

კვეკვესკირი თენგიზი

მაქანდარაშვილი გია

მირზაშვილი გიორგი

კლოიანი მანველ

კორახაშვილი ლუკა

ლაცაბიძე გიორგი

საბაშვილი ირაკლი

ურიდია დავითი

ქოჩლაძე გურამ

შონია საბა

ციყელაშვილი გიორგი

ჭანკვეტაძე მარიამი

ძიუდოს შეჯიბრის ტაბლოს პროგრამული უზრუნველყოფა

# სარჩევი

[სარჩევი 1](#_Toc518568853)

[პროექტის აღწერა 2](#_Toc518568854)

[ტექნიკური პარამეტრები და მახასიათებლები 3](#_Toc518568855)

[მთავარი გვერდი 4](#_Toc518568856)

[რეგიონების გვერდი 5](#_Toc518568857)

[სპორტსმენების გვერდი 5](#_Toc518568858)

[ტურნირების გვერდი 7](#_Toc518568859)

[ტურნირზე მონაწილე სპორტსმენების გვერდი 8](#_Toc518568860)

[ტურნირის შემადგენელი შეჯიბრებების გვერდი 9](#_Toc518568861)

[შეჯიბრის კონტროლის გვერდი 10](#_Toc518568862)

[შეჯიბრის გვერდი 11](#_Toc518568863)

[მომხმარებლების გვერდი 12](#_Toc518568864)

[ტაბლოს მოდელი 13](#_Toc518568865)

[შეწყვილების ალგორითმი 15](#_Toc518568866)

# პროექტის აღწერა

## შესავალი

აპლიკაცია გათვლილია საქართველოს ძიუდოს ფედერაციის მიერ ეროვნულ დონეზე ჩატარებულ შეჯიბრებზე, მისი მიმდინარეობისას საინფორმაციო ტაბლოს მართვასა და ინფორმაციული სახით ჩვენებაზე. ის ასევე აერთიანებს საქართველოში მოასპარეზე სპორტსმენებისა და ორგანიზებული ტურნირების ბაზას. აპლიკაცია შეიცავს ალგორითმს, რომლის დახმარებითაც შესაძლებელია ტურნირში მონაწილე სპორტსმენების შეწყვილების ცხრილის შედგენა, მოდიფიკაცია. აგრეთვე, მასში განთავსებულია მესამე ადგილის შეწყვილების ალგორითმი.

პროგრამის მოხმარება შეუძლია არაავტორიზებულ მომხმარებელს საინფორმაციო მიზნებით და ავტორიზებულ მომხმარებელს სხვადასხვა დონეზე სამართავად. აპლიკაცია გამიზნულია live რეჟიმზე. აქედან გამომდინარე, ნებისმიერ მომხმარებელს შეუძლია თვალი ადევნოს ტურნირის მსვლელობას დისტანციურად, ხოლო ტურნირზე დამსწრე მაყურებელს - ტაბლოს სახით.

## ძირითადი ამოცანები

* სისტემის ორგანიზება:

სისტემაში უნდა არსებობდეს მონაცემთა ბაზა, სადაც შენახული იქნება ინფორმაცია სპორტსმენებისა და ტურნირების შესახებ;

* ტურნირის შედგენის ამოცანა:

საწყის საფეხურზე იქმნება ტურნირი, შემდგომ აღნიშნულ შეჯიბრსვე ემატება მონაწილეები. შედეგის მიხედვით იქმნება შეწყვილების ცხრილი, რომლის მიხედვითაც ტარდება ტურნირი.

* შეწყვილების ალგორითმი. მესამე ადგილის შეწყვილების ცხრილის შედგენის ამოცანა

დამუშავების პროცესშია;

* ტაბლოს მართვა:

ტაბლოს მართვა დაფუძნებულია ძიუდოს შეჯიბრის პრინციპზე. შეჯიბრი მიმდინარეობს ორ სპორტსმენს შორის ძირითადი 4 წუთის განმავლობაში. თუ ძირითადი 4 წუთის ამოწურვის შემდგომ ვერ გამოვლინდა გამარჯვებული, ათვლას დაიწყებს დამატებითი დრო გამარჯვებულის გამოვლენამდე. გამარჯვება განისაზღვრება ორი ტიპის ქულათა სისტემით. 1. იპონი - მისი მნიშვნელობა შეიძლება იყოს 1 ან 0. თუ რომელიმე სპორტსმენი გააკეთებს იპონს შეჯიბრი სრულდება და გამარჯვებულად ცხადდება ზემოაღნიშნული მოჭიდავე. 2. ვაზარი - მისი მნიშვნელობა შეიძლება იყოს 0 ან ნებისმიერი ნატურალური რიცხვი. თითო ილეთის გაკეთება აისახება 1 ქულის მომატების სახით. გამარჯვებულია ის მონაწილე რომელიც მეტ ვაზარის ქულას დააგროვებს. ამას დამატებით ჯარიმის სახით თითოეულმა მონაწილემ შეიძლება მიიღოს მაქსიმუმ 3 გაფრთხილება. მესამე გაფრთხილებისას მონაწილე ეთიშება შეჯიბრს და გამარჯვებულად ცხადდება მისი ოპონენტი. მონაწილე შეიძლება ასევე დისკვალიფიცირდეს ტურნირის მიმდინარეობისას წონითი კატეგორიის დარღვევის გამო. ამ შემთხვევაშიც, გამარჯვებულად ცხადდება მისი ოპონენტი სპორტსმენი.

# ტექნიკური პარამეტრები და მახასიათებლები

* Programming Language: C#;
* Web Framework: ASP.NET MVC 5;
* Software Architecture: N-tier (Multilayered) Architecture;
* Design-Patterns: Inversion of Control, Dependency Injection;
* Dependency Principal (Design-Patterns Implementation) : Ninject;
* Database: MSSQL Server;
* Data Access Technology: Entity Framework;
* UI Library: Telerik Kendo UI MVC;
* Real-time Web Functionality => (Sockets): ASP.NET SignalR;
* Helper Library for File Generations: DevExpress;
* UI Theme Library: Keen Themes - Metrinic;
* Front-End Languages: HTML, CSS, Javascript, JQuery;
* Front-End Component Library: Bootstrap.

