

VISOKA ŠKOLA STRUKOVNIH STUDIJA ZA INFORMACIONE I KOMUNIKACIONE TEHNOLOGIJE



Napredno web programiranje

Dokumentacija projekta:

<https://sms-gateway-ict.000webhostapp.com/>

Mentor:

dr. Nenad Kojić

Studenti:

Goran Urukalo 1014/17

Nikola Simonović 1021/17

Beograd 2018.

Sadržaj

1. Opis funkcionalnosti.....	3
2. Skica struktura stranica	7
3. Dijagram baze podataka	9
4. Sitemap	10
5. Klasni dijagram	11
6. MVC organizacija	12
7. Slike stranica	18
8. Kod	33
1. PHP	33
2. JavaScript	117
3. CSS.....	138
9. Literatura	177

1. Opis funkcionalnosti

Slam Jam je web aplikacija koja se kreira kako bi se omogućilo održavanje Game Jam event-ova kao i učešćivanje u istim. Game Jam je bitan jer spaja nečiju ideju s ljudima koji mogu da je ostvare. Poenta je da kreatori napreduju u radu i razvoju svojih igara, kao i posmatrajuci druge projekte koje su drugi uesnici napravili.

Business spec

User dolazi na sajt kako bi video sta sve aplikacija ima. Posto je on neulogovan, nema ogroman broj funkcionalnosti ali ima ono zbog cega je i dosao prvobitno a to je razgledanje Game Jam event-ova i Game Submission-a, kao i pretrazivanje istih. Bice mu prikazana mogucnost da hostuje Game Jam event i da se prijavi na neki kao i da glasa za Game Submission i ostavlja komentare kao i mogucnost da proba iste, ali ce ga to sve voditi na **Login page** kako bi te funkcionalnosti otključao. Ako User nema nalog da se uloguje, postoji mogucnost registracije na kojoj mora da popuni neke licne stvari kao i da izabere da li zeli da bude Jam Maker, Jam Developer, oba ili nista od toga.

– Ako ne zeli neke od rola znaci da zeli samo da probava Game Submissione i da mozda komentarisuje i glasa.

– Ako izabere da zeli da bude Jam Maker, otvara mu se mogucnost za kreiranje novih Game Jam event-ova koji bi trebali da opisu taj event, kao i pregledanje i filtriranje svojih event-ova. Na kraju svakog Game Jam eventa sistem sam bira pobednika ali ukoliko postoji vise Game Submission-a na prvom mestu, Jam Maker mora da se odluci za prvo mesto. Jam Maker prilikom kreiranja Game Jam-a moze da izabere da li zeli da omoguci ostalima da glasaju za Game Submission-e. – Ako izabere Jam Developer ulogu, sada User moze da se prijavi na neki od Game Jam event-ova i da na istim i ucestvuje. Pre kraja event-a Jam Developer bi trebalo da submit-uje svoj projekat kako bi drugi ili samo Jam Maker ocenio taj projekat po nekim kriterijumima vec zadatim od strane Jam Maker-a. Ako bilo ko primeti nesto odstupa od pravila **sajta**, ima mogucnost da prijavi to **Admin-ima** kako bi oni mogli da pravilno reaguju.

Developer spec (usecase)

Kao Developer zelim da system ima uloge

- Admin (regulise sve moguće aktivnosti u okviru website-a)
- JamDeveloper (prijavljuje na jam-ove I da uploaduje file na taj prijavljeni jam event)
- JamMakere (stvara jam evente I odredjuju pobednika na kraju event-a)

Kao anonimni user ja zelim

- Da se registrujem na sajt
- Da se logujem na sajt
- Da napravim zahtev za reset password-a
- Da vidim listu game jam-ova
- Da filtriram game jam-ove
- Da vidim pojedinačan game jam
- Da vidim all game submissions
- Da filtriram all game submissions
- Da vidim pojedinačan game submission

Kao logovani user zelim

- Da se odlogujem
- Da probam game submission [skinuti]
- Da ostavljam komentare na game submissions
- Da mogu da glasam ako je dopusteno na game submissions
- Da glasam za Bedz
- Da promenim password
- Da promenim svoje informacije
- Da napravim report za game submission

Kao JamDeveloper zelim

- Da se prijavim na jam event
- Da se odjavim sa jam event-a
- Da napravi game submission za game jam event
- Da editujem game submission pre kraja jam event-a
- Da obrisem game submission pre kraja jam event-a

- Da vidim sve svoje game submission-e

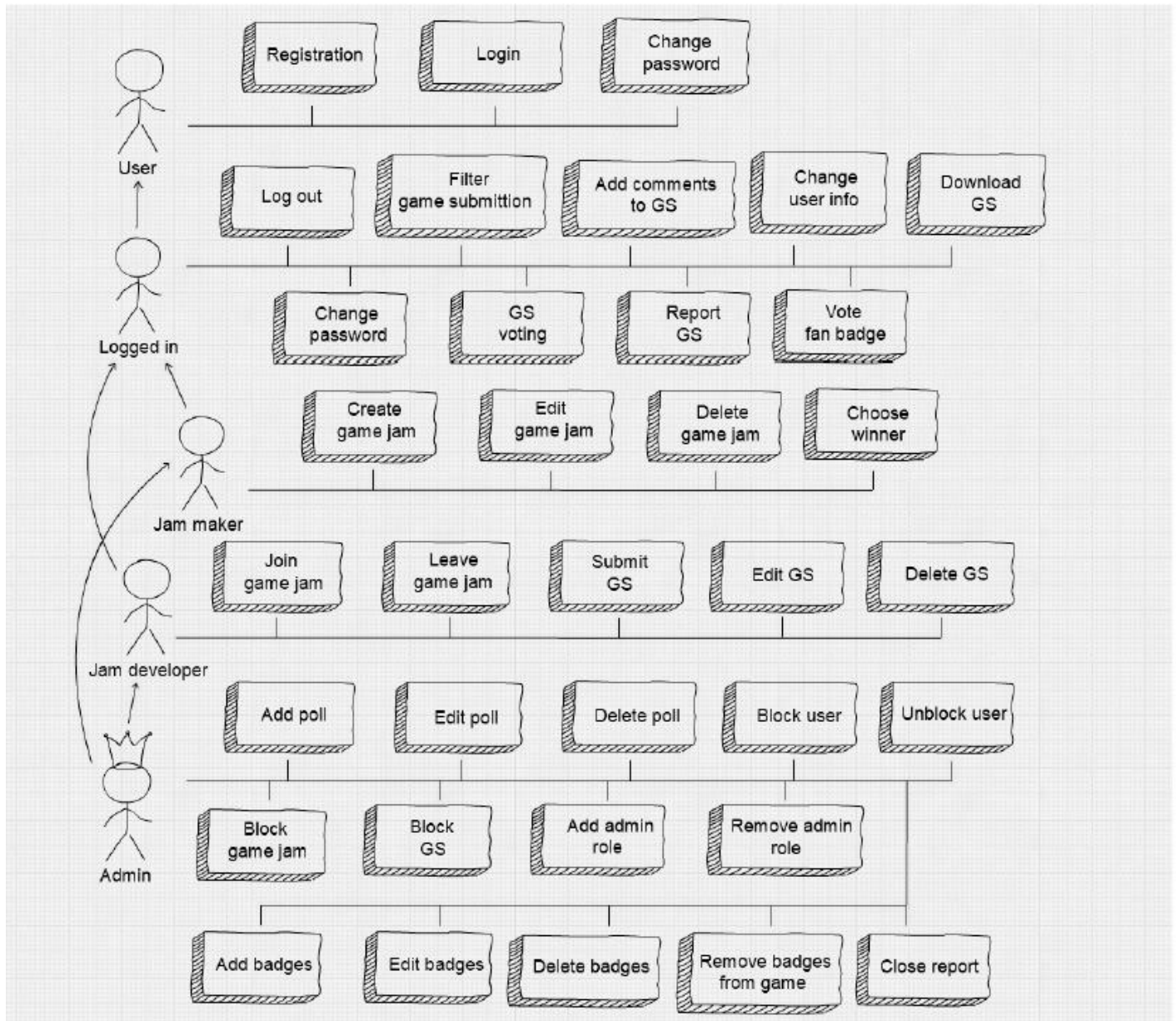
Kao JamMaker zelim

- Da napravim novi jam event
- Da editujem jam event pre pocetka
- Da obrisem jam event pre pocetka
- Da izaberem jam pobednika kada završi jam event

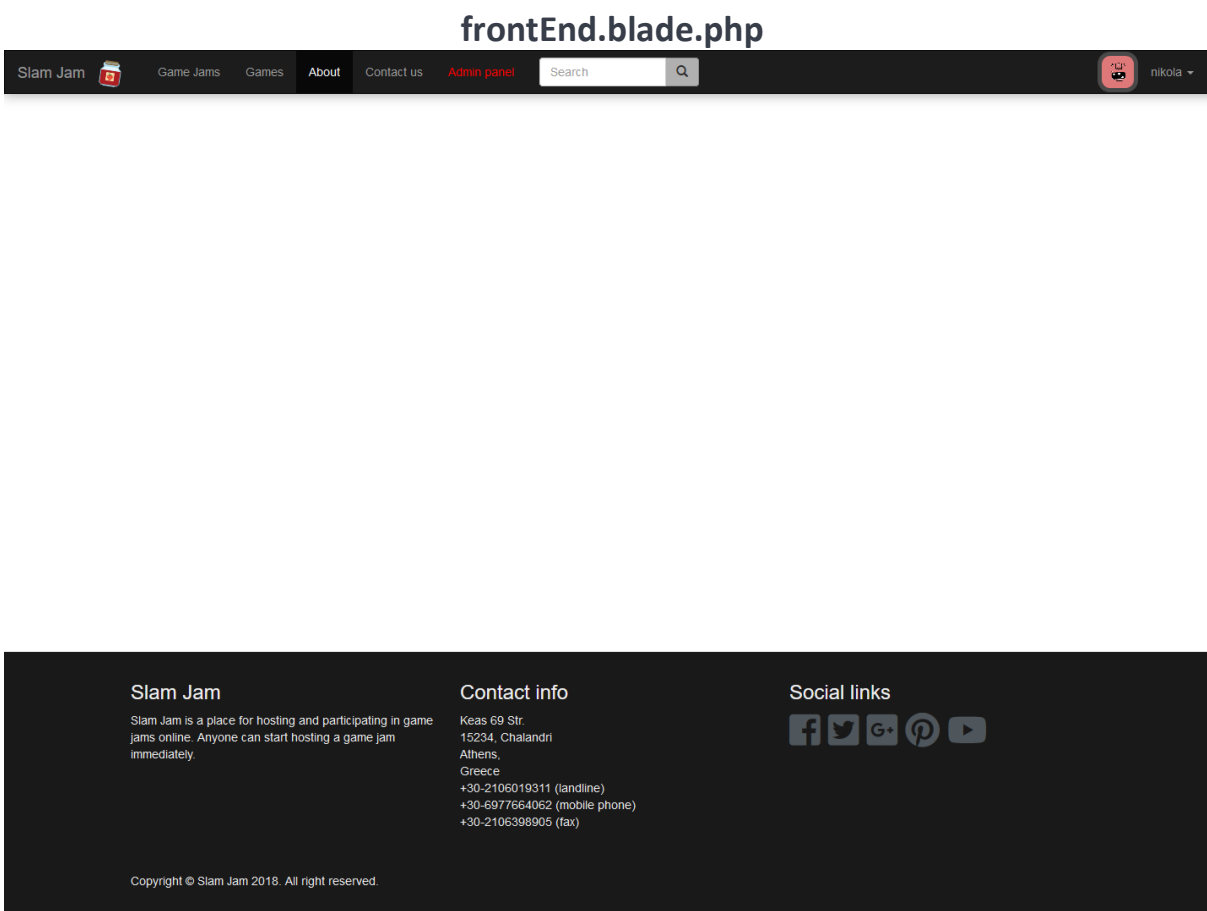
Kao Admin zelim

- Da dodam Admin ulogu korisniku
- Da uklonim Admin ulogu korisniku
- Da dodam Poll
- Da editujem Poll
- Da obrisem Poll
- Da block-ujem Usera
- Da unblock-ujem Usera
- Da block-ujem Game Jam
- Da unblock-ujem Game Jam
- Da block-ujem Game Submission
- Da unblock-ujem Game Submission
- Da dodam Bedz
- Da editujem Bedz
- Da obrisem Bedz
- Da obrisem Bedz sa Game Submission-a
- Da zatvorim Report kada ga obradim
- Da dodam Kategoriju
- Da editujem Kategoriju
- Da obrisem Kategoriju
- Da dodam Criteria
- Da editujem Criteria
- Da obrisem Criteria
- Da dodam ImageCategory
- Da editujem ImageCategory
- Da obrisem ImageCategory

