове промене би могле значајно унапредити корисничко искуство.

Поред нових функционалности могле би се усавршити постојеће, попут приказа статистике. Повезивањем веб сајта са неким поузданим агрегатором статистике попут Опте [15], корисници би имали приступ већем број статистичких параметара и посао администратора би био олакшан. Проширивањем обима података би се такође могла обогатити визуелна презентација истих.

Закључно, овај рад представља темељ за развој свеобухватног софтверског система који служи за успостављање бољих односа фудбалског клуба са својим обожаваоцима. Кроз даље проширење и унапређење, могу се постићи још бољи резултати у циљу креирања веб сајта који пружа уникатно искуство и помаже фудбалском клубу да успостави упечатљиво присуство у онлајн свету.

# ЛИТЕРАТУРА

1. C# програмски језик – <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>
2. Веб сајт Арсенала – <https://www.arsenal.com/>
3. Веб сајт Партизана – <https://partizan.rs/sr>
4. .NET 6.0 – <https://learn.microsoft.com/en-us/dotnet/fundamentals/>
5. MongoDB – <https://www.mongodb.com/>
6. MongoDB Driver –

<https://www.mongodb.com/docs/drivers/csharp/v2.25/>

1. Angular framework – <https://angular.io/>
2. TypeScript програмски језик – <https://www.typescriptlang.org/>
3. SweetAlert2 – <https://sweetalert2.github.io/>
4. Use Cases – <https://en.wikipedia.org/wiki/Use_case>
5. PowerDesigner 16.7 – <https://www.powerdesigner.biz/>
6. Onion архитектура – <https://www.linkedin.com/pulse/onion-architecture-arshad-shahoriar-r6ehc>
7. CRUD – <https://www.codecademy.com/article/what-is-crud>
8. Swagger – <https://swagger.io/>
9. Opta – <https://www.statsperform.com/opta/>

# БИОГРАФИЈА

Лука Мандић рођен је 27.08.2000. у Новом Саду, где је стекао првих 6 година основног образовања у Основној школи Доситеј Обрадовић. У Гимназији Јован Јовановић Змај је завршио своје основношколско образовање, као и средњошколско на природно-математичком смеру. Школске 2019/20. Године се уписао на Факултет техничких наука на студијски програм Рачунарство и Аутоматика. Положио је све испите предвиђене планом и програмом и стекао услов за одбрану завршног рада.