# მთავარი გვერდი

## აღწერა:

მთავარ გვერდს უნდა ჰქონდეს ვიზუალური დატვირთვა. მასზე უნდა განთავსდეს ანიმაციური სლაიდერი, რომელზეც გამოსახული იქნება ძიუდოსთან დაკავშირებული სურათები.

## უფლებები:

გვერდზე დაიშვებიან ავტორიზებული და არაავტორიზებული მომხმარებლები;

## ხილვადობა თავსართში:

* SuperAdmin: მთავარი, ტურნირები, სპორტსმენები, რეგიონები, მომხმარებლები;
* Admin: მთავარი, ტურნირები, სპორტსმენები, რეგიონები, მომხმარებლები;
* User: მთავარი, ტურნირები, სპორტსმენები;
* არაავტორიზებული მომხმარებელი: მთავარი, ტურნირები, სპორტსმენები.

## დამატებითი ინფორმაცია:

თავსართის ზემოთ განთავსებული უნდა იყოს ლოგო, დასახელება. იმ შემთხვევაში, თუ მომხმარებელი ავტორიზებულია - მომხმარებლიდან გამოსვლა და პაროლის შეცვლა.



# რეგიონების გვერდი

## აღწერა:

რეგიონების გვერდზე განთავსებული უნდა იყოს საქართველოში არსებული ყველა ქალაქის ჩამონათვალი, რომლებსაც შემდგომში სპორტსმენების სისტემაში დამატებისას მივუთითებთ.

## ფუნქციონალი:

* რეგიონის დამატება;
* არსებული რეგიონის ჩასწორება;
* არსებული რეგიონის წაშლა.

## უფლებები:

გვერდზე შესვლა და ფუნქციონალის გამოყენება შეუძლია მხოლოდ SuperAdmin-სა და Admin-ს.

## ვიზუალიზაცია:

ინფორმაციის გამოსატანად და მათ დასამუშავებლად გამოყენებული იქნება Telerik UI Grid Inline Editing - ის ფუნქციონალით.

## ვალიდაცია:

Client Side ვალიდაციისთვის გამოყენებული იქნება Remote Validation, რომელიც შეამოწმებს რომ არ მოხდეს ქალაქის სახელის დუბლირება.

## ლოგირება:

Action Log ცხრილში შეინახოს ყველა POST ტიპის მეთოდი (Add, Edit, Delete).

## ბაზის მოდელი:

* იდენტიფიკატორი: int Id [Key]
* დასახელება: string Name

ყველა ველი სავალდებულოა

# სპორტსმენების გვერდი

## აღწერა:

სპორტსმენების გვერდზე განთავსებული უნდა იყოს საქართველოში მოასპარეზე ყველა სპორტსმენის ჩამონათვალი, რომლებსაც შემდგომში ტურნირების სისტემაში სპორტსმენების დამატებისას მივუთითებთ.

## C:\Users\HP\Desktop\Docs\36596104_2059824644087619_7441865919401295872_n.png

## ფუნქციონალი:

* სპორტსმენის დამატება;
* არსებული სპორტსმენის ჩასწორება;
* არსებული სპორტსმენის წაშლა.

## უფლებები:

გვერდზე დაიშვებიან ავტორიზებული და არაავტორიზებული მომხმარებლები; ფუნქციონალის გამოყენება შეუძლია მხოლოდ SuperAdmin-სა და Admin-ს.

## ვიზუალიზაცია:

ინფორმაციის გამოსატანად და მათ დასამუშავებლად გამოყენებული იქნება Telerik UI Grid Window Editing - ის ფუნქციონალით.

## ლოგირება:

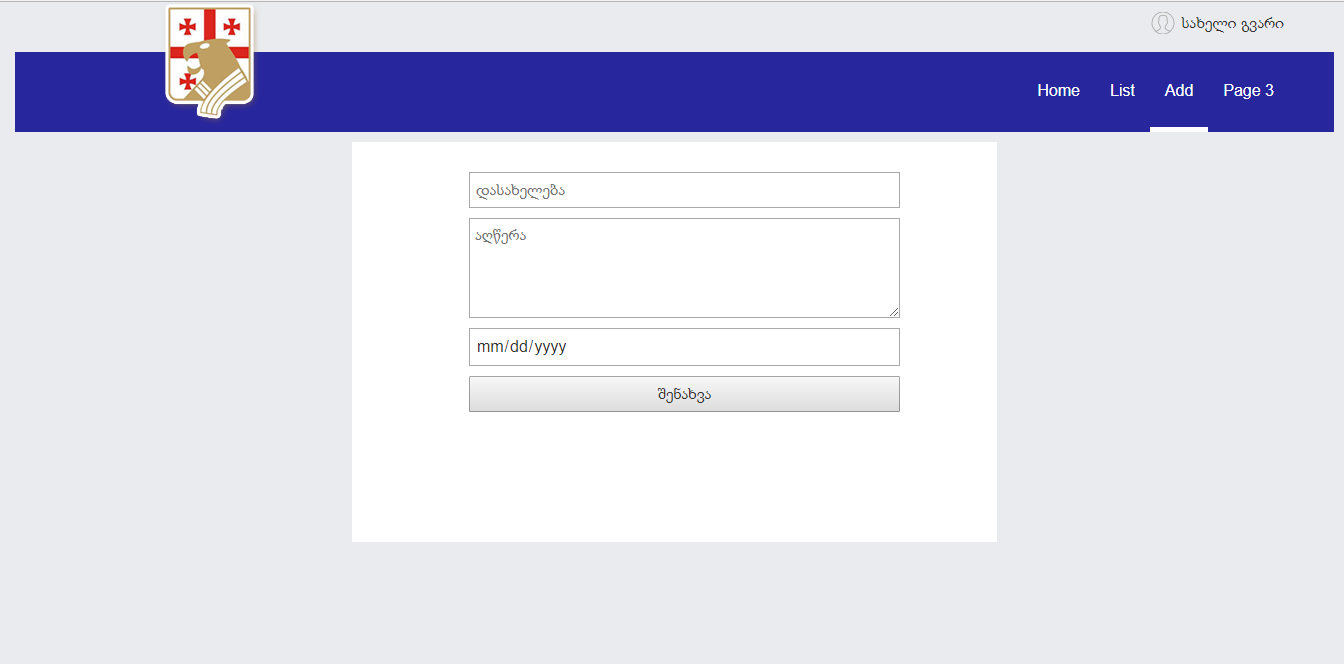
Action Log ცხრილში შეინახოს ყველა POST ტიპის მეთოდი (Add, Edit, Delete).