Usecase diagram



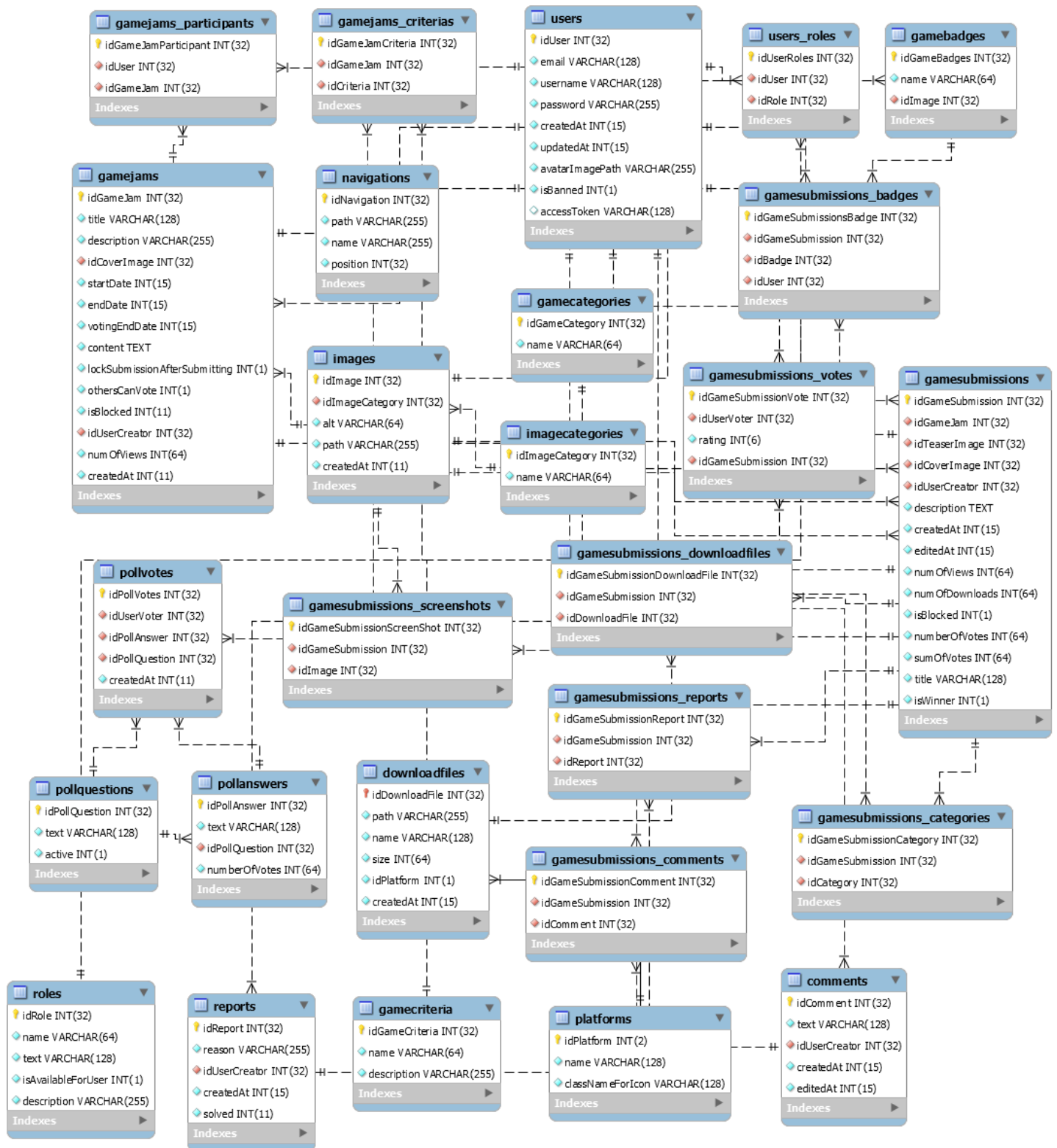
2. Skica struktura stranica



admin.blade.php



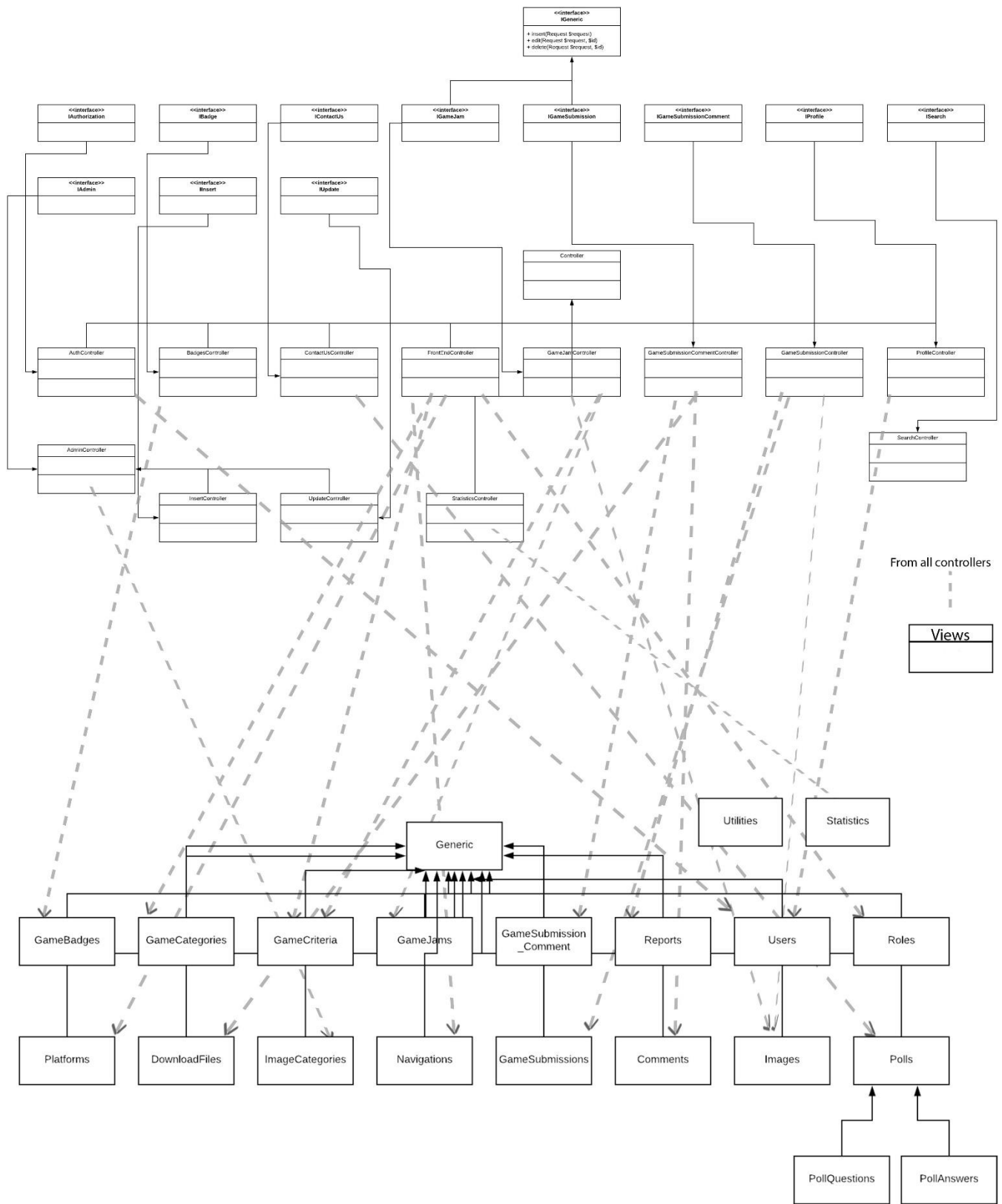
3. Dijagram baze podataka



4. Sitemap

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns=http://www.sitemaps.org/schemas/sitemap/0.9
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9
http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd">
<url>
<loc>https://slamjamphp.000webhostapp.com/</loc> <lastmod>2018-06-
25T19:53:28+00:00</lastmod><priority>1.00</priority>
</url>
<url>
<loc>https://slamjamphp.000webhostapp.com/games</loc>
<lastmod>2018-06-25T19:53:28+00:00</lastmod>
<priority>0.80</priority>
</url>
<url>
<loc>https://slamjamphp.000webhostapp.com/about</loc>
<lastmod>2018-06-25T19:53:28+00:00</lastmod>
<priority>0.80</priority>
</url>
<url>
<loc>https://slamjamphp.000webhostapp.com/contact-us</loc>
<lastmod>2018-06-25T19:53:28+00:00</lastmod>
<priority>0.80</priority>
</url>
<url>
<loc>https://slamjamphp.000webhostapp.com/doc.pdf</loc><lastmod>2018-06-
24T21:40:15+00:00</lastmod><priority>0.80</priority>
</url>
<url>
<loc>https://slamjamphp.000webhostapp.com/login</loc>
<lastmod>2018-06-25T19:53:28+00:00</lastmod>
<priority>0.80</priority>
</url>
<url>
<loc>https://slamjamphp.000webhostapp.com/register</loc><lastmod>2018-06-
25T19:53:28+00:00</lastmod><priority>0.80</priority>
</url>
</urlset>
```

5. Klasni diagram



6. MVC organizacija

Kontroleri

AuthController
login
logout
register

BadgesController
get
add
remove

GameJamController
getChartGameJams
getFilteredGameJams
insert
joinUserToGameJam
removeUserFromGameJam
update
delete

ContactUsController
pollVote
postContact

GameSubmissionCommentController
get
add
edit
remove

SearchController
search

GameSubmissionController
oneGameSubmission
insert

FrontEndController
gameJams
oneGameJam
createGameJam
editGameJam
games
createGameSubmission
editGameSubmission
register
login
about
contactUs
profile
editProfile
getUserProfileInfo

edit
delete
downloadFile
report
ProfileController
edit
getUsersGameJams
getUsersGames
getUsersWins

InsertController
users
gameCategories
gameCriteria
roles
imageCategories
platforms
navigations
pollquestions
pollanswers

AdminController
getTypeByTableName
index
block
insert
update
delete
getById
getAll

UpdateController
users
gameCategories

StatisticsController
getAllChart
getAllCount

gameCriteria
reports
roles
imageCategories
platforms
navigations
pollquestions
pollanswers
setActivePollQuestion

Models

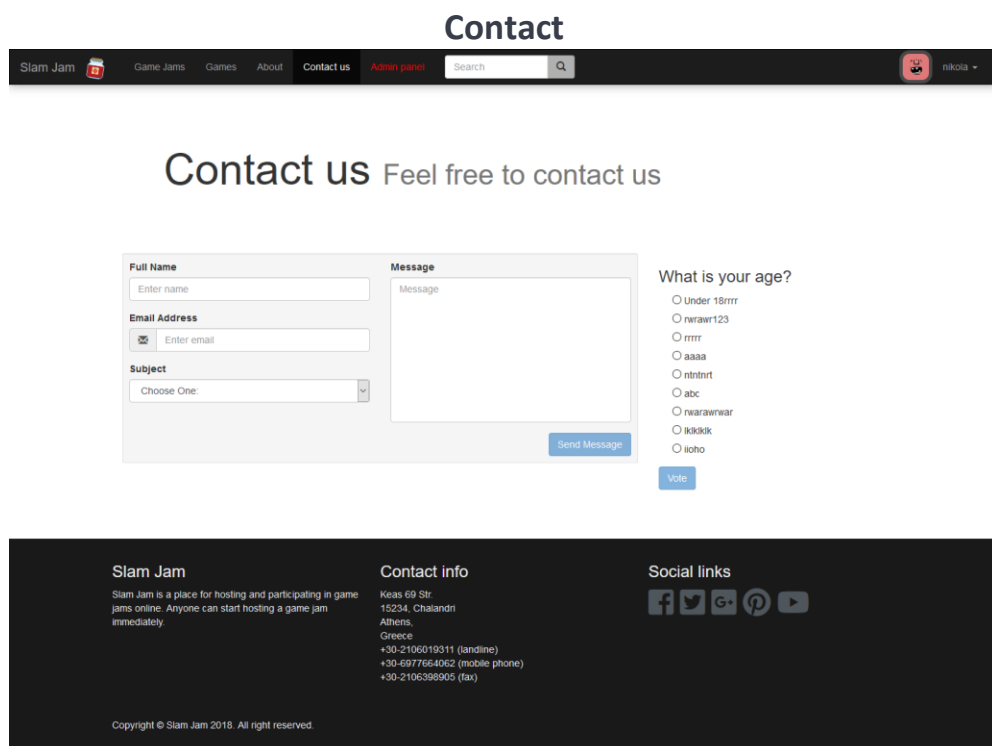
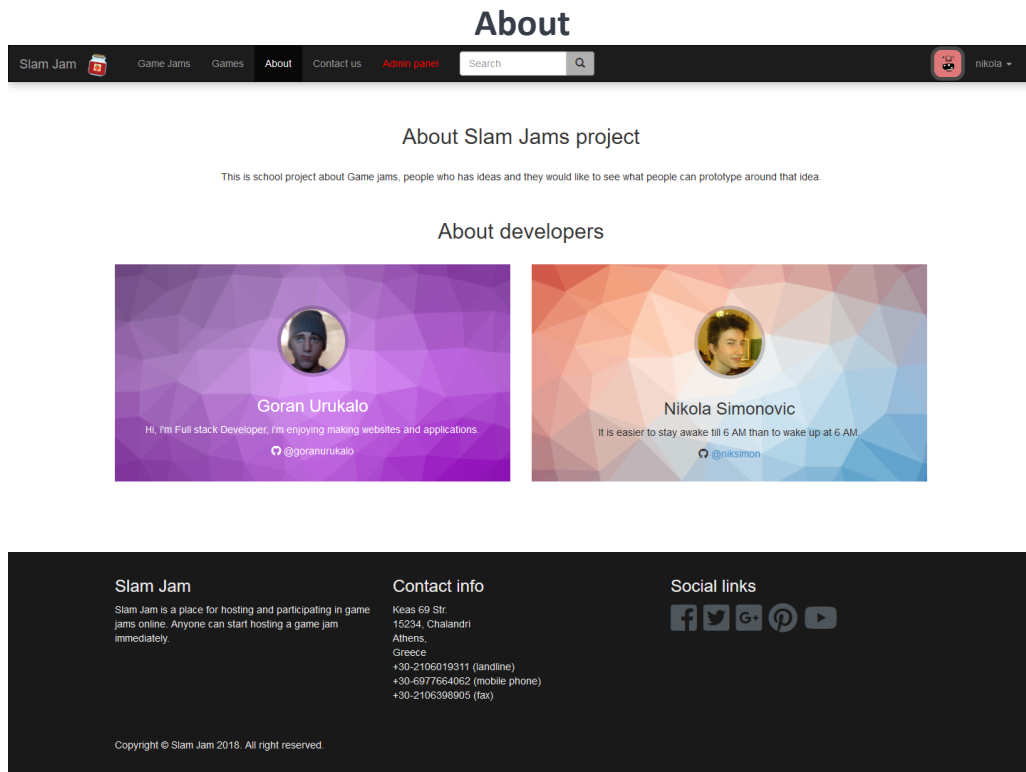
Comments
DownloadFiles
GameBadges
GameCategories
GameCriteria
GameJams
GameSubmission_Comments
GameSubmissions
Generic
ImageCategories
Images
Navigations
Platforms
PollAnswers
PollQuestions
Polls
Reports
Roles
Statistics
Users
Utilities

Views

admin/ajax/gamecategories.blade.php
admin/ajax/gamecriteria.blade.php
admin/ajax/gamejams.blade.php
admin/ajax/gamesubmissions.blade.php
admin/ajax/imagecategories.blade.php
admin/ajax/navigations.blade.php
admin/ajax/platforms.blade.php
admin/ajax/polls.blade.php
admin/ajax/reports.blade.php
admin/ajax/roles.blade.php
admin/ajax/users.blade.php
admin/gamecategories.blade.php
admin/gamecriteria.blade.php
admin/gamejams.blade.php
admin/gamesubmissions.blade.php
admin/imagecategories.blade.php
admin/navigations.blade.php
admin/platforms.blade.php
admin/polls.blade.php
admin/reports.blade.php
admin/roles.blade.php
admin/users.blade.php
ajax/loadGameJamsInProgress.blade.php
ajax/loadGameJamsUpcoming.blade.php

ajax/loadGames.blade.php
ajax/searchGameJams.blade.php
ajax/searchGameSubmissions.blade.php
auth/login.blade.php
auth/register.blade.php
gameJams/createGameJam.blade.php
gameJams/editGameJam.blade.php
gameJams/gameJams.blade.php
gameJams/oneGameJam.blade.php
gameSubmissions/createGameSubmissions.blade.php
gameSubmissions/editGameSubmission.blade.php
gameSubmissions/games.blade.php
gameSubmissions/oneGameSubmissions.blade.php
layouts/admin.blade.php
layouts/frontEnd.blade.php
layouts/mail.blade.php
mailPages/contactUs.blade.php
mailPages/registrationWelcome.blade.php
mailPages/template.blade.php
other/about.blade.php
other/contactUs.blade.php
other/search.blade.php
user/userEdit.blade.php
user/userProfile.blade.php
user/usersGameJams.blade.php
user/usersGames.blade.php
user/usersWins.blade.php

7. Slike stranica



Edit profile

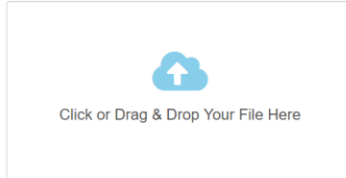
Edit profile







Upload avatar image:



- ☒ **Jam Maker**
User with this role can host game jams.
- ☒ **Jam Developer**
User with this role can create Game submission.

[Update](#)

Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info

Keas 69 Str.
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977654062 (mobile phone)
+30-2106399906 (fax)

Social links



Game Jams

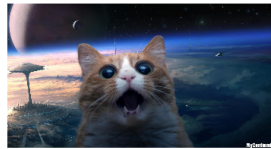
Game Jams on Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately. Here you can find some of the game jams that are going on.

[Host own Game Jam](#)

June 2018							
	Sat 23	Sun 24	Mon 25	Tue 26	Wed 27	Thu 28	Fri 29
	Fun game jam	Fun game jam					
	Optical game jam						
							Test game
	Sat 23	Sun 24	Mon 25	Tue 26	Wed 27	Thu 28	Fri 29
June 2018							

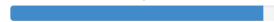
Game Jams in progress



Optical game jam

Hosted by: [nikola](#)

Submissions closes in: 5 days



1 joined 0 submissions

1

Upcoming Game Jams



Test game jam

Hosted by: [nikola](#)

Starts in 3 days

0 joined

1

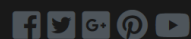
Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info

Keas 69 Str.
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977664062 (mobile phone)
+30-2106398905 (fax)

Social links



Games

Slam Jam 

Game Jams

Games

About

Contact us

Search



perica ▾

Games (10 results)

Newest ▾



GTA V

Submitted by: thelastman

Racing



Goat simulator

Submitted by: thelastman

Racing FPS



Arma 3

Submitted by: thelastman

RTS



Counter-Strike

Submitted by: thelastman

FPS RPG VR



Warframe

Submitted by: thelastman

RPG



Euro Truck Simulator 2

Submitted by: thelastman

FPS



Skyrim

Submitted by: thelastman

RPG



Oblivion

Submitted by: machineguy

VR



Portal 2

Submitted by: machineguy

Racing

1 2

Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info

Keas 69 Str.
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977664062 (mobile phone)
+30-2106398905 (fax)

Social links




Copyright © Slam Jam 2018. All right reserved.

Login

Login to start your session





[Register a new membership](#) [Login](#)

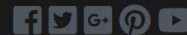
Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info

Keas 69 Str
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977664062 (mobile phone)
+30-2106398905 (fax)


Social links





Copyright © Slam Jam 2018. All right reserved.


Register

Register a new membership









☐ **Jam Maker**
User with this role can host game jams.

☐ **Jam Developer**
User with this role can create Game submission.

[I already have an account](#) [Register](#)

Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info

Keas 69 Str
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977664062 (mobile phone)
+30-2106398905 (fax)

Social links



Copyright © Slam Jam 2018. All right reserved.

My game jams

My Game Jams



Fun game jam

Hosted by: nikola

Starts in -90 hr

0 joined



Optical game jam

Hosted by: nikola

Starts in -2756 hr

1 joined



Test game jam

Hosted by: nikola

Starts in 3 days

0 joined

1

Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info

Keas 69 Str.
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977664062 (mobile phone)
+30-2106396905 (fax)

Social links



Copyright © Slam Jam 2018. All right reserved.

My games

My Games



Oblivion

Submitted by: machineguy

VR



Portal 2

Submitted by: machineguy

Rating



PUBG

Submitted by: machineguy

VR

1

Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info

Keas 69 Str.
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977664062 (mobile phone)
+30-2106396905 (fax)

Social links



Copyright © Slam Jam 2018. All right reserved.

Game

Slam Jam



[Game Jams](#)

[Games](#)

[About](#)

[Contact us](#)

[Admin panel](#)

Search



nikola



Arma 3

Submitted by [thelastman](#) at 1970-01-02 10:30:14

55
downloads

21
views

There is currently no screenshots for this game.

Downloads:

Badges:

Select game badge...

Add Badge

Description:

game description

There is currently no badges for this game.

Comments:

Post a new Comment

Comment text...

Add Comment