## ბაზის მოდელი:

* იდენტიფიკატორი: int Id [Key]
* სახელი: string Name
* გვარი: string Surname
* წონა: double Weight
* რეიტინგი: int Rank
* რეგიონი: RegionId

ყველა ველი სავალდებულოა

# ტურნირების გვერდი



## აღწერა:

ტურნირების გვერდზე განთავსებული უნდა იყოს საქართველოს ძიუდოს ფედერაციის მიერ საქართველოში ორგანიზებული ყველა ტურნირის ჩამონათვალი. ტაბების სახით უნდა გამოიყოს მიმდინარე, მომავალი და უკვე ჩატარებული ტურნირები. ადმინისტრატორს უნდა შეეძლოს ტურნირის მართვა (სპორტსმენების დამატება და შეწყვილების ცხრილის აგება), User-ს და არაავტორიზებულ მომხმარებელს მიმდინარე და მომავალი ტურნირის დეტალების(ინფორმაციის და შედგენილი შეწყვილების ცხრილის) დათვალიერება.

## ფუნქციონალი:

* ტურნირის დამატება;
* არსებული ტურნირის ჩასწორება;
* არსებული ტურნირის წაშლა;
* არსებული ტურნირის მართვა;
* არსებული ტურნირის დეტალური ინფორმაციის ნახვა.

## უფლებები:

გვერდზე დაიშვებიან ავტორიზებული და არაავტორიზებული მომხმარებლები;

* SuperAdmin: მიმდინარე (შესაძლებელია მოდიფიკაცია) , მომავალი (შესაძლებელია მოდიფიკაცია) და უკვე ჩატარებული ტურნირები;
* Admin: მიმდინარე (შესაძლებელია მოდიფიკაცია) , მომავალი (შესაძლებელია მოდიფიკაცია) და უკვე ჩატარებული ტურნირები;
* User: მიმდინარე და მომავალი ტურნირები;
* არაავტორიზებული მომხმარებელი: მიმდინარე და მომავალი ტურნირები.

## ვიზუალიზაცია:

ინფორმაციის გამოსატანად გამოყენებული იქნება 3 tab-ი (მომდინარე, მომავალი და უკვე ჩატარებული ტურნირები)-თვის. თითოეულისთვის გამოჩნდება შესაბამისი კონტენტი ჩამონათვალის სახით. მომხმარებლისთვის, რომლისთვისაც დაშვებულია მოდიფიკაცია, ტურნირის single layout-ზე უნდა იყოს განთავსებული მართვის ღილაკი. თითოეულ მათგანზე უნდა გამოჩნდეს ტურნირის დასახელება, აღწერა, თარიღი, წონითი კატეგორია და ტურნირის მონაწილეთა მაქსიმალური რაოდენობა.

## ვალიდაცია:

Client Side ვალიდაციისთვის გამოყენებული იქნება Remote Validation, რაც შეამოწმებს რომ არ მოხდეს ტურნირის სახელის დუბლირება.

## ლოგირება:

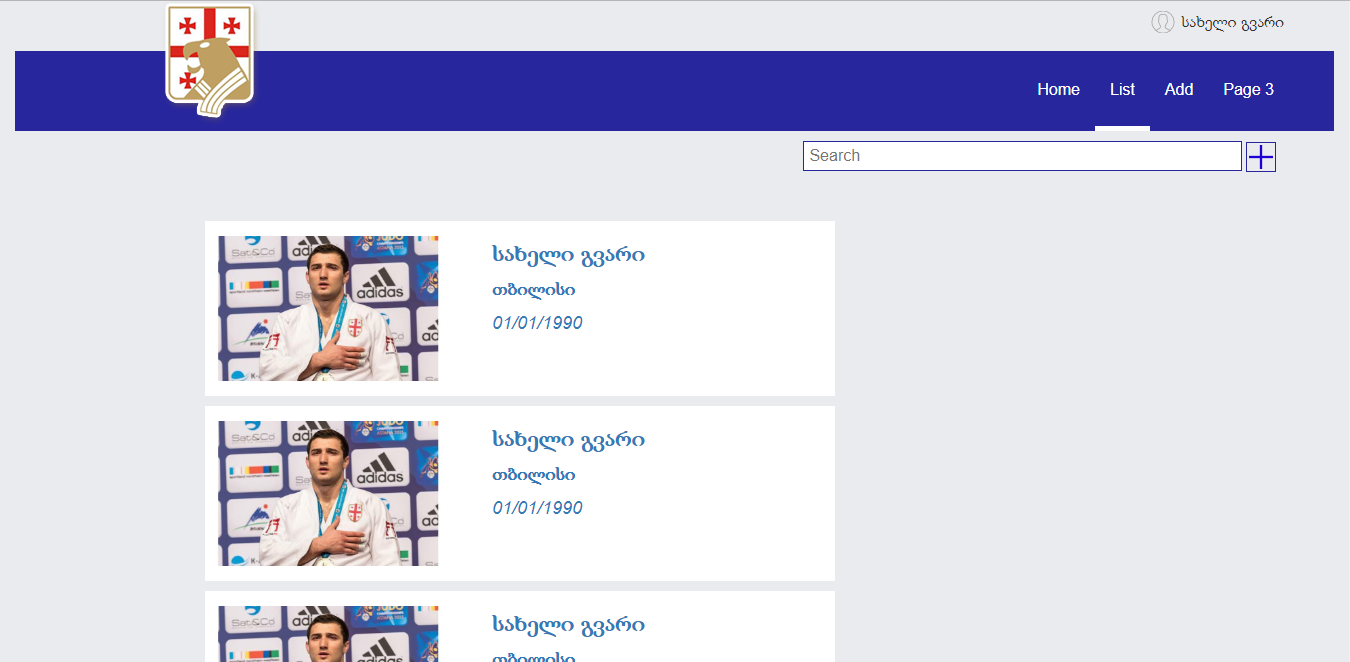
Action Log ცხრილში შეინახოს ყველა POST ტიპის მეთოდი (Add, Edit, Delete).