nikola me just now

hello

Report

Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info

Keas 69 Str.
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977664062 (mobile phone)
+30-2106398905 (fax)

Social links



Copyright © Slam Jam 2018. All right reserved.

Game Jam

Slam Jam

Game Jams

Games

About

Contact us

Search



perica



Optical game jam

Hosted by nikola

1 joined
0 submissions
3 views

Submissions due in



Leave game jam

Add game submission

Description:

Description

Criteria:

Design

Sound

Overview

Participants

Game submissions



Content rwanwar

Slam Jam

Slam Jam is a place for hosting and participating in game jams online. Anyone can start hosting a game jam immediately.

Contact info


Keas 69 Str.
15234, Chalandri
Athens,
Greece
+30-2106019311 (landline)
+30-6977664062 (mobile phone)
+30-2106398905 (fax)

Social links



Copyright © Slam Jam 2018. All right reserved.

Profile



Username: nikola

Joined: 05/26/2018

User type:

- Administrator
- Jam Maker
- Jam Developer

Email: [warwar](#)


Banned: No

[Edit my profile](#)

Search


Results for "game"

Game Jams




Fun game jam

Hi! RvVawar wararawara awerawar
VAHwar Anwar awerwar.



Optical game jam

Description




Test game jam

Test game jam


1

Game Jams




Goat simulator

game description




Arma 3

game description




Counter-Strike

game description




Warframe

game description



Euro Truck Simulator 2

game description



Skyrim

game description

1 2

Admin/Game Categories

SLAM JAM
ADMIN PANEL

DASHBOARDS

Overview

Users

Game jams

Game submissions

Game categories

Game criteria

Image categories

Navigations

Polls

Roles

Reports

Platforms

×

Game Categories

+

🗑

<div></div>	ID	Name	Edit	Delete
<div></div>	1	FPS	<div></div>	<div></div>
<div></div>	2	RPG	<div></div>	<div></div>
<div></div>	14	Racing	<div></div>	<div></div>
<div></div>	16	RTS	<div></div>	<div></div>
<div></div>	29	VR	<div></div>	<div></div>

↩

Admin/Game Criteria

SLAM JAM
ADMIN PANEL

DASHBOARDS

Overview

Users

Game jams

Game submissions

Game categories

Game criteria

Image categories

Navigations

Polls

Roles

Reports

Platforms

×

Game Criteria

+

🗑

<div></div>	ID	Name	Description	Edit	Delete
<div></div>	1	Gameplay	How the game plays	<div></div>	<div></div>
<div></div>	2	Design	Design of the game	<div></div>	<div></div>
<div></div>	3	Sound	The audio of the game	<div></div>	<div></div>
<div></div>	4	Performance	The game performance	<div></div>	<div></div>

↩

Admin/Game Jams

Game Jams

ID	Title	Description	Cover	Dates	Content	Lock Submissions	Others Can Vote	Created By	Views	Block
1	Fun game jam	Hi! RWawrw...		22/06/2018 02:11 AM 24/06/2018 02:11 AM 26/06/2018 02:11 AM	Hi!	No	Yes	nikola	11	<input type="checkbox"/>
2	wrarwarawr	warwarwarw...		15/01/1970 01:16 AM 05/09/1970 07:53 PM 15/01/1970 09:07 AM	trwarawr	No	No	machineguy	5424	<input checked="" type="checkbox"/>
3	wrarwarawr	warwarwarw...		15/01/1970 01:16 AM 05/09/1970 07:53 PM 15/01/1970 09:07 AM	trwarawr	No	No	machineguy	5424	<input checked="" type="checkbox"/>
4	wrarwarawr	warwarwarw...		15/01/1970 01:16 AM 05/09/1970 07:53 PM 15/01/1970 09:07 AM	trwarawr	No	No	machineguy	5424	<input type="checkbox"/>
5	wrarwarawr	warwarwarw...		15/01/1970 01:16 AM 05/09/1970 07:53 PM 15/01/1970 09:07 AM	trwarawr	No	No	machineguy	5424	<input checked="" type="checkbox"/>
6	Optical game jam	Description		03/03/2018 12:36 AM 30/06/2018 10:10 PM 04/07/2018 10:10 PM	Content rw...	No	Yes	nikola	5	<input type="checkbox"/>
7	Test game jam	Test game jam		28/06/2018 10:12 PM 30/06/2018 10:12 PM 03/07/2018 10:12 PM	Test game jam	Yes	No	nikola	3	<input type="checkbox"/>

Admin/Game Submissions

SLAM JAM
ADMIN PANEL



Game Submissions

ID	Title	Created by	Game Jam	Cover	Description	Dates	Views	Downloads	Rating	Is Winner	Block
1	Goat simulator	thelastman	Fun game jam		game description	17/02/1970 05:36 PM 17/02/2015 11:03 PM	20	55	9.5	No	<input checked="" type="checkbox"/>
2	Arma 3	thelastman	Fun game jam		game description	17/02/1970 05:36 PM 17/02/2015 11:03 PM	21	55	9.5	No	<input type="checkbox"/>
3	Counter-Strike	thelastman	Fun game jam		game description	17/02/1970 05:36 PM 17/02/2015 11:03 PM	14	55	9.5	No	<input type="checkbox"/>
4	Warframe	thelastman	Fun game jam		game description	17/02/1970 05:36 PM 17/02/2015 11:03 PM	14	55	9.5	No	<input type="checkbox"/>
5	ETS 2	thelastman	Fun game jam		game description	17/02/1970 05:36 PM 17/02/2015 11:03 PM	14	55	9.5	No	<input checked="" type="checkbox"/>
6	Skyrim	thelastman	Fun game jam		game description	17/02/1970 05:36 PM 17/02/2015 11:03 PM	14	55	9.5	No	<input checked="" type="checkbox"/>
7	GTA V	thelastman	Fun game jam		game description	14/08/1973 10:10 PM 07/12/1973 02:55 PM	1	0	0	No	<input type="checkbox"/>
8	Oblivion	machineguy	Test game jam		descr	01/01/1970 03:27 AM 03/01/1970 11:28 AM	0	0	0	No	<input type="checkbox"/>
9	Portal 2	machineguy	Test game jam		descr	01/01/1970 03:27 AM 03/01/1970 11:28 AM	0	0	0	No	<input type="checkbox"/>
11	PUBG	machineguy	Test game jam		descr	01/01/1970 03:27 AM 03/01/1970 11:28 AM	0	0	0	No	<input type="checkbox"/>

Admin/Image Categories

SLAM JAM
ADMIN PANEL

DASHBOARDS

Overview

Users

Game jams

Game submissions

Game categories

Game criteria

Image categories

Navigations

Polls

Roles

Reports

Platforms

×

Image Categories

+

🗑️

	ID	Name	Edit	Delete
🗑️	1	Cover	✎	🗑️
🗑️	2	Avatar	✎	🗑️
🗑️	3	Teaser	✎	🗑️
🗑️	4	Badge	✎	🗑️

[↩️](#)

Admin/Navigations

SLAM JAM
ADMIN PANEL

DASHBOARDS

Overview

Users

Game jams

Game submissions

Game categories

Game criteria

Image categories

Navigations

Polls

Roles

Reports

Platforms

×

Navigations

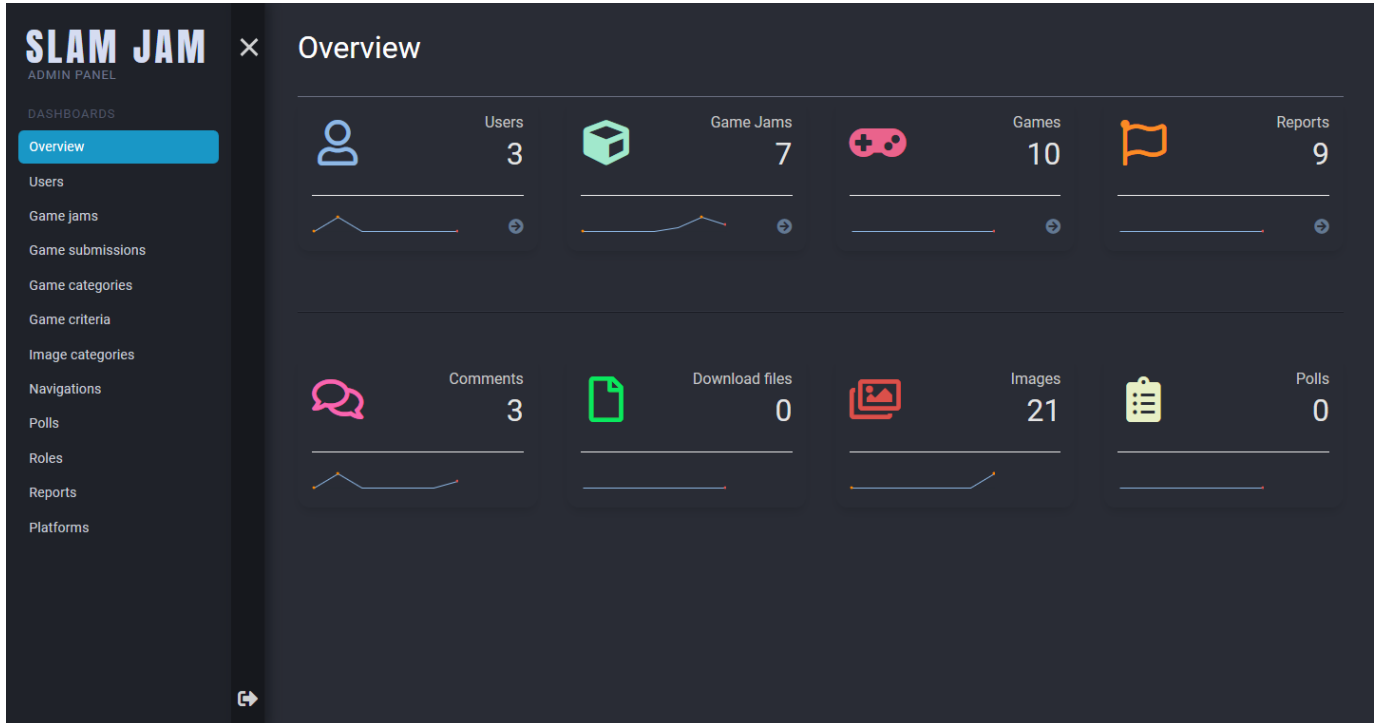
+

🗑️

	ID	Path	Name	Position	Edit	Delete
🗑️	1	/	Game Jams	1	✎	🗑️
🗑️	2	/games	Games	2	✎	🗑️
🗑️	3	/about	About	3	✎	🗑️
🗑️	4	/contact-us	Contact us	4	✎	🗑️

[↩️](#)

Admin/Overview



Admin/Platforms

SLAM JAM
ADMIN PANEL

DASHBOARDS

- Overview
- Users
- Game jams
- Game submissions
- Game categories
- Game criteria
- Image categories
- Navigations
- Polls
- Roles
- Reports
- Platforms

Platforms

ID	Name	Font Awesome Class	Edit	Delete
1	windows	fab fa-windows		
2	linux	fab fa-linux		
3	apple	fab fa-apple		
4	android	fab fa-android		
5	playstation	fab fa-playstation		
6	xbox	fab fa-xbox		
7	nintendo switch	fab fa-nintendo-switch		

Admin/Polls

SLAM JAM

ADMIN PANEL

DASHBOARDS

Overview

Users

Game jams

Game submissions

Game categories

Game criteria

Image categories

Navigations

Polls

Roles

Reports

Platforms

×

Polls

+

	ID	Text	Active	Edit	Delete
▼	1	What is your age?	<input type="checkbox"/>		
▼	2	Question warwar warwa wara wra	<input type="checkbox"/>		
▼	12	rwarwaree	<input type="checkbox"/>		
▼	19	Rt32ttwatwatwt	<input checked="" type="checkbox"/>		
▼	22	rwrrr	<input type="checkbox"/>		
▼	23	bbb123	<input type="checkbox"/>		
▼	24	abc	<input type="checkbox"/>		
▼	25	abc	<input type="checkbox"/>		
▼	26	gegeg	<input type="checkbox"/>		
▼	27	gegeg	<input type="checkbox"/>		
▼	28	awraawrrra	<input type="checkbox"/>		

↩

Admin/Reports

SLAM JAM

ADMIN PANEL

DASHBOARDS

Overview

Users

Game jams

Game submissions

Game categories

Game criteria

Image categories

Navigations

Polls

Roles

Reports

Platforms

×

Reports

ID	User	Reason	Created at	Close?
1	thelastman	Blabla	20/06/2018 02:36 AM	<input type="checkbox"/>
3	nikola	rawrwarwatwa	26/05/2018 02:06 PM	<input type="checkbox"/>
8	nikola	rawrwarwatwa	26/05/2018 02:06 PM	<input type="checkbox"/>
9	nikola	rawrwarwatwa	26/05/2018 02:06 PM	<input type="checkbox"/>

↩

Admin/Roles

SLAM JAM

ADMIN PANEL

DASHBOARDS

Overview

Users

Game jams

Game submissions

Game categories

Game criteria

Image categories

Navigations

Polls

Roles

Reports

Platforms

×

Roles

+

🗑️

ID	Name	Text	Is Available For User	Description	Edit	Delete
1	admin	Administrator	No	User with this role can manage everything on website.		
2	jamMaker	Jam Maker	Yes	User with this role can host game jams.		
3	jamDeveloper	Jam Developer	Yes	User with this role can create Game submission.		

Admin/Users

SLAM JAM

ADMIN PANEL

DASHBOARDS

Overview

Users

Game jams

Game submissions

Game categories

Game criteria

Image categories

Navigations

Polls

Roles

Reports

Platforms

×

Users

+

🗑️

ID	Username	E-mail	Avatar	Created at	Updated at	Banned	Roles	Edit	Delete
2	nikola	wrrrrr		26/05/2018 02:06 PM	25/06/2018 04:52 PM	No			
118	machineguy	perica@gmail.com		20/06/2018 02:36 AM	22/06/2018 02:32 AM	No			
119	thelastman	pera2@gmail.com		20/06/2018 02:36 AM	20/06/2018 02:36 AM	No			

8. Kod

1. PHP

```
#### app\Http\Controllers\AuthController.php

...

<?php

namespace App\Http\Controllers;

use \App\Http\Interfaces\IAuthorization;
use App\Http\Models\Roles;
use App\Http\Models\Users;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class AuthController extends Controller implements IAuthorization
{
    public function login(Request $request)
    {
        $validation = Validator::make($request->all(), [
            'tbUsernameEmail' => 'required',
            'tbPassword' => 'required|min:6'
        ]);

        $validation->setAttributeNames([
            'tbUsernameEmail' => 'username',
            'tbPassword' => 'password'
        ]);

        if ($validation->fails()) {
            return back()->withInput()->withErrors($validation);
        }

        $user = new Users();

        $dbUser = $user->getByUsernameOrEmailAndPassword($request-
>get('tbUsernameEmail'), $request->get('tbPassword'));

        if (!empty($dbUser)) {
            if($dbUser->isBanned == 1){
                return back()->withErrors(['message' => "This account is removed, go
to \"Contact us\" for more information."]);
            }

            $userRoles = $user->getAllRoles($dbUser->idUser);

            $request->session()->push("user", $dbUser);
            $request->session()->push("roles", $userRoles);

            $isAdmin = Roles::arrayOfRolesHasRoleByName($userRoles, 'admin');

            $redirectPath = "/";
            if ($isAdmin) {
                $redirectPath = "/admin";
            }
        }
    }
}
```

```

    }

    //add token to user
    $cookieToken = time() . str_random(60);
    $user->updateAccessToken($dbUser->idUser, $cookieToken);

    //default redirect
    return redirect($redirectPath)->withCookie(cookie()->forever('authToken',
$cookieToken));
}

    return back()->withInput()->withErrors(['message' => "Username/Email or
password is incorrect"]);
}

public function logout(Request $request)
{
    // remove access token from users table
    if (session()->has('user')) {
        $idUser = session()->get('user')[0]->idUser;

        $user = new Users();
        $user->removeAccessToken($idUser);
    }

    // remove session
    $request->session()->flush();

    return redirect('/');
}

public function register(Request $request)
{
    $validacija = Validator::make($request->all(), [
        'tbEmail' => 'required|unique:users,email',
        'tbUsername' => 'required|unique:users,username',
        'tbPassword' => 'required|confirmed|min:6',
    ]);
    $validacija->setAttributeNames([
        'tbEmail' => 'email',
        'tbUsername' => 'username',
        'tbPassword' => 'password',
    ]);

    if ($validacija->fails()) {
        // redirecija na pocetnu i ispis gresaka
        return back()->withInput()->withErrors($validacija);
    }

    $userRoles = $request->get('userRoles');
    $user = new Users();

    $userId = $user->insert(
        strtolower($request->get('tbEmail')),
        strtolower($request->get('tbUsername')),
        $request->get('tbPassword')
    );
};

```

```

        //something went wrong and user isn't inserted
        if (empty($userId)) {
            return back()->withInput()->with('messages', 'Registration failed!');
        }

        //add roles
        if (!empty($userRoles)) {
            foreach ($userRoles as $role) {
                $user->addRole($role, $userId);
            }
        }

        return redirect('/login')->with('messages', 'You are successfully
registered!');
    }
}

...

```

app\Http\Controllers\BadgesController.php

...

<?php

```

namespace App\Http\Controllers;

use App\Http\Interfaces\IBadge;
use App\Http\Models\GameBadges;
use App\Http\Models\GameSubmissions;
use Illuminate\Http\Request;

class BadgesController extends Controller implements IBadge
{
    public function get(Request $request, $gameId)
    {
        if (!preg_match("/^\d+$/", $gameId)) {
            return response()->json(["error"=>["message"=>"Invalid game id!"]], 400);
        }

        try {
            $badges = new GameBadges();
            $result = $badges->getByGameSubmissionId($gameId);

            return response()->json($result, 200);
        } catch (\Exception $e) {
            //todo, log this in some file
            return response()->json(null, 500);
        }
    }

    public function add(Request $request, $gameId, $badgeId)

```

```

{
    if (!preg_match("/^\d+$/", $gameId)) {
        return response()->json(["error"=>["message"=>"Invalid game id!"]], 400);
    } else if (!preg_match("/^\d+$/", $badgeId)) {
        return response()->json(["error"=>["message"=>"Invalid badge id!"]],
400);
    }
    $userId = $request->attributes->get('userInfo')->idUser;

    //
    $bages = new GameBadges();
    $games = new GameSubmissions();
    $gameSubmission = $games->getById($gameId);

    if(empty($gameSubmission))
    {
        //game doesn't exist
        return response()->json(["error"=>["message"=>"Game doesn't exist!"]],
400);
    }

    if($gameSubmission->idUserCreator == $userId)
    {
        //user is owner
        return response()->json(["error"=>["message"=>"User creator cannot add
badges!"]], 400);
    }

    $gameBages = $bages->getAllByGameId($gameId);

    $isAdded = false;
    $userHasAddedIt = false;

    foreach($gameBages as $gb)
    {
        if($gb->idBadge == $badgeId)
        {
            $isAdded = true;
            break;
        }

        if($gb->idUser == $userId)
        {
            $userHasAddedIt = true;
            break;
        }
    }

    if($isAdded)
    {
        //user is owner
        return response()->json(["error"=>["message"=>"Badge already added!"]],
400);
    }

    if($userHasAddedIt)
    {
        //user is owner

```

```

        return response()->json(["error"=>["message"=>"User has already added
badge!"]], 400);
    }

    // insert and return data
    $newBadge = $games->addBadge($gameId, $badgeId, $userId);

    return response()->json($newBadge, 201);
}

public function remove(Request $request, $gameId, $badgeId)
{
    if (!preg_match("/^\d+$/", $gameId)) {
        return response()->json(["error"=>["message"=>"Invalid game id!"]], 400);
    } else if (!preg_match("/^\d+$/", $badgeId)) {
        return response()->json(["error"=>["message"=>"Invalid badge id!"]],
400);
    }

    $userId = $request->attributes->get('userInfo')->idUser;

    try {
        $games = new GameSubmissions();
        $gameBadge = $games->getOneGameSubmissionBadgeById($badgeId);

        // da li postoji badge
        if(empty($gameBadge))
        {
            //game doesn't exist
            return response()->json(["error"=>["message"=>"Game badge doesn't
exist!"]], 400);
        }

        // da li badge pripada zapravo tom game submission-u
        if($gameBadge->idGameSubmission != $gameId)
        {
            //badge doesn't exist for that game
            return response()->json(["error"=>["message"=>"Game badge doesn't
exist!"]], 400);
        }

        // da li ga prava osoba brise [aka, user creator = user id
        if($gameBadge->idUser != $userId)
        {
            // its not the badges owner
            return response()->json(["error"=>["message"=>"You cannot remove
others badge!"]], 400);
        }

        $games->removeBadge($badgeId);

        return response()->json(null, 204);
    } catch (\Exception $e) {
        //todo, log this in some file
        return response()->json(null, 500);
    }
}
}

```

...

app\Http\Controllers\ContactUsController.php

...

<?php

namespace App\Http\Controllers;

use App\Http\Interfaces\IContactUs;
use App\Http\Models\Polls;
use App\Mail\ContactUsMail;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;
use Illuminate\Support\Facades\Validator;

class ContactUsController extends Controller implements IContactUs
{

public function index(Request \$request)
{

}

public function pollVote(Request \$request)
{
 \$validation = Validator::make(\$request->all(), [
 'idPollQuestion' => 'required|numeric',
 'pollAnswer' => 'required|numeric'
]);

 if(\$validation->fails()){
 return back()->withInput()->withErrors(\$validation);
 }

 \$userId = \$request->session()->get('user')[0]->idUser;
 \$questionId = \$request->get('idPollQuestion');
 \$answerId = \$request->get('pollAnswer');

 \$poll = new Polls();
 \$firstTimeVoting = \$poll->pollVote(\$userId, \$questionId, \$answerId);

 if(!\$firstTimeVoting)
 {
 return back()->with('message', 'Sorry but you have already voted.'); }

 return back()->with('message', 'Thank you for voting!');}

public function postContact(Request \$request)

```

{
    $validation = Validator::make($request->all(), [
        'tbFullName' => 'required',
        'tbEmail' => 'required',
        'soSubject' => 'required',
        'tbMessage' => 'required',
    ]);

    $validation->setAttributeNames([
        'tbFullName' => 'Full name',
        'tbEmail' => 'Email',
        'soSubject' => 'Subject',
        'tbMessage' => 'Message',
    ]);

    if($validation->fails()){
        return back()->withInput()->withErrors($validation);
    }

    $sendData = [
        'fullName' => $request->get('tbFullName'),
        'email' => $request->get('tbEmail'),
        'subject' => $request->get('soSubject'),
        'userMessage' => $request->get('tbMessage'),
    ];

    Mail::to('admins@slam-jam.com')->send(new ContactUsMail($sendData));
    return back()->with('message', 'Thank you for contacting us!');
}
}

...

```

```
#### app\Http\Controllers\Controller.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Routing\Controller as BaseController;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
```

```
class Controller extends BaseController
```

```
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
}
```

```
...
```

```
#### app\Http\Controllers\FrontEndController.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Controllers;
use App\Http\Models\GameCategories;
use App\Http\Models\Navigations;
use App\Http\Models\Platform;
use App\Http\Models\Polls;
use App\Http\Models\Roles;
use Illuminate\Http\Request;
use App\Http\Models\Users;
use App\Http\Models\GameCriteria;
use App\Http\Models\GameSubmissions;
use App\Http\Models\GameJams;
use Illuminate\Support\Facades\Redirect;

class FrontEndController extends Controller
{
    private $viewData = [];
    public function __construct()
    {
        //$navs = new Navigations();
        //$this->viewData['navigation'] = $navs->getAllSortedByPosition();
    }
    //
    // game jams
    public function gameJams(Request $request) {
        //todo
        //move to own controller

        $page = empty($request->get("page")) ? 1 : $request->get("page");

        $gameJams = new GameJams();

        $gameJamsInProgress = $gameJams->getFilteredGameJams("progress", ($page - 1)
* 6, 6);
        $this->viewData["inProgressGameJams"] = $gameJamsInProgress["result"];
        $this->viewData["gamesJamsInProgressCount"] = $gameJamsInProgress["count"];
        $this->viewData["currentPageGameJamsInProgress"] = $page;