## ბაზის მოდელი:

* იდენტიფიკატორი: int Id [Key]
* დასახელება: string Name
* აღწერა: string Description
* მონაწილეთა მაქსიმალური რაოდენობა: int NumberOfContestants
* თარიღი: DateTime Date
* წონოთი კატეგორია: int Weight
* სტატუსი: int StatusId

ყველა ველი სავალდებულოა

# ტურნირზე მონაწილე სპორტსმენების გვერდი



## აღწერა:

ტურნირზე მონაწილე სპორტსმენების გვერდზე გადასვლა მოხდება ტურნირების გვერდიდან კონკრეტული ტურნირზე განთავსებული მართვის ღილაკზე დაჭერით. გვერდზე განთავსებული იქნება აღნიშნულ ტურნირზე მონაწილე ყველა სპორტსმენის ჩამონათვალი და შესაძლებელი იქნება მათი მოდიფიკაცია ტურნირის დაწყებამდე. სპორტსმენების დამატების შემდგომ შესაძლებელი იქნება ტურნირის შედგენა ღილაკზე დაჭერით, რითიც მოხდება სპორტსმენების დაწყვილება და შეწყვილების ცხრილის შედგენა სპეციალური ალგორითმის მეშვეობით.

## ფუნქციონალი:

* არსებულ ტურნირზე სპორტსმენის დამატება, განთესვა;
* არსებულ ტურნირზე სპორტსმენის წაშლა;
* არსებული ტურნირის მართვა;
* არსებული ტურნირის დეტალური ინფორმაციის ნახვა;
* არსებული ტურნირის შეწყვილების ცხრილის აგება.

## უფლებები:

გვერდზე შესვლა და ფუნქციონალის გამოყენება შეუძლია მხოლოდ SuperAdmin-სა და Admin-ს.

## ვიზუალიზაცია:

ინფორმაციის გამოსატანად და მათ დასამუშავებლად გამოყენებული იქნება Telerik UI Grid Window Editing - ის ფუნქციონალით.

## ვალიდაცია:

Client Side ვალიდაციისთვის გამოყენებული იქნება Remote Validation, რაც შეამოწმებს რომ არ მოხდეს ტურნირზე სპორტსმენის დუბლირება.

## ლოგირება:

Action Log ცხრილში შეინახოს ყველა POST ტიპის მეთოდი (Add, Edit, Delete).

## ბაზის მოდელი:

* იდენტიფიკატორი: int Id [Key]
* ტურნირის იდენტიფიკატორი: int TournamentId
* სპორტსმენის იდენტიფიკატორი: int SportsmanId
* განთესვა: bool Seeded

ყველა ველი სავალდებულოა

# ტურნირის შემადგენელი შეჯიბრებების გვერდი

## აღწერა:

ტურნირის შემადგენელი შეჯიბრებების გვერდზე განთავსებულია დეტალური ინფორმაცია ტურნირის შესახებ და ასევე შეწყვილების ცხრილი, თუ ის უკვე შედგენილია. მასზე გამოსვლა ხდება ტურნირების გვერდიდან კონკრეტული ტურნირზე განთავსებული დეტალების ღილაკზე დაჭერით. იმ შემთხვევაში, თუ ტურნირი ჯერ არ დაწყებულა ადმინისტრატორის მიერ შესაძლებელია შეწყვილების ცხრილის შედგენა ტურნირზე მონაწილე სპორტსმენების გვერდზე „ტურნირის შედგენის’ ღილაკზე დაჭერით. ასევე აღნიშნულ გვერდზე შესაძლებელი უნდა იყოს სპორტსმენების გადაადგილება შეწყვილების ცხრილში.

## ფუნქციონალი:

* არსებულ შეწყვილების ცხრილში სპორტსმენების გადაადგილება.

## უფლებები:

გვერდზე დაიშვებიან ავტორიზებული და არაავტორიზებული მომხმარებლები; ფუნქციონალის გამოყენება შეუძლია მხოლოდ SuperAdmin-სა და Admin-ს.

## ვიზუალიზაცია:

ინფორმაციის სახით უნდა გამოჩნდეს ტურნირის შესახებ დეტალური ინფორმაცია და აიგოს შეწყვილების ცხრილის ვიზუალური მხარე.

## ლოგირება:

Action Log ცხრილში შეინახოს ყველა POST ტიპის მეთოდი.

## ბაზის მოდელი:

* იდენტიფიკატორი: int Id [Key]
* ტურნირის იდენტიფიკატორი: int TournamentId
* შეჯიბრის იდენტიფიკატორი: int WrestleId

ყველა ველი სავალდებულოა

# შეჯიბრის კონტროლის გვერდი

## აღწერა:

შეჯიბრის კონტროლის გვერდზე განთავსებულია ორ სპორტსმენს შორის შეჯიბრის მართვის ფუნქციონალი და ინფორმაციის გამოტანა.

## ფუნქციონალი:

* სპორტსმენებისთვის იპონის ქულის დამატება/დაკლება/გამოჩენა;
* სპორტსმენებისთვის ვაზარის ქულის დამატება/დაკლება/გამოჩენა;
* სპორტსმენებისთვის გაფრთხილების ქულის დამატება/დაკლება/გამოჩენა;
* შეჯიბრის დროის დაწყება, გაჩერება, განულება;
* მოქმედების ტექსტის დამატება/წაშლა/გამოჩენა;
* მოქმედების ტექსტის ფაილად გენერირება.

## უფლებები:

გვერდზე შესვლა და ფუნქციონალის გამოყენება შეუძლია მხოლოდ ავტორიზებულ მომხმარებელს.

## ლოგირება:

Action Log ცხრილში შეინახოს ყველა POST ტიპის მეთოდი.

## დამატებითი ინფორმაცია:

ფუნქციონალის ქმედებების შესრულებისას ინფორმაცია უნდა შეინახოს ბაზაში და აისახოს როგორც შეჯიბრის კონტროლის გვერდზე ასევე შეჯიბრის გვერდზე, SignalR - ის გამოყენებით.

## ბაზის მოდელი:

* იდენტიფიკატორი: int Id [Key]
* პირველი სპორტსმენის იდენტიფიკატორი: int SportsmanOneId
* მეორე სპორტსმენის ერთის იდენტიფიკატორი: int SportsmanTwoId
* პირველი სპორტსმენის ვაზარის ქულა: int SporstmanOneWazariPoint
* მეორე სპორტსმენის ვაზარის ქულა: int SporstmanTwoWazariPoint
* პირველი სპორტსმენის იპონის ქულა: bool SporstmanOneIppon
* მეორე სპორტსმენის იპონის ქულა: bool SporstmanTwoIppon
* პირველი სპორტსმენის ჯარიმის ქულა: int SportmanOneFine
* მეორე სპორტსმენის ჯარიმის ქულა: int SportmanTwoFine
* შეჯიბრის დრო: Time time
* დროის მიმართულება: bool Decreesing
* სტატუსი: int statusId
* გამარჯვებული სპორტსმენის იდენტიფიკატორი: int WinnerId

ყველა ველი სავალდებულოა

# შეჯიბრის გვერდი

## აღწერა:

შეჯიბრის გვერდზე განთავსებულია ორ სპორტსმენს შორის შეჯიბრის ინფორმაციის გამოტანა.

## ფუნქციონალი:

* ინფორმაციის გამოჩენა

## უფლებები:

გვერდზე დაიშვებიან ავტორიზებული და არაავტორიზებული მომხმარებლები;

# მომხმარებლების გვერდი

## აღწერა:

მომხმარებლების გვერდზე უნდა განთავსდეს სისტემაში არსებული ყველა მომხმარებლების ჩამონათვალი.

## ფუნქციონალი:

* მომხმარებლის დამატება;
* არსებული მომხმარებლის რედაქტირება;
* არსებული მომხმარებლის წაშლა;

## უფლებები:

გვერდზე შესვლა და ფუნქციონალის გამოყენება შეუძლია მხოლოდ Administrator მომხმარებელს.

* SuperAdmin: ყველა ტიპის მომხმარებელზე მოდიფიკაცია;
* Admin: User ტიპის მომხმარებლებზე მოდიფიკაცია;

## ვიზუალიზაცია:

ინფორმაციის გამოსატანად და მათ დასამუშავებლად გამოყენებული იქნება Telerik UI Grid Window Editing - ის ფუნქციონალით.

## ვალიდაცია:

Client Side ვალიდაციისთვის გამოყენებული იქნება Remote Validation, რომელიც შეამოწმებს რომ არ მოხდეს მომხმარებლების დუბლირება.

## ლოგირება:

Action Log ცხრილში შეინახოს ყველა POST ტიპის მეთოდი (Add, Edit, Delete).

## ბაზის მოდელი:

* იდენტიფიკატორი: string Id [Key]
* სახელი: string FirstName
* გვარი: string LastName
* მომხმარებლის სახელი: string UserName
* პაროლი: string PasswordHash
* სტატუსი: bool IsDeleted

ყველა ველი სავალდებულოა

# ტაბლოს მოდელი

როგორც ზემოთ ავღნიშნეთ ძიუდოს შეჯიბრის ტაბლოს პროგრამული უზრუნველყოფის ამოცანაში მოითხოვებოდა ორი გვერდი, უშუალოდ ტაბლო და მისი სამართავი პანელი. სამართავ პანელზე ასახული ცვლილებები ავტომატურად უნდა ასახულიყო ტაბლოზე გვერდის გადატვირთვის გარეშე. ამასთანავე ცვლილებები უნდა განხორციელებულიყო მონაცემთა ბაზაში. შეწყვილების ალგორითმის შედეგად, პროგრამა ირჩევს ორ შესაფერის სპორტსმენს. ტაბლოზე გამოსახულია სპორტსმენის პირადი ინფორმაცია: სახელი, გვარი, რეგიონი და წონითი კატეგორია, აგრეთვე იპონის, ან ვაზარის ამსახველი ქულა. აგრეთვე გვაქვს წამზომი, რომლის ათვლა იწყება 4:00 წუთიდან და კლებით ჩამოდის 0:00-მდე. ამ 4 წუთის განმავლობაში თუ გამარჯვებული არ გამოვლინდა, თვლა იწყება 0:00-დან ზრდადი მიმართულებით, სანამ არ გამოვლინდება გამარჯვებული. თუ ზრდადი დროის პერიოდში რომელიმე სპორტსმენი მოიპოვებს ვაზარის ქულას, იგი ავტომატურად გამოცხადდება როგორც გამარჯვებული. ტაბლოზე არსებობს გაფრთხილების ბარათებიც, რომლებიც თავიდან არ ჩანს. იმ შემთხვევაში, თუ რომელიმე სპორტსმენი დაარღვევს წესს, ბარათი გამოისახება ტაბლოზე. თუ სპორტსმენი მიიღებს სამ გაფრთხილებას, იგი ავტომატურად აგებს. გამარჯვების შემთხვევაში, ტაბლოზე გამოდის ფანჯარა, სადაც წერია გამარჯვებული სპორტსმენის სახელი და გვარი.

## https://scontent.ftbs4-1.fna.fbcdn.net/v/t1.15752-9/36640281_1789983327714219_3098904803409920000_n.png?_nc_cat=0&oh=b2250e8b0d95496037e60a0a8d1cf89d&oe=5BE4166B