        $gameJamsUpcoming = $gameJams->getFilteredGameJams("upcoming", ($page - 1) *
6, 6);
        $this->viewData["upcomingGameJams"] = $gameJamsUpcoming["result"];
        $this->viewData["gamesJamsUpcomingCount"] = $gameJamsUpcoming["count"];
        $this->viewData["currentPageGameJamsUpcoming"] = $page;

        if ($request->ajax()) {
            if($request->get("gameJamsType") === "inProgress") {
                return view('ajax.loadGameJamsInProgress', $this->viewData)-
>render();
            }
        }
    }
}
```



```

        }
        else{
            return view('ajax.loadGameJamsUpcoming', $this->viewData)->render();
        }
    }

    return view('gameJams.gameJams', $this->viewData);
}

public function oneGameJam($id){
    if (!preg_match("/^\d+$/", $id)) {
        return back()->with('message', 'Invalid game jam id.');
```

}

 \$gameJams = new GameJams();

 \$gameJam = \$gameJams->getById(\$id);

 if(empty(\$gameJam)){
 return Redirect::back()->withInput()->with("message", "Game jam doesn't exist!");
 }

 \$this->viewData["userCanEditAndDeleteGameJam"] = \$this->viewData["userJoinedGameJam"] = false;

 if(session()->has('user')) {
 \$idUser = session()->get('user')[0]->idUser;
 \$this->viewData["userJoinedGameJam"] = \$gameJams->userAlreadyJoined(\$idUser, \$id);

 if(\$gameJam->startDate > time()){
 \$this->viewData["userCanEditAndDeleteGameJam"] = \$gameJams->userOwnsGameJam(\$idUser, \$id);
 }
 }

 \$gameJams->increaseViews(\$id);

 \$this->viewData["gameJam"] = \$gameJams->getOne(\$id);

 return view('gameJams.oneGameJam', \$this->viewData);
 }

 public function createGameJam(){
 \$gameCriteria = new GameCriteria();
 \$this->viewData['criteria'] = \$gameCriteria->getAll();
 return view('gameJams.createGameJam', \$this->viewData);
 }

 public function editGameJam(\$id){
 if (!preg_match("/^\d+\$/", \$id)) {
 return back()->with('message', 'Invalid game jam id.');

}

 \$gameJams = new GameJams();
 \$gameCriteria = new GameCriteria();

 \$gameJam = \$gameJams->getOne(\$id);

 \$this->viewData["userCanEditGameJam"] = false;

```

        if($gameJam->startDate < time()){
            return Redirect::back()->withInput()->with("message", "You can no longer
edit this game jam!");
        }
        else {
            if(session()->has('user')){
                $idUser = session()->get('user')[0]->idUser;
                if(!$gameJams->userOwnsGameJam($idUser, $id)){
                    return Redirect::back()->withInput()->with("message", "You can't
edit this game jam!");
                }
            }
        }

        foreach($gameJam->criteria as $c) {
            $this->viewData["gameHasCriteria"][] = $c->idGameCriteria;
        }

        $this->viewData["gameCriteria"] = $gameCriteria->getAll();
        $this->viewData["gameJam"] = $gameJams->getOne($id);

        return view('gameJams.editGameJam', $this->viewData);
    }
    //
    // games submissions views
    public function games(Request $request) {
        //todo
        //move to own controller
    } //this needs to move in GameSubmission controller

    public function createGameSubmission($idGameJam){
        //todo
        //move to own controller
        if (!preg_match("/^\d+$/", $idGameJam)) {
            return back()->with('message', 'Invalid game jam id.');
```

```

    }

    $gameCategories = new GameCategories();
    $gamePlatform = new Platform();

    $this->viewData["gameJamId"] = $idGameJam;
    $this->viewData["gameCategories"] = $gameCategories->getAll();
    $this->viewData["gamePlatforms"] = $gamePlatform->getAll();

    return view('gameSubmissions.createGameSubmission', $this->viewData);
}
//removed one game submission get route from frontend

public function editGameSubmission($id){
    //todo
    //move to own controller
    if (!preg_match("/^\d+$/", $id)) {
        return back()->with('message', 'Invalid game submission id.');
```

}

```

    $gameSubmissions = new GameSubmissions();

    # does game exist
    $gameSubData = $gameSubmissions->getById($id);
    if(empty($gameSubData)){
        return Redirect::back()->withInput()->with('message', 'This game doesn\'t
exist!');
```

}

```

    # is this user creator
    $idUser = session()->get('user')[0]->idUser;
    if($gameSubData->idUserCreator != $idUser){
        return Redirect::back()->withInput()->with('message', 'You cannot edit
this game!');
```

}

```

    $gameCategories = new GameCategories();
    $gamePlatform = new Platform();

    $this->viewData["gameSubmissionId"] = $id;
    $this->viewData["gameCategories"] = $gameCategories->getAll();
    $this->viewData["gamePlatforms"] = $gamePlatform->getAll();
    $this->viewData["gameSubData"] = $gameSubData;
    $this->viewData["gameSubCategories"] = $gameSubmissions-
>getCategoriesIds($id);
    $this->viewData["gameSubPlatform"] = $gameSubmissions-
>getGamePlatformId($id);

    #dd($this->viewData);

    return view('gameSubmissions.editGameSubmission', $this->viewData);
}
//
// auth
```

```

public function register(){
    $roles = new Roles();
    $userRoles = $roles->getAllAvailable();
    $this->viewData['userAvailableRoles'] = $userRoles;
    return view('auth.register', $this->viewData);
}
public function login(){
    return view('auth.login', $this->viewData);
}
//
// other
public function about(){
    return view('other.about', $this->viewData);
}
public function contactUs() {
    $poll = new Polls();
    $question = $poll->getActivePollQuestion();
    if(!empty($question)) {
        $this->viewData['pollQuestion'] = $question;
        $this->viewData['pollAnswers'] = $poll->getAnswersByQuestionId($question-
>idPollQuestion);
    }
    //return view('mailPages.contactUs', $this->viewData);
    return view('other.contactUs', $this->viewData);
}
//
// profile
public function profile() {
    $this->viewData['isEditButtonDisplayed'] = true;
    $users = new Users();

    $this->viewData['userData'] = $users->getById(session()->get('user')[0]-
>idUser);
    $this->viewData['userRoles'] = session()->get('roles')[0];
    return view('user.userProfile', $this->viewData);
}
public function editProfile() {
    $this->viewData['userData'] = session()->get('user')[0];
    $this->viewData['userRoles'] = session()->get('roles')[0];
    $roles = new Roles();
    $userRoles = $roles->getAllAvailable();
    $users = new Users();
    $userHasRolesDb = $users->getAllRoles(session()->get('user')[0]->idUser);
    $userHasRoles = [];
    $this->viewData['userAvailableRoles'] = $userRoles;
    foreach($userHasRolesDb as $e){
        if($e->name != 'admin'){
            $userHasRoles[] = $e->idRole;
        }
    }
    $this->viewData['userHasRoles'] = $userHasRoles;
    return view('user.userEdit', $this->viewData);
}
public function getUserProfileInfo($username)
{
    $this->viewData['isEditButtonDisplayed'] = false;
    $user = new Users();
    $userData = $user->getByUsername($username);
    if(empty($userData))

```

```

        {
            return back()->with('message', 'Sorry but user with that username does
not exist!');
        }
        $userRoles = $user->getAllRoles($userData->idUser);
        $this->viewData['userData'] = $userData;
        $this->viewData['userRoles'] = $userRoles;
        return view('user.userProfile', $this->viewData);
    }
}
...

```

app\Http\Controllers\GameJamController.php

...

<?php

```

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\File;
use App\Http\Models\GameJams;
use App\Http\Models\GameCriteria;
use App\Http\Models\Images;
use Illuminate\Support\Facades\Validator;
use Psy\Util\Json;
use Illuminate\Support\Facades\Redirect;

class GameJamController extends Controller
{
    public function getChartGameJams(Request $request) {
        $gameJamDB = new GameJams();
        $gameJamsResult = $gameJamDB->getAllWhereVotingEndDateNotFinished();

        return Json::encode($gameJamsResult);
    }

    public function getFilteredGameJams(Request $request) {
        // TODO
    }

    public function insert(Request $request) {
        $startDate = strtotime($request->get("dStartDate"));
        $endDate = strtotime($request->get("dEndDate"));
        $votingEndDate = strtotime($request->get("dVotingEndDate"));

        if($startDate < time() + 86400){
            $dateError = "Game jam must start at least 1 day from now.";
        }
        else if($endDate < $startDate + 86400){
            $dateError = "Game jam duration must be at least 1 day.";
        }
    }
}

```

```

    }
    else if($votingEndDate < $endDate + 86400){
        $dateError = "Voting period must be at least 1 day.";
    }

    if(isset($dateError)){
        return back()->withInput()->with('dateError', $dateError);
    }

    $validation = Validator::make($request->all(), [
        'tbTitle' => 'required|regex:/^[a-zA-Z0-9\s]+$/|min:3',
        'taDescription' => 'required',
        'fCoverImage' => 'required|max:2000|mimes:jpg,jpeg,png'
    ]);

    $validation->setAttributeNames([
        'tbTitle' => 'title',
        'taDescription' => 'description',
        'fCoverImage' => 'image'
    ]);

    if($validation->fails()) {
        return back()->withInput()->withErrors($validation);
    } else {
        $photo = $request->file('fCoverImage');
        $extension = $photo->getClientOriginalExtension();
        $tmp_path = $photo->getPathName();

        $folder = 'images/cover/';
        $file_name = time() . "." . $extension;
        $new_path = public_path($folder).$file_name;

        try {
            // insert cover image
            File::move($tmp_path, $new_path);

            $cover = new Images();
            $coverId = $cover->insert(1, 'Cover image',
            'images/cover/'.$file_name);

            // others can vote, locked
            $othersCanVote = $lock = 0;

            $othersCanVote = $request->has('chbOthers') ? 1 : 0;
            $lock = $request->has('chbLock') ? 1 : 0;

            // insert game jam
            $gameJam = new GameJams();

            $gameJamId = $gameJam->insert(
                $request->get('tbTitle'),
                $request->get('taDescription'),
                $coverId,
                $startDate,
                $endDate,
                $votingEndDate,
                $request->get('taContent'),
                $lock,
                $othersCanVote,

```

```

        $request->session()->get('user')[0]->idUser
    );

    if(empty($gameJamId))
    {
        return back()->withInput()->with('error', 'Game jam creation
failed!');
    }

    // insert criterias
    $criterias = $request->get('chbCriteria');
    $gameCriteria = new GameCriteria();

    if(!empty($criterias)) {
        foreach ($criterias as $criteria)
        {
            $gameJam->insertCriteria($gameJamId, $criteria);
        }
    }

    return redirect('/game-jams/create')->with('message', 'Successfully
created Game jam!');
}
catch(\Illuminate\Database\QueryException $ex){
    return redirect()->back()->withInput()->with('error', 'Database
error.');
```

```

    }
    catch(\Symfony\Component\HttpFoundation\File\Exception\FileException $ex)
    {
        return redirect()->back()->withInput()->with('error', 'Failed to
upload image!');
    }
    catch(\ErrorException $ex) {
        return redirect()->back()->withInput()->with('error', 'An error
occured.');
```

```

    }
}

public function joinUserToGameJam($idGameJam) {
    $idUser = session()->get('user')[0]->idUser;

    $gameJams = new GameJams();

    if($gameJams->exist($idGameJam)) {
        if($gameJams->userOwnsGameJam($idUser, $idGameJam)){
            return Redirect::back()->withInput()->with('message', 'You can\'t
join your own Game Jam.');
```

```

        }
        if($gameJams->getById($idGameJam)->endDate < time()){
            return Redirect::back()->withInput()->with('message', 'You can no
longer join this game jam.');
```

```

        }
        else if($gameJams->userAlreadyJoined($idUser, $idGameJam)) {
            return Redirect::back()->withInput()->with('message', 'You are
already in this game jam!');
```

```

        }
        else {
            $result = $gameJams->joinUserToGameJam($idUser, $idGameJam);

```

```

        if(empty($result)) {
            return Redirect::back()->withInput()->with('message', 'Failed to
join game jam!');
        }
        else {
            return Redirect::back()->withInput()->with('message',
'Congratulations, you have joined this game jam!');
        }
    }
    }
    else {
        return Redirect::back()->withInput()->with('message', 'Selected game jam
doesn\'t exist :(');
    }
}

public function removeUserFromGameJam($idGameJam) {
    $idUser = session()->get('user')[0]->idUser;

    $gameJams = new GameJams();

    if($gameJams->exist($idGameJam)) {
        if($gameJams->userAlreadyJoined($idUser, $idGameJam)) {
            $result = $gameJams->removeUserFromGameJam($idUser, $idGameJam);
            return Redirect::back()->withInput()->with('message', 'You have left
this game jam. Good bye!');
        }
        else {
            return Redirect::back()->withInput()->with('message', 'You can\'t
leave because you never joined!');
        }
    }
    else {
        return Redirect::back()->withInput()->with('message', 'Selected game jam
doesn\'t exist :(');
    }
}

public function update(Request $request) {
    $gameJams = new GameJams();
    $idGameJam = $request->get("hiddenIdGameJam");
    $gameJam = $gameJams->getById($idGameJam);

    if($gameJam->startDate < time()){
        return Redirect::back()->withInput()->with("message", "You can no longer
edit this game jam!");
    }
    else {
        if(session()->has('user')){
            $idUser = session()->get('user')[0]->idUser;
            if(!$gameJams->userOwnsGameJam($idUser, $idGameJam)){
                return Redirect::back()->withInput()->with("message", "You can't
edit this game jam!");
            }
        }
    }

    $updateData = [];

```



```

$timeOffset = $request->get("hiddenTimeOffset");

$startDate = strtotime($request->get("dStartDate")) + $timeOffset;
$endDate = strtotime($request->get("dEndDate")) + $timeOffset;
$votingEndDate = strtotime($request->get("dVotingEndDate")) + $timeOffset;

$updateData['startDate'] = intval($startDate);
$updateData['endDate'] = intval($endDate);
$updateData['votingEndDate'] = intval($votingEndDate);

if($startDate < time() + 3600){
    $dateError = "Game jam must start at least 1 hour from now.";
}
else if($endDate < $startDate + 86400){
    $dateError = "Game jam duration must be at least 1 day.";
}
else if($votingEndDate < $endDate + 86400){
    $dateError = "Voting period must be at least 1 day.";
}

if(isset($dateError)){
    return back()->withInput()->with('dateError', $dateError);
}

$validation = Validator::make($request->all(), [
    'tbTitle' => 'required|regex:/^[a-zA-Z0-9\s]+$/|min:3',
    'taDescription' => 'required'
]);