## აღწერა:

ტაბლოს და პანელის დიზაინისთვის უმეტეს წილად გამოვიყენეთ Bootstrap 3 ტექნოლოგია. ეს ტექნოლოგია ავირჩიეთ იქიდან გამომდინარე, რომ ყველაზე მოსახერხებელ და ოპტიმალურ ტექნოლოგიად გვესახებოდა მოცემული სამუშაოს შესასრულებლად. ძირითადი მიზეზი რაშიც Bootstrap გამოგვადგა, იყო Responsive დიზაინი, რაც საშუალებას გვაძლევს, რომ გამოსახულება შესაბამისად მოერგოს სხვადასხვა რეზოლუციის მქონე კომპიუტერულ მოწყობილობას, იქნება ეს ტაბლო, პლანშეტი, თუ პერსონალური კომპიუტერი. მაგრამ, მიუხედავად იმისა, რომ Bootstrap ტექნოლოგია ბევრ ასპექტში დაგვეხმარა, იგი საკმარისი არ იყო ზოგიერთი სტრუქტურული და სტილისტური დეტალების შესასრულებლად, რისთვისაც უკვე გამოვიყენეთ HTML5 და CSS3 ტექნოლოგიები. სტრუქტურა ძირითადად აწყობილია HTML5 და მასში ინტეგრირებული Bootstrap 3 ტექნოლოგიებით. ეს უკანასკნელი გვაძლევს საშუალებას კომპაქტურად და მოსახერხებლად ავაგოთ დიზაინის სტრუქტურა, ნაკლები კოდის გამოყენებით. ყველაზე მეტად დაგვეხმარა იმაში, რომ დიზაინი ყოფილიყო მაქსიმალურად Responsive. Bootstrap 3-ით, ასევე სტილისტურად გავაფორმეთ აგებული სტრუქტურა, მაგრამ პატარა დეტალების გამო, რომელთა უზრუნველყოფაც მან ვერ შეძლო, გამოვიყენეთ СSS 3 ტექნოლოგია. CSS3-ის მეშვეობით მოვახდინეთ ტექსტის ზომების სწორად შერჩევა და პოზიციონირება, ფონის ფერების შერჩევა, ტექსტის Responsive-ობა vw ერთეულებით, ბარათების გამოსახვა, პოზიციონირება და ფერები, წონის და ტაიმერის ცენტრში პოზიციონირება.

## ფუნქციონალი:

ტაბლოს ფუნქციონალისათვის საჭირო გახდა გამოგვეყენებინა რეალურ დროში მონაცემთა გაცვლისა და ვიზუალიზაციის ტექნოლოგია, ASP.NET Razor სინტაქსი, Javascript და JQuery.

არსებობს რამოდენიმე გზა, რათა განხორციელდეს რეალურ დროში კომუნიკაცია Web აპლიკაციებში, როგორიცაა: Long Pooling, Interval Pooling, Server Sent Events, Web Sockets და სხვა. მათ შორის HTML5 Web Sockets არის ერთ-ერთი ყველაზე პოპულარული გზა. Web Sockets-მა, რომ გამართულად იმუშაოს მას სჭირდება როგორც კლიენტური მხარდაჭერა, ასევე სერვერულიც. თუ რომელიმე მხარს არ უჭერს Web Sockets-ს, რეალურ დროში კომუნიკაცია არ დამყარდება. ასეთ შემთხვევაში ჩვენ გვჭირდება PolyFill, რომელიც მოიცავს რამდენიმე Fallback მექანიზმს. ASP.NET SignalR უზრუნველყოფს ამ ფუნქციას უფასოდ. თუ ჩვენ გამოვიყენებთ SignalR-ს, ჩვენ აღარ მოგვიწევს ხელით აღმოვაჩინოთ ტექნოლოგიები, რომლებიც გაშვებულ პლათფორმას მხარდაჭერას უწევენ. SignalR-ი ყველა არხულ ლოგიკას აბსტრაგირებს, რომ აღმოაჩინოს ყველა მათგანი და აირჩიოს საუკეთესო. SignalR-ი ამოწმებს და იყენებს ერთ-ერთ ხერხს ქვემოთ მოცემული სიიდან:

* Web Sockets: HTML5 რეალურ დროში კომუნიკაციის ფუნქცია, რომელიც იყენებს სპეციალურ პროტოკოლს ‘ws’
* Server Sent Events: HTML5 რეალურ დროში კომუნიკაციის ფუნქცია HTTP-სა და HTTPS-ზე
* Forever Frames: ამატებს დამალულ iFrames-ს გვერდზე და მართავს კომუნიკაციას ამ Frame-ის გამოყენებით
* Long Pooling: კლიენტი გამუდმებით რაიმე ინტერვალით ითხოვს ინფორმაციას ბაზიდან, რათა მუდმივად უახლეს ინფორმაციას ფლობდეს.

ბაზასთან ურთიერთობისათვის გამოვიყენეთ ASP.NET Entoty Framework. ეს იყო ყველაზე ოპტიმალური მიდგომა, ვინაიდან ავირჩიეთ Code First, რომელიც მდგომარეობს იმაში, რომ, პირველ რიგში, იწერება აპლიკაცია და იქმნება მოდელები და ამის შემდგომ გენერირდება ბაზა. ეს Code First მიდგომა საბაზისოდ იყენებს Entity Framework-ს. შესაბამისად, ამ Framework-ის გამოყენება ყველაზე ოპტიმალური გამოსავალი იყო.

ASP.NET-ამ ეს პლატფორმამ საშუალება მოგვცა, რომ ყველა ზემოთხსენებულ ტექნოლოგიას ემუშავა სინქრონულად და ამოცანის ყოველი პუნქტი დაეკმაყოფილებინა.

## სისტემური მოდელი:

ძიუდოს შეჯიბრის ტაბლოს პროგრამული უზრუნველყოფის ამოცანა, პირველ რიგში, თემატურად დავყავით სამ ნაწილად: ტაბლოს და სამართავი პანელის დიზაინი, ფუნქციონალი და ბაზასთან ურთიერთობა. ამის შემდეგ თითოეული ნაწილი დავამუშავეთ და შევქმენით მათი მოდელები.

დიზაინის მხარეს, ფოტოშოპში შევქმენით კოცეპტუალური მოდელი და შემდეგ ეს მოდელი განვახორციელეთ HTML5, CSS3 და Bootstrap 3-ის მეშვეობით.

ტაბლოსა და სამართავ პანელზე გამოტანილია სპორტსმენის და თამაშის მონაცემები. ამ მონაცემებს ვიღებთ მონაცემთა ბაზიდან. სამართავ პანელში ხდება თამაშიდან გამომდინარე შესაბამისი ცვლილება. მონაცემთა ბაზაში არსებული მონაცემების ცვლილება აისახება სამართავ პანელზე და ტაბლოზე ერთდროულად. შესაბამისად სამართავ პანელში შეცვლილი მონაცემები აგრეთვე ახლდება ბაზაში.

პირობითად, მატჩის დროს ინფორმაცია გამოდის სამართავ პანელზე, შემდეგ, თამაშის პროცესში, მომხარებლის მიერ შეტანილი ცვლილებები აისახება ბაზაში და შესაბამისად განახლდება ტაბლოზე გამოტანილი ინფორმაციაც.

# შეწყვილების ალგორითმი

## აღწერა:

ძიუდოს შეჯიბრის აპლიკაციას ესაჭიროება შეჯიბრის მონაწილეების შეწყვილება მათი ძალების მიხედვით. შეწყვილების ისეთი სქემაა საჭირო რომლითაც შეჯიბრებაში მონაწილეთაგან უძლიერესები ტურნირში ერთმანეთს შეხვდებიან რაც შესაძლებელია გვიან, ტურნირის დასასრულში.

ამოცანის დასმა - შემოდის მონაცემები ტურნირის მონაწილეთა შესახებ. საჭიროა მათი სიძლიერის (შედარებადი ველი) მიხედვით დახარისხება/შეწყვილება და ისეთი სახით დაბრუნება რომ გარკვევადი იყოს თუ ვინ ვის ეჯიბრება და რა ეტაპზეა ტურნირის დაწყებამდე:

* ნორმალური შემთხვევა - როდესაც მონაწილე კარგად ზის ტურნირის შეხვედრათა ცხრილში.
* დამატებითი შემთხვევა - შეჯიბრთა ორობითობისა და მონაწილეთა რაოდენობიდან გამომდინარე შესაძლოა ზოგიერთ მონაწილეს ე.წ. დამატებითი შეხვედრების ჩატარება.

გამოყენებული ტექნიკა - ალგორითმი მონაწილეთა შეწყვილებისათვის და  
მონაცემთა ადვილი აღმისათვის იყენებს ორობითი ხის სისტემას. იგი დაწერილია C# და იყენებს .Net-ის სტანდარტულ ბიბლიოთეკებს მონაცემთა სიაში შენახვისა და დასორტირებისთვის.

## ფუნქციონალი:

**მონაწილეთა შეწყვილება**

თავდაპირველად საჭიროა მონაწილეთა რაოდენობის მიხედვით გამოთვლილი რიცხვის შესაბამისი მქონე ცარიელი ორობითი ხის ინიციალიზება.  
მთავარი ხის სიღრმე გამოითვლება მონაწილეთა რაოდენობიდან ორობითი ფუძის ლოგარითმის ამოღებით და მთელობამდე ნაკლებობით დამრგვალებით:

Depth = floor( log( n ));

შემდგომ თვით შეწყვილებას სანამ დავიწყებთ საჭიროა კიდევ ერთი მონაცემის, დამატებითი შეხვედრების რაოდენობის განსაზღვრა. იძებნება უახლოესი, მცირე, 2 ის ხარისხი რიცხვი და იგი აკლდება მონაწილეთა რაოდენობას. ორობით ხეში ყოველი დამატებითი შეხვედრა არა ერთი, არამედ ორი დამატებითი კვანძის გაჩენას იწვევს, შედეგად საჭირო დამატებით კვანძთა რაოდენობა ორმაგდება.

Additional = 2 \* (n – 2floor( log(n ));

ახლა ჩვენ უკვე შეგვიძლია დავიწყოთ მონაწილეთა სიის დამუშავება მათი შეწყვილების მიზნით. ზევით ინიციალიზებული ცარიელი ხე, მონაწილეთა სია და დამატებით კვანძთა რაოდენობა გადაეცემა ფუნციას რომელიც გარკვეული სახით ორად ხლეჩს სიას და რეკურსიულად იძახებს თავის თავს მარცხენა და მარჯვენა შვილებისთვის ყოველი ახლადშექმნილი სიით. რეკურსიული გამოძახება მოხდება მანამ, სანამ გადმოცემული სია არ ჩამოვა ერთ ელემენტამდე, ანუ მივაღწევთ ხის ფოთლამდე, ამ შემთხვევაში ხდება მონაწილის მონაცემის შენახვა კვანძში და ფუნქციის დასრულება. განსაკუთრებული ყურადღება სჭირდება დამატებითი კვანძების შექმნას, ყოველთვის როდესაც ვცდილობთ გადავცეთ მონაწილეთა სია შვილებს, მაგრამ აღმოჩნდება რომ ასეთი შვილი კვანძი არ იყო განსაზღვრული საწყისად ინიციალიზებულ ხეში, ეს ნიშნავს რომ ასეთი კვანძი არის დამატებითი შეჯიბრისათვის, შესაძლებლობისდაგვარად ემატება ახალი კვანძი და რიცხვითი ცვლადი მცირდება ერთით. სიის გახლეჩა - ყოველ ბიჯზე შემომავალი სია უნდა იყოს დახარისხებული (ზრდადობით ან კლებადობით); იქმნება ორი დამატებითი, ცარიელი სია მარცხენა და მარჯვენა შვილებისთვის გადასაცემად. სიაზე ხორციელდება ინდექსირებული ჩამოვა ( for ციკლი ), ინდექსატორი მონაცვლეობს 0-დან (საწყისი) სიაში ჩანაწერთა რაოდენობა**მდე** (არჩათვლით) ერთის ბიჯით; შემდეგი მანიპულაცია მარტივია, ყოველი სიის ელემენტის ინდექსი მოწმდება ლუწობა კენტობაზე და შედეგის მიხედვით ემატება სიას მარცხენა ან მარჯვენა შვილისათვის.