$validation->setAttributeNames([
    'tbTitle' => 'title',
    'taDescription' => 'description'
]);

if($validation->fails()) {
    return back()->withInput()->withErrors($validation);
} else {
    $updateData['title'] = $request->get('tbTitle');
    $updateData['description'] = $request->get('taDescription');
    $updateData['content'] = $request->get('taContent');

    if(!empty($request->file('fCoverImage'))){
        $photo = $request->file('fCoverImage');
        $extension = $photo->getClientOriginalExtension();
        $tmp_path = $photo->getPathName();

        $folder = 'images/cover/';
        $file_name = time() . "." . $extension;
        $new_path = public_path($folder).$file_name;

        try {
            // insert cover image
            File::move($tmp_path, $new_path);

            $cover = new Images();
            $coverId = $cover->insert(1, 'Cover image',
            'images/cover/'.$file_name);

```

```

        $updateData["idCoverImage"] = $coverId;
    }
    catch(\Illuminate\Database\QueryException $ex){
        \Log::error($ex->getMessage());
        return redirect()->back()->with('error','Greska pri dodavanju
posta u bazu!');
    }
    catch(\Symfony\Component\HttpFoundation\File\Exception\FileException
$ex) {
        \Log::error('Problem sa fajlom!!'.$ex->getMessage());
        return redirect()->back()->with('error','Greska pri dodavanju
slike!');
    }
    catch(\ErrorException $ex) {
        \Log::error('Problem sa fajlom!!'.$ex->getMessage());
        return redirect()->back()->with('error','Desila se greska..');
    }
}

// others can vote, locked
$othersCanVote = $lock = 0;

$othersCanVote = $request->has('chbOthers') ? 1 : 0;
$lock = $request->has('chbLock') ? 1 : 0;

$updateData["othersCanVote"] = $othersCanVote;
$updateData["lockSubmissionAfterSubmitting"] = $lock;

// update game jam
$gameJamUpdate = $gameJams->update($idGameJam, $updateData);

if(empty($gameJamUpdate))
{
    return back()->withInput()->with('message', 'Game jam update
failed!');
}

// update criteria
$gameCriteriaGet = $request->get('chbCriteria');
$gameCriteria = new GameCriteria();
$selectedCriteria = [];

if(!empty($gameCriteriaGet)){
    foreach($gameCriteriaGet as $val){
        $selectedCriteria[] = (int)$val;
    }
}

$gameJamHasCriteriaDb = $gameJams->getCriteria($idGameJam);
$deleteCriteria = [];
$newCriteria = [];
$gameJamHasCriteria = [];

foreach($gameJamHasCriteriaDb as $e){
    $gameJamHasCriteria[] = $e->idGameCriteria;
}

// delete criteria
foreach($gameJamHasCriteria as $idGameCriteria){

```

```

        if(!in_array($idGameCriteria, $selectedCriteria)){
            $deleteCriteria[] = $idGameCriteria;
        }
    }

    foreach($deleteCriteria as $idGameCriteria){
        $gameJams->deleteCriteria($idGameJam, $idGameCriteria);
    }

    // add criteria
    foreach($selectedCriteria as $idGameCriteria){
        if(!in_array($idGameCriteria, $gameJamHasCriteria)){
            $newCriteria[] = $idGameCriteria;
        }
    }

    foreach ($newCriteria as $idGameCriteria)
    {
        $gameJams->insertCriteria($idGameJam, $idGameCriteria);
    }

    return redirect("/game-jams/" . $idGameJam)->with('message', 'Game jam
updated successfully!');
    }
}

public function delete($id) {
    $idUser = session()->get('user')[0]->idUser;
    $idGameJam = $id;

    $gameJams = new GameJams();

    if($gameJams->exist($idGameJam)) {
        if(!$gameJams->userOwnsGameJam($idUser, $idGameJam)){
            return Redirect::back()->withInput()->with('message', "You can't
delete this game jam.");
        }
        else if($gameJams->getById($idGameJam)->startDate < time()){
            return Redirect::back()->withInput()->with('message', 'You can\'t
delete an active game jam.');
```

delete this game jam.");

```

        }
        else{
            $gameJams->update($idGameJam, ["isBlocked" => 1]);
            return Redirect::to('/')->withInput()->with('message', "Game jam
deleted.");
        }
    }
    else {
        return Redirect::back()->withInput()->with('message', "Selected game jam
doesn\'t exist :(");
    }
}

...

```

```

#### app\Http\Controllers\GameSubmissionCommentController.php
...

<?php

namespace App\Http\Controllers;

use App\Http\Interfaces\IGameSubmissionComment;
use App\Http\Models\Comment;
use App\Http\Models\GameSubmission_Comment;
use Illuminate\Http\Request;

class GameSubmissionCommentController extends Controller implements
IGameSubmissionComment
{

    public function get(Request $request, $gameId)
    {
        if (!preg_match("/^\d+$/", $gameId)) {
            return response()->json(["error" => ["message" => "Invalid game id!"]],
400);
        }

        try {
            $comments = new GameSubmission_Comment();
            $result = $comments->getByGameSubmissionId($gameId);

            return response()->json($result, 200);
        } catch (\Exception $e) {
            //todo, log this in some file
            return response()->json(null, 500);
        }
    }

    public function add(Request $request, $gameId)
    {
        if (!preg_match("/^\d+$/", $gameId)) {
            return response()->json(["error" => ["message" => "Invalid game id!"]],
400);
        }
        if (!$request->has('text')) {
            return response()->json(["error" => ["message" => "Missing comment
text!"]], 400);
        }

        $commentText = $request->get('text');

        // validate comment text
        if (!preg_match("/^[\\w\\.\\s\\,\\\"\\'\\!\\?\\:\\.\\;\\[\\]]+$/", $commentText)) {
            return response()->json(["error" => ["message" => "Comment text has
unsupported characters!"]], 400);
        }

        //insert comment
        try {

```

```

$time = time();
$userId = $request->attributes->get('userInfo')->idUser;

$comment = new Comments();
$commentId = $comment->insertGetId([
    "text" => $commentText,
    "idUserCreator" => $userId,
    "createdAt" => $time,
    "editedAt" => $time,
]);

if (empty($commentId)) {
    return response()->json(["error" => ["message" => "Comment not
inserted!"]], 500);
}

$gameComment = new GameSubmission_Comment();
$gameCommentId = $gameComment->insertGetId([
    "idGameSubmission" => $gameId,
    "idComment" => $commentId,
]);

$result = $gameComment->getOneById($gameCommentId);

    return response()->json($result, 200);
} catch (\Exception $e) {
    //todo, log this in some file
    return response()->json(null, 500);
}

}

public function edit(Request $request, $gameId, $commentId)
{
    if (!preg_match("/^\d+$/", $gameId)) {
        return response()->json(["error" => ["message" => "Invalid game id!"]],
400);
    }
    if (!preg_match("/^\d+$/", $commentId)) {
        return response()->json(["error" => ["message" => "Invalid comment
id!"]], 400);
    }

    $commentText = $request->get('text');

    // validate comment text
    if (!preg_match("/^[\\w\\.\\s\\,\\\"\\'\\!\\?\\:\\.\\;\\[\\]]+$/", $commentText)) {
        return response()->json(["error" => ["message" => "Comment text has
unsupported characters!"]], 400);
    }

    try {
        $gsComments = new GameSubmission_Comment();
        $commentGSData = $gsComments->getById($commentId);

        //comment doesn't exists
        if (empty($commentGSData)) {

```

```

        return response()->json(["error" => ["message" => "Comment doesn't
exist!"]], 400);
    }

    // comment is not for this game submission
    if ($gameId != $commentGSData->idGameSubmission) {
        return response()->json(["error" => ["message" => "Comment doesn't
exist!"]], 400);
    }

    $comments = new Comments();
    $commentData = $comments->getById($commentGSData->idComment);

    $userId = $request->attributes->get('userInfo')->idUser;

    if ($commentData->idUserCreator != $userId) {
        return response()->json(["error" => ["message" => "You cannot update
comments from different users!"]], 400);
    }

    $comments->update($commentData->idComment, [
        "text"=> $commentText
    ]);

    return response()->json(null, 204);

} catch (\Exception $e) {
    return response()->json(null, 500);
}
}

public function remove(Request $request, $gameId, $commentId)
{
    if (!preg_match("/^\d+$/", $gameId)) {
        return response()->json(["error" => ["message" => "Invalid game id!"]],
400);
    }
    if (!preg_match("/^\d+$/", $commentId)) {
        return response()->json(["error" => ["message" => "Invalid comment
id!"]], 400);
    }

    try {
        $gsComments = new GameSubmission_Comment();
        $commentGSData = $gsComments->getById($commentId);

        //comment doesn't exists
        if (empty($commentGSData)) {
            return response()->json(["error" => ["message" => "Comment doesn't
exist!"]], 400);
        }

        // comment is not for this game submission
        if ($gameId != $commentGSData->idGameSubmission) {
            return response()->json(["error" => ["message" => "Comment doesn't
exist!"]], 400);
        }
    }
}

```

```

    }

    $comments = new Comments();
    $commentData = $comments->getById($commentGSData->idComment);

    $userId = $request->attributes->get('userInfo')->idUser;
    //todo
    if ($commentData->idUserCreator != $userId) {
        return response()->json(["error" => ["message" => "You cannot remove
comments from different users!"]], 400);
    }

    $comments->delete($commentData->idComment);

    return response()->json(null, 204);

} catch (\Exception $e) {
    return response()->json(null, 500);
}
}
}
...

```

```
#### app\Http\Controllers\GameSubmissionController.php
```

```
...
```

```
<?php
```

```

namespace App\Http\Controllers;

use App\Http\Enums\ImageCategories;
use App\Http\Models\DownloadFiles;
use App\Http\Models\GameBadges;
use App\Http\Models\GameJams;
use App\Http\Models\Images;
use App\Http\Models\Navigations;
use App\Http\Interfaces\IGameSubmission;
use App\Http\Models\Reports;
use Illuminate\Http\Request;
use App\Http\Models\GameSubmissions;
use Illuminate\Support\Facades\Input;
use Illuminate\Support\Facades\Redirect;
use Illuminate\Support\Facades\Response;
use Illuminate\Support\Facades\Validator;

class GameSubmissionController extends Controller implements IGameSubmission
{
    private $viewData;

```

```

public function oneGameSubmission($id)
{
    if (!preg_match("/^\d+$/", $id)) {
        return back()->with('message', 'Invalid game submission id.');
```

```
    }

    $gameSubmissions = new GameSubmissions();
    $gameBadges = new GameBadges();
    $gameSubmissions->increaseViews($id);
    $this->viewData["gameSubmission"] = $gameSubmissions->getOne($id);
    $this->viewData["gameSubmissionScreenShots"] = $gameSubmissions-
>getScreenShots($id);
    $this->viewData["gameSubmissionDownloadFiles"] = $gameSubmissions-
>getDownloadFiles($id);
    $this->viewData["gameBadgesList"] = $gameBadges->getAll();

    return view('gameSubmissions.oneGameSubmission', $this->viewData);
}

//
// this function is used as AJAX and normal REQUEST handler
public function getFilteredGames(Request $request)
{
    $page = empty(Input::get("page")) ? 1 : Input::get("page");
    $sortBy = empty(Input::get("sort")) ? "new" : Input::get("sort");
    $sort["name"] = "createdAt";
    $sort["direction"] = "desc";

    switch ($sortBy) {
        case "old":
            $sort["direction"] = "asc";
            break;
        case "top":
            $sort["name"] = "rating";
            break;
        case "views":
            $sort["name"] = "numOfViews";
            break;
        case "download":
            $sort["name"] = "numOfDownloads";
            break;
    }

    $games = new GameSubmissions();
    // todo: add combobox games per page 6 9 12 all?
    $getGames = $games->get(($page - 1) * 9, 9, $sort);

    $this->viewData["games"] = $getGames;
    $this->viewData["gamesCount"] = $games->count();
    $this->viewData["currentPage"] = $page;
    $this->viewData["currentSort"] = $sortBy;

    if ($request->ajax()) {
        return view('ajax.loadGames', $this->viewData)->render();
    }

    //
    // add navigagtion for page only if not ajax

```



```

        // $this->setupNavigation();

        return view('gameSubmissions.games', $this->viewData);
    }

    public function insert(Request $request)
    {
        $validation = Validator::make($request->all(), [
            'tbTitle' => 'required|regex:/^[a-zA-Z0-9\s]+$/|min:3',
            'taDescription' => 'required|regex:/^[\\w\\.\\s\\,\\\"\\'\\!\\?\\:;]+$/|',
            'fCoverImage' => 'required|max:2000|mimes:jpg,jpeg,png',
            'fScreenShots.*' => 'required|max:2000|mimes:jpg,jpeg,png',
            'soGamePlatform' => 'required',
            'fGameFiles' => 'required|max:2000|mimes:zip,rar',
            'hIdGameJam' => 'required',
        ]);

        $validation->setAttributeNames([
            'tbTitle' => 'title',
            'taDescription' => 'description',
            'fCoverImage' => 'cover image',
            'fScreenShots' => 'screen shot',
            'soGamePlatform' => 'game platform',
            'fGameFiles' => 'game file',
        ]);

        if ($validation->fails()) {
            return back()->withInput()->withErrors($validation);
        }

        $gameJams = new GameJams();
        $gameSubmissions = new GameSubmissions();

        $gameJamId = $request->get('hIdGameJam');

        $gj = $gameJams->getById($gameJamId);
        if (empty($gj)) {
            // game jam doesn't exist
            return back()->withInput()->with('message', 'Game jam doesn\'t exist!');
        }

        // check if game jam hasn't pass end date
        if ($gj->endDate < time()) {
            return back()->withInput()->with('message', 'Game jam already ended!');
        }

        //
        $idUser = session()->get('user')[0]->idUser;
        if (!$gameJams->userAlreadyJoined($idUser, $gameJamId)) {
            return back()->withInput()->with('message', 'You have not joined this
game jam!');
        }

        if (!empty($gameSubmissions->getUserAndGameJam($idUser, $gameJamId))) {
            return back()->withInput()->with('message', 'You have already submitted
your game!');
        }
    }

```

```

//this should be file manager but its ok for now
$imagesManager = new Images();

$fScreenShotsPhotos = $request->file('fScreenShots');
if (count($fScreenShotsPhotos) > 8) {
    return back()->withInput()->with('error', 'There is more then 8 screen
shoots!');
}
$screenshotPhotosIds = [];
foreach ($fScreenShotsPhotos as $photo) {
    $screenshotPhotosIds[] = $imagesManager-
>saveImageAndGetId(ImageCategories::$SCREENSHOT, 'Game screen shot', $photo);
}

//save them

$fCoverImage = $request->file('fCoverImage');
$fCoverImageId = $imagesManager->saveImageAndGetId(ImageCategories::$COVER,
'Cover image', $fCoverImage);
// save cover image
// save teaser id as same as cover image for now

//
// save zip or rar file
$fGameFile = $request->file('fGameFiles');
$fGameFilePath = $imagesManager->saveFileInFolder(null, $fGameFile);
$downloadFilesManager = new DownloadFiles();

$fInfo = $fGameFile->getClientOriginalName();
$fGameFileId = $downloadFilesManager->insertGetId([
    "path" => $fGameFilePath,
    "name" => pathinfo($fInfo, PATHINFO_FILENAME),
    "size" => filesize($fGameFilePath),
    "createdAt" => time(),
    "idPlatform" => $request->get('soGamePlatform'),
    "fileExtension" => pathinfo($fInfo, PATHINFO_EXTENSION),
]);
//save it in vezna tabela

$timeCreatedAt = time();

$gameSubmissionsManager = new GameSubmissions();
$idGameSubmission = $gameSubmissionsManager->insertGetId([
    "idGameJam" => $request->get('hIdGameJam'),
    "idTeaserImage" => $fCoverImageId,
    "idCoverImage" => $fCoverImageId,
    "idUserCreator" => $idUser,
    "description" => $request->get('taDescription'),
    "createdAt" => $timeCreatedAt,
    "editedAt" => $timeCreatedAt,
    "numOfViews" => 0,
    "numOfDownloads" => 0,
    "isBlocked" => 0,
    "numberOfVotes" => 0,
    "sumOfVotes" => 0,
    "title" => $request->get('tbTitle'),
    "isWinner" => 0,
]);

```

```

//saving screen shots to table
foreach ($screenShotPhotosIds as $ssId) {
    $imagesManager->insertScreenShots($idGameSubmission, $ssId);
}

//saving download file
$gameSubmissionsManager->saveRelationshipWithDownloadFile($idGameSubmission,
$fGameFileId);

//save game categories
$cbGameCategories = $request->get('cbCategories');
foreach ($cbGameCategories as $c) {
    $gameSubmissionsManager->saveGameCategories([
        "idGameSubmission" => $idGameSubmission,
        "idCategory" => $c,
    ]);
}

return redirect('/game-jams/' . $gameJamId)->with('message', 'Successfully
created Game Submission');
}

public function edit(Request $request, $id)
{
    if (!preg_match("/^\d+$/", $id)) {
        return back()->with('message', 'Invalid game submission id.');
```

```

$gameSubmissions = new GameSubmissions();

# does game exist
$gameSubData = $gameSubmissions->getById($id);

if (empty($gameSubData)) {
    return Redirect::back()->withInput()->with('message', 'This game doesn\'t
exist!');
}

# is this user creator
$idUser = session()->get('user')[0]->idUser;
if ($gameSubData->idUserCreator != $idUser) {
    return Redirect::back()->withInput()->with('message', 'You cannot edit
this game!');
}

//data to save
$gsUpdateData = [];

$imagesManager = new Images();
// images to save

$fCoverImage = $request->file('fCoverImage');
$fCoverImageId = null;
if (!empty($fCoverImage)) {
    $fCoverImageId = $imagesManager-
>saveImageAndGetId(ImageCategories::$COVER, 'Cover image', $fCoverImage);
    $gsUpdateData["idTeaserImage"] = $fCoverImageId;
    $gsUpdateData["idCoverImage"] = $fCoverImageId;
}
//if fcover image id exist concat replace data

// - title
// - description
$gsUpdateData["title"] = $request->get('tbTitle');
$gsUpdateData["description"] = $request->get('taDescription');

// - category list (delete all and then insert selected ones)
$gameSubmissions->removeAllCategories($id);
$cbGameCategories = $request->get('cbCategories');
foreach ($cbGameCategories as $c) {
    $gameSubmissions->saveGameCategories([
        "idGameSubmission" => $id,
        "idCategory" => $c,
    ]);
}

// - game platform

$fGameFile = $request->file('fGameFiles');
$downloadFilesManager = new DownloadFiles();
if (!empty($fGameFile)) {

```

```

        $fGameFilePath = $imagesManager->saveFileInFolder(null, $fGameFile);

        $fInfo = $fGameFile->getClientOriginalName();
        $fGameFileId = $downloadFilesManager->insertGetId([
            "path" => $fGameFilePath,
            "name" => pathinfo($fInfo, PATHINFO_FILENAME),
            "size" => filesize($fGameFilePath),
            "createdAt" => time(),
            "idPlatform" => $request->get('soGamePlatform'),
            "fileExtension" => pathinfo($fInfo, PATHINFO_EXTENSION),
        ]);

        $gameSubmissions->removeRelationshipWithDownloadFile($id);
        $gameSubmissions->saveRelationshipWithDownloadFile($id, $fGameFileId);
    } else {
        $idDownloadFile = $gameSubmissions->getDownloadFileIdByGameId($id);
        $downloadFilesManager->update($idDownloadFile->idDownloadFile, [
            "idPlatform" => $request->get('soGamePlatform'),
        ]);
    }
}

/*
 * This is section for saving screen shots
 * This should be changed, but for know its ok
 * */

$fScreenShotsPhotos = $request->file('fScreenShots');
if (count($fScreenShotsPhotos) > 8) {
    return back()->withInput()->with('error', 'There is more then 8 screen
shoots!');
}
$screenShotPhotosIds = [];
foreach ($fScreenShotsPhotos as $photo) {
    $screenShotPhotosIds[] = $imagesManager-
>saveImageAndGetId(ImageCategories::$SCREENSHOT, 'Game screen shot', $photo);
}

//remove all files
$imagesManager->removeScreenShots($id);

//saving screen shots to table
foreach ($screenShotPhotosIds as $ssId) {
    $imagesManager->insertScreenShots($id, $ssId);
}

//
// update data
$gameSubmissions->update($id, $gsUpdateData);

//dd("cool");
return Redirect::back()->with('message', 'Successfully updated!');
}

public function delete(Request $request, $id)
{

```

```

    $idUser = session()->get('user')[0]->idUser;

    $gameJams = new GameJams();
    $gameSubmissions = new GameSubmissions();

    if ($gameSubmissions->exist($id)) {
        $gsData = $gameSubmissions->getById($id);

        if ($gsData->idUserCreator != $idUser || $gsData->isBlocked == 1) {
            return Redirect::back()->withInput()->with('message', "You can't
delete this game submission.");
        } else if ($gameJams->getById($gsData->idGameJam)->endDate < time()) {
            return Redirect::back()->withInput()->with('message', 'You can\'t
delete an active game jam.');
        } else {
            $gameSubmissions->update($id, ["isBlocked" => 1]);
            return Redirect::to('/')->withInput()->with('message', "Game deleted
successfully.");
        }
    } else {
        return Redirect::back()->withInput()->with('message', "Selected game
doesn\'t exist!");
    }
}

public function downloadFile(Request $request, $idDownloadFile)
{
    if (!preg_match("/^\d+$/", $idDownloadFile)) {
        return back()->with('message', 'Invalid download file id.');
```

```

        'taReason' => 'reason',
        'gameId' => 'game',
    ]);

    if ($validation->fails()) {
        return back()->withInput()->withErrors($validation);
    }

    $reason = $request->get('taReason');
    $idGame = $request->get('gameId');
    $idUser = session()->get('user')[0]->idUser;

    //if game exists
    $gameSubmissions = new GameSubmissions();
    if (!$gameSubmissions->exist( $idGame )) {
        return back()->withInput()->with('message', 'Game doesn\'t exist.');
```

```

    }

    //da nije vec reportovo
    $reportManager = new Reports();
    $hasReported = $reportManager->userHasReportedGame($idGame, $idUser);
```

```

    if($hasReported){
        return back()->withInput()->with('message', 'You already reported this
game.');
```

```

    }

    //save it
    $reportManager->insertGetId([
        "reason"=> $reason,
        "idUserCreator"=> $idUser,
        "idReportObject"=> $idGame,
        "createdAt"=> time(),
        "solved"=> 0,
    ]);
```

```

    return back()->with('message', 'You have successfully reported the game.');
```

```

}

/*private function setupNavigation(){
    $navs = new Navigations();
    $this->viewData['navigation'] = $navs->getAllSortedByPosition();
}*/
```

```

}

#### app\Http\Controllers\ProfileController.php
```

```

...
```

```

<?php
```

```

namespace App\Http\Controllers;

use App\Http\Interfaces\IProfile;
use App\Http\Models\GameJams;
use App\Http\Models\GameSubmissions;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\File;
use App\Http\Models\Roles;
use App\Http\Models\Users;
use App\Http\Models\Images;
use Illuminate\Support\Facades\Validator;

class ProfileController extends Controller implements IProfile
{
    private $viewData = [];

    public function edit(Request $request){
        $idUser = session()->get('user')[0]->idUser;
        $users = new Users();
        $updateData = [];

        $currentUser = $users->getById($idUser);

        // change password
        if(!empty($request->get('tbCurrentPassword'))){
            {
                if(md5($request->get('tbCurrentPassword')) === $currentUser->password){
                    // update password if field is not empty
                    if(!empty($request->get('tbPassword')) && !empty($request->
>get('tbPassword_confirmation'))){
                        $validacija = Validator::make($request->all(), [
                            'tbPassword' => 'required|confirmed|min:6',
                        ]);
                        $validacija->setAttributeNames([
                            'tbPassword' => 'password',
                        ]);

                        if ($validacija->fails()) {
                            // redirekcija na pocetnu i ispis gresaka
                            return back()->withInput()->withErrors($validacija);
                        }

                        $updateData['password'] = md5($request->get('tbPassword'));
                    }
                }
            }
            else{
                return back()->withInput()->with('error', 'Wrong password!');
            }
        }

        // update roles
        $userRoles = $request->get('userRoles');
        $selectedRoles = [];

        if(!empty($userRoles)){
            foreach($userRoles as $val){
                $selectedRoles[] = (int)$val;
            }
        }
    }
}

```



```

    }

    $userHasRolesDb = $users->getAllRoles($idUser);
    $deleteRoles = [];
    $newRoles = [];
    $userHasRoles = [];

    foreach($userHasRolesDb as $e){
        if($e->name != 'admin'){
            $userHasRoles[] = $e->idRole;
        }
    }

    // delete roles
    foreach($userHasRoles as $idRole){
        if(!in_array($idRole, $selectedRoles) && $idRole != 1){
            $deleteRoles[] = $idRole;
        }
    }

    foreach($deleteRoles as $idRole){
        $users->deleteRole($idUser, $idRole);
    }

    // add roles
    foreach($selectedRoles as $idRole){
        if(!in_array($idRole, $userHasRoles) && $idRole != 1){
            $newRoles[] = $idRole;
        }
    }

    foreach ($newRoles as $idRole)
    {
        $users->addRole($idRole, $idUser);
    }

    $userHasRolesDb = $users->getAllRoles($idUser);

    $request->session()->forget('roles');
    $request->session()->push("roles", $userHasRolesDb);
    //

    // update avatar
    if(!empty($request->file('fAvatarImage'))){
        $photo = $request->file('fAvatarImage');
        $extension = $photo->getClientOriginalExtension();
        $tmp_path = $photo->getPathName();

        $folder = 'images/avatars/';
        $file_name = $request->session()->get('user')[0]->username . "_" . time()
        . ". " . $extension;
        $new_path = public_path($folder) . $file_name;

        try {
            // insert avatar image
            File::move($tmp_path, $new_path);

            $updateData['avatarImagePath'] = 'images/avatars/' . $file_name;

```

```

    }
    catch(\Symfony\Component\HttpFoundation\File\Exception\FileException $ex)
    {
        \Log::error('File error!'.$ex->getMessage());
        return redirect()->back()->with('error','Error with image file!');
    }
    catch(\ErrorException $ex) {
        \Log::error('File error!'.$ex->getMessage());
        return redirect()->back()->with('error','Error');
    }
}

// update user
$userId = $users->updateUser($idUser, $updateData);

if(empty($userId))
{
    return back()->withInput()->with('error', 'Update failed!');
}

return back()->withInput()->with('messages', 'Successfully updated!');
}

public function getUsersGameJams(Request $request, $username) {
    $user = new Users();
    $usersData = $user->getIdByUsername($username);
    $userId = $usersData->idUser;

    if(empty($userId))
    {
        return back()->with('message', 'User with this username doesn\'t
exist!');
    }

    $page = empty($request->get("page")) ? 1 : $request->get("page");

    $gameJams = new GameJams();
    $gjs = $gameJams->getAllUsersGameJams($userId, ($page - 1) * 6, 6);

    $this->viewData["upcomingGameJams"] = $gjs["result"];
    $this->viewData["gamesJamsUpcomingCount"] = $gjs["count"];
    $this->viewData["currentPageGameJamsUpcoming"] = $page;

    return view('user.usersGameJams', $this->viewData);
}

public function getUsersGames(Request $request, $username)
{
    $user = new Users();
    $usersData = $user->getIdByUsername($username);
    $userId = $usersData->idUser;

    if(empty($userId))
    {
        return back()->with('message', 'User with this username doesn\'t
exist!');
    }

    $page = empty($request->get("page")) ? 1 : $request->get("page");

```

```

        $gameSubmissions = new GameSubmissions();
        $gss = $gameSubmissions->getAllUsersGameSubmissions($userId, ($page - 1) * 6,
6);

        $this->viewData["games"] = $gss["result"];
        $this->viewData["gamesCount"] = $gss["count"];
        $this->viewData["currentPage"] = $page;
        $this->viewData["currentSort"] = "none";

        return view('user.usersGames', $this->viewData);
    }

    public function getUsersWins(Request $request, $username){
        $user = new Users();
        $userData = $user->getIdByUsername($username);
        $userId = $userData->idUser;

        if(empty($userId))
        {
            return back()->with('message', 'User with this username doesn\'t
exist!');
        }

        $page = empty($request->get("page")) ? 1 : $request->get("page");

        $gameSubmissions = new GameSubmissions();
        $gss = $gameSubmissions->getAllUsersGameSubmissionsWins($userId, ($page - 1)
* 6, 6);

        $this->viewData["games"] = $gss["result"];
        $this->viewData["gamesCount"] = $gss["count"];
        $this->viewData["currentPage"] = $page;
        $this->viewData["currentSort"] = "none";

        return view('user.usersWins', $this->viewData);
    }
}

...

```

```

#### app\Http\Controllers\SearchController.php

```

```

...

```

```

<?php

```

```

namespace App\Http\Controllers;

use App\Http\Interfaces\ISearch;

```

```

use App\Http\Models\GameJams;
use App\Http\Models\GameSubmissions;
use Illuminate\Http\Request;

class SearchController extends Controller implements ISearch
{
    private $viewData = [];

    public function search(Request $request)
    {
        // make it more global
        // maybe some utilities model
        $pageSizeConfig = 6;