სიის დამუშავების ალგორითმის ფსევდოკოდი:

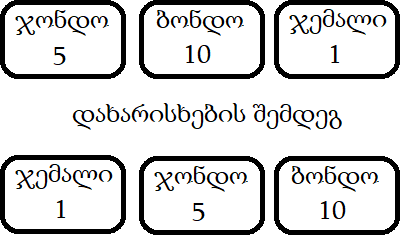
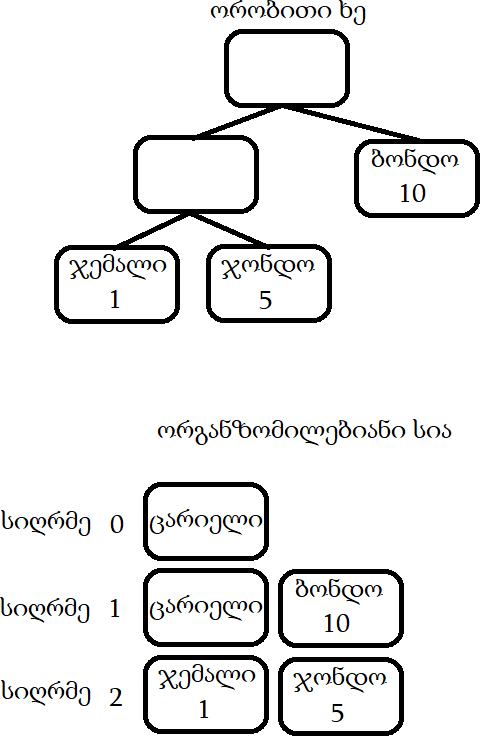
დაამუშავე\_სია( კვანძი, სია, დამატებითი )  
{  
 თუ სია შეიცავს ერთ ელემენტს   
 მაშინ   
 მიანიჭე კვანძს სიის ერთადერთი ელემენტი  
 დაასრულე ფუნქცია  
 შექმენი ცარიელი სია მარცხენასთვის  
 შექმენი ცარიელი სია მარჯვენასთვის  
 ინდექსი ჩამოივლის მთელ რიცხვებს 0 დან სიის ელემენტთა რაოდენობამდე  
 ყოველი ინდექსის მნიშვნელობისთვის  
 თუ ინდექსი ლუწია  
 მაშინ   
 სიის ინდექსის შესაბამისი ელემენტი დაამატე მარცხენასთვის სიას  
 წინააღმდეგ  
 სიის ინდექსის შესაბამისი ელემენტი დაამატე მარჯვენასთვსი სიას  
 თუ საჭიროა კვანძის მარცხენა შვილის დამატება და დამატებითი დადებითია  
 მაშინ  
 დაამატე კვანძის მარცხენა შვილი ცარიელი მონაცემით  
 დამატებითი შეამცირე ერთით.  
 თუ საჭიროა კვანძის მაჯვენა შვილის დამატება და დამატებითი დადებითია  
 მაშინ  
 დაამატე კვანძის მარჯვენა შვილი ცარიელი მონაცემით  
 დამატებითი შეამცირე ერთით.  
 დაამუშავე\_სია(კვანძის მარცხენა შვილი, მარცხენასთვის, დამატებითი)  
 დაამუშავე\_სია(კვანძის მარჯვენა შვილი, მარჯვენასთვის, დამატებითი)  
}

აღწერილი ალგორითმი თავისი სიმარტივის მიუხედავად მშვენივრად ასრულებს საკუთარ დანიშნულებას და მონაწილეები გადანაწილდებიან ხეზე იმგვარად, რომ ყველზე ძლიერთა შეხვედრა მოხდება რაც შესაძლებელია გვიან; და ეს არ ეხება მხოლოდ 2 უძლიერეს მონაწილეს, ყოველი წყვილისთვის უძლიერესებისა, მათი შეხვედრა იქნება შეძლებისდაგვარად უგვიანესი!

## სისტემური მოდელი:

ორობითი ხე - ორობითი ხე არის მონაცემთა სტრუქტურა რომლის მონაცემები მოთავსებულია კვანძებში. ყოველი კვანძი შეიცავს ინფორმაციას ხეში საკუთარი სიღრმის, მშობელი კვანძის, მარცხენა და მარჯვენა შვილი კვანძების შესახებ. ყოველ ხეს გააჩნია ე.წ. ფესვი - ეს არის კვანძი რომელსაც არ ჰყავს მშობელი კვანძი, მისი პირობითი სიღრმეა 0 და ყოველი სხვა კვანძი ამ ხისა არის მისი შვილი და შვილთა შვილი სხვადასხვა სიღრმეში. ფოთოლი - არის კვანძი რომელსაც არ ჰყავს არც მარჯვენა და არც მარცხენა შვილი კვანძი.

მუშაობის ეტაპები - ალგორითმის მუშაობა პირობით სამ ნაწილად შეიძლება:

* მონაცემთა მიღება, სიაში ჩაწერა და დახარისხება. დახარისხება ხდება მონაწილეთა სიძლიერის მიხედვით. დახარისხებისთვის გამოიყენება .Net ის სტანდარტულ ბიბლიოთეკაში quicksort-ის იმპლემენტატორი რომელიც ახარისხებს List ტიპის კოლექციას და მუშაობისთვის ითხოვს რომ კოლექციაში მყოფი ობიექტები იყვნენ System.IComparable ინტერფეისის მაინმპლიმენტერები და გადატვირთავდნენ ნახსენები ინტერფეისის CompareTo(object o) ფუნქციას.
* დახარისხებული მონაწილეთა კოლექციის გამოყენებით მონაწილეთა შეხვედრების ამსახველი ორობითი ხის გენერირება. უკეთ ვისაუბრებთ მოგვიანებით.
* შეწყვილების შედეგად მიღებული ხის უკეთ აღსაქმელი ფორმით გამოტანა. ორობითი ხე არ არის ინფორმაციის მარტივად გასაგები ფორმა, საჭიროა მთლიანი ხის გარკვეული ხით ჩამოვლა და მონაცემთა გადატანა.  
  დაწერილია ფუნქცია რომელიც იღებს ხის ფესვს და აბრუნებს ორგანზომილებიან შეჯიბრის სიას: პირველი განზომილება აღნიშნავს შეჯიბრის სიღრმეს ( ფინალიდან შერჩევითებამდე ზრდადობით ), მეორე განზომილებაში მოქცეულია მონაცემები მოცემულ სიღრმეში მონაწილეთა წყვილების შესახებ ( დაწყებული ნულიდან ზრდადობით, მიმდევრობით ).  
  ტურნირი იწყება ყველზე ღრმა სიიდან მონაწილეთა წყვილების არჩევით და შემდეგი სიმაღლის ცარიელ უჯრებში გამარჯვებულის გადაყვანით.