        $query = $request->query("q");
        if (empty($query)) {
            return back()->with('message', 'Please insert a search value first.');
```

value.');

```

        } else if (!preg_match("/^\w+$/", $query)) {
            return back()->with('message', 'Please insert only a text for search

        $gjManager = new GameJams();
        $gsManager = new GameSubmissions();

        $this->viewData["searchedText"] = $query;

        // so exec only some stuff
        $execGJ = true;
        $execGS = true;

        if ($request->ajax()) {
            $type = $request->query("type");
            switch ($type) {
                case "gameSubmissions":
                    $execGJ = false;
                    break;
                case "gameJams":
                    $execGS = false;
                    break;
            }
        }

        $pagePosition = $request->query("page");
        $pagePosition = empty($pagePosition) ? 1 : $pagePosition;

        if ($execGJ) {
            //
            // game jams stuff

            $this->viewData["gameJams"] = $gjManager->getAllSearched($query,
($pagePosition - 1) * $pageSizeConfig, $pageSizeConfig);
            $this->viewData["gamesJamsCount"] = $gjManager->countAllSearched($query);
            $this->viewData["currentPageGameJams"] = $pagePosition;
        }

        if ($execGS) {

```

```

        //
        // game submission stuff

        $this->viewData["gameSubmissions"] = $gsManager->getAllSearched($query,
($pagePosition - 1) * $pageSizeConfig, $pageSizeConfig);
        $this->viewData["gameSubmissionsCount"] = $gsManager-
>countAllSearched($query);
        $this->viewData["currentPageGameSubmissions"] = $pagePosition;
    }

    if ($request->ajax()) {
        if (!$execGS) {
            return view('ajax.searchGameJams', $this->viewData)->render();
        }
        else if (!$execGJ) {
            return view('ajax.searchGameSubmissions', $this->viewData)->render();
        }
    }

    return view('other.search', $this->viewData);
}

}

...

```

```

#### app\Http\Controllers\Admin\AdminController.php

```

```

...

```

```

<?php

```

```

namespace App\Http\Controllers\Admin;

use \App\Http\Interfaces\Admin\IAdmin;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Http\Models\Generic;
use App\Http\Models\PollQuestions;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\File;

class AdminController extends Controller implements IAdmin
{
    private $viewData = [];

    private $columns = [
        "gamecategories" => [
            // field name, field type
            ["name", "text"]
        ],
        "gamecriteria" => [
            ["name", "text"],

```

```

        ["description", "text"]
    ],
    "imagecategories" => [
        ["name", "text"]
    ],
    "roles" => [
        ["name", "text"],
        ["text", "text"],
        ["isAvailableForUser", "checkbox"],
        ["description", "text"]
    ],
    "users" => [
        ["email", "text"],
        ["username", "text"],
        ["password", "text"],
        ["avatarImagePath", "text"],
        ["isBanned", "checkbox"]
    ],
    "platforms" => [
        ["name", "text"],
        ["classNameForIcon", "text"]
    ],
    "navigations" => [
        ["path", "text"],
        ["name", "text"],
        ["position", "text"]
    ],
    "reports" => [
        ["solved", "checkbox"]
    ],
    "pollquestions" => [
        ["text", "text"]
    ],
    "pollanswers" => [
        ["text", "text"],
        ["idPollQuestion", "number"]
    ]
];

public static function getTypeByTableName($tableName){
    $result = "App\\Http\\Models\\";
    $c = "";
    switch ($tableName){
        case 'comments': $c = "Comments"; break;
        case 'downloadfiles': $c = "DownloadFiles"; break;
        case 'gamebadges': $c = "GameBadges"; break;
        case 'gamecategories': $c = "GameCategories"; break;
        case 'gamecriteria': $c = "GameCriteria"; break;
        case 'gamejams': $c = "GameJams"; break;
        case 'gamesubmission_comment': $c = "GameSubmission_Comment"; break;
        case 'gamesubmissions': $c = "GameSubmissions"; break;
        case 'generic': $c = "Generic"; break;
        case 'imagecategories': $c = "ImageCategories"; break;
        case 'images': $c = "Images"; break;
        case 'navigations': $c = "Navigations"; break;
        case 'platform': $c = "Platform"; break;
        case 'platforms': $c = "Platforms"; break;
        case 'pollanswers': $c = "PollAnswers"; break;
        case 'pollquestions': $c = "PollQuestions"; break;
    }
}

```

```

        case 'polls': $c = "Polls"; break;
        case 'reports': $c = "Reports"; break;
        case 'roles': $c = "Roles"; break;
        case 'statistics': $c = "Statistics"; break;
        case 'users': $c = "Users"; break;
        case 'utilities': $c = "Utilities"; break;
    }

    return $result . $c;
}

public function index($page = null) {
    if(empty($page)) {
        return view("admin.index", $this->viewData);
    }

    return AdminController::getAll(str_replace("-", "", $page));
}

public function block(Request $request) {
    $id = $request->get("id");
    $table = $request->get("tableName");
    $view = explode(".", $request->get("viewName"))[1];
    //$type = "App\\Http\\Models\\" . $table;
    $type = AdminController::getTypeByTableName($table);

    $model = new $type($table);

    $model->update($id, ["isBlocked" => $request->get("isBlocked")]);

    return AdminController::getAll($table, "ajax." . $view);
}

public function insert(Request $request) {
    $table = $request->get("tableName");
    $view = explode(".", $request->get("viewName"))[1];

    $model = new Generic($table);

    $insertData = [];

    foreach($this->columns[$table] as $column) {
        $value = $request->get($column[0]);
        if($column[1] === "checkbox") {
            $insertData[$column[0]] = $request->has($column[0]) ? 1 : 0;
        }
        else if(!empty($value)) {
            if($column[1] === "text" || $column[1] === "number") {
                $insertData[$column[0]] = $value;
            }
        }
    }

    if(count($insertData)){
        $result = $model->insertGetId($insertData);

        if(empty($result)) {
            return response()->json(["message" => "Failed to insert data."],
500);

```

```

    }
}
else {
    return response()->json(["message" => "No data provided."], 400);
}

return AdminController::getAll($table, "ajax." . $view);
}

public function update(Request $request) {
    $id = $request->has("hiddenId") ? $request->get("hiddenId") : $request-
>get("id");
    $table = $request->get("tableName");
    $view = explode(".", $request->get("viewName"))[1];

    $model = new Generic($table);

    $updateData = [];

    foreach($this->columns[$table] as $column) {
        $value = $request->get($column[0]);
        if($column[1] === "checkbox") {
            $updateData[$column[0]] = $request->has($column[0]) ? 1 : 0;
        }
        else if(!empty($value)) {
            if($column[1] === "text") {
                $updateData[$column[0]] = $value;
            }
        }
    }

    $result = $model->update($id, $updateData);

    if(empty($result)) {
        return response()->json(["message" => "Not updated."], 500);
    }

    return AdminController::getAll($table, "ajax." . $view);
}

public function delete(Request $request) {
    $ids = $request->get("ids");
    $count = count(explode(".", $request->get("viewName")));
    $view = $count > 1 ? explode(".", $request->get("viewName"))[1] : $request-
>get("viewName");

    $table = $request->get("tableName");

    $generic = new Generic($table, null);

    $generic->deleteMultiple($ids);

    return AdminController::getAll($table, "ajax." . $view);
}

public function getById(Request $request) {
    $id = $request->get("id");
    $table = $request->get("tableName");

```



```

        // $type = "App\\Http\\Models\\" . $table;
        $type = AdminController::getTypeByTableName($table);
        $model = new $type($table);

        $result = $model->getById($id);

        return !empty($result) ? response()->json(json_encode($result), 200) :
response()->json(null, 500);
    }

    public static function getAll($table, $view = null) {
        $table = strtolower($table);

        $view = empty($view) ? $table : $view;
        // $type = "App\\Http\\Models\\" . $table;

        $type = AdminController::getTypeByTableName($table);

        $model = new $type($table);
        $result = null;

        $result = $model->getAll();

        return view("admin." . $view, ["tableData" => $result, "tableName" =>
$table])->render();
    }
}

```

```
#### app\Http\Controllers\Admin\InsertController.php
```

```
...
```

```
<?php
```

```

namespace App\Http\Controllers\Admin;

use \App\Http\Interfaces\Admin\IInsert;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\File;
use App\Http\Models\Generic;
use App\Http\Models\Users;

class InsertController extends AdminController implements IInsert
{
    private $viewData = [];

    public function users(Request $request)
    {
        $validacija = Validator::make($request->all(), [

```

```

        'tbEmail' => 'required|email|unique:users,email',
        'tbUsername' => 'required|unique:users,username',
        'tbPassword' => 'required|min:6',
    ]);
    $validacija->setAttributeNames([
        'tbEmail' => 'email',
        'tbUsername' => 'username',
        'tbPassword' => 'password',
    ]);

    if ($validacija->fails()) {
        return response()->json(["error" => ["message" => $validacija]], 500);
    }

    $userRoles = $request->get('userRoles');
    $user = new Users();

    $avatar = null;

    if(!empty($request->file('fAvatarImage'))){
        $photo = $request->file('fAvatarImage');
        $extension = $photo->getClientOriginalExtension();
        $tmp_path = $photo->getPathName();

        $folder = 'images/avatars/';
        $file_name = "changed" . "_" . time() . "." . $extension;
        $new_path = public_path($folder) . $file_name;

        try {
            // insert avatar image
            File::move($tmp_path, $new_path);

            $avatar = 'images/avatars/'.$file_name;
        }
        catch(\Symfony\Component\HttpFoundation\File\Exception\FileException $ex)
        {
            \Log::error('File error!'.$ex->getMessage());
            return response()->json(["error" => ["message" => "File error!"]],
500);
        }
        catch(\ErrorException $ex) {
            \Log::error('File error!'.$ex->getMessage());
            return response()->json(["error" => ["message" => "Error!"]], 500);
        }
    }

    $isBanned= $request->has('isBanned') ? 1 : 0;

    $userId = $user->insert(
        strtolower($request->get('tbEmail')),
        strtolower($request->get('tbUsername')),
        $request->get('tbPassword'),
        $isBanned,
        $avatar
    );

    //something went wrong and user isn't inserted
    if (empty($userId)) {
        return response()->json(["message" => "User not updated!"], 500);
    }

```

```

    }

    //add roles
    if (!empty($userRoles)) {
        foreach ($userRoles as $role) {
            $user->addRole($role, $userId);
        }
    }

    return AdminController::getAll("users", "ajax.users");
}

public function gameCategories(Request $request) {
    return parent::insert($request);
}

public function gameCriteria(Request $request) {
    return parent::insert($request);
}

public function roles(Request $request) {
    return parent::insert($request);
}

public function imageCategories(Request $request) {
    return parent::insert($request);
}

public function platforms(Request $request) {
    return parent::insert($request);
}

public function navigations(Request $request) {
    return parent::insert($request);
}

public function pollquestions(Request $request) {
    return parent::insert($request);
}

public function pollanswers(Request $request) {
    return parent::insert($request);
}
}
...

```

app\Http\Controllers\Admin\StatisticsController.php

...

<?php

```

namespace App\Http\Controllers\Admin;

use App\Http\Models\Statistics;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class StatisticsController extends Controller
{
    private $tablesMetaData = [];

    public function __construct()
    {
        $this->tablesMetaData = [
            ["resultName" => "users", "tableName"=> "users", "tableTimeColumnName"=>
"createdAt"],//done
            ["resultName" => "gameJams", "tableName"=> "gamejams",
"tableTimeColumnName"=> "createdAt"],//done
            ["resultName" => "games", "tableName"=> "gamesubmissions",
"tableTimeColumnName"=> "createdAt"],//done
            ["resultName" => "reports", "tableName"=> "reports",
"tableTimeColumnName"=> "createdAt"],//done

            ["resultName" => "comments", "tableName"=> "comments",
"tableTimeColumnName"=> "createdAt"],//done
            ["resultName" => "downloadFiles", "tableName"=> "downloadfiles",
"tableTimeColumnName"=> "createdAt"],//done
            ["resultName" => "images", "tableName"=> "images",
"tableTimeColumnName"=> "createdAt"],//done
            ["resultName" => "polls", "tableName"=> "pollvotes",
"tableTimeColumnName"=> "createdAt"], //done
        ];
    }

    public function getAllChart(Request $request)
    {
        try {
            $result = [];
            //get from all tables statistics

            $statManager = new Statistics();

            foreach($this->tablesMetaData as $item){
                $result[] = [$item['resultName'] => $statManager->getForChart($item['tableName'], $item['tableTimeColumnName'])];
            }

            return response()->json($result, 200);
        } catch (\Exception $e) {
            //todo, log this in some file
            return response()->json($e, 500);
        }
    }

    public function getAllCount(Request $request)
    {
        try {
            $result = [];
            //get from all tables statistics

```

```

        $statManager = new Statistics();

        foreach($this->tablesMetaData as $item){
            $result[] = [$item['resultName'] => $statManager->getTableCount($item['tableName'])];
        }

        return response()->json($result, 200);
    } catch (\Exception $e) {
        //todo, log this in some file
        return response()->json(null, 500);
    }
}
}
...

```

app\Http\Controllers\Admin\UpdateController.php

...

<?php

```

namespace App\Http\Controllers\Admin;

use \App\Http\Interfaces\Admin\IUpdate;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\File;
use App\Http\Models\Generic;
use App\Http\Models\Users;
use App\Http\Models\Polls;

class UpdateController extends AdminController implements IUpdate
{
    private $viewData = [];

    public function users(Request $request) {
        $idUser = $request->get("hiddenId");

        $users = new Users();
        $updateData = [];

        $user = $users->getById($idUser);

        $username = $user->username;

        // change username
        if($user->username !== $request->get("tbUsername")) {
            $validacija = Validator::make($request->all(), [
                'tbUsername' => 'required|unique:users,username'
            ]);
        }
    }
}

```

```

    });
    $validacija->setAttributeNames([
        'tbUsername' => 'username',
    ]);

    if ($validacija->fails()) {
        return response()->json(["message" => "Bad username."], 500);
    }

    $username = $request->get('tbUsername');
    $updateData['username'] = $request->get('tbUsername');
}

// change email
if($user->email !== $request->get("tbEmail")) {
    $validacija = Validator::make($request->all(), [
        'tbEmail' => 'required|email|unique:users,email'
    ]);
    $validacija->setAttributeNames([
        'tbEmail' => 'e-mail',
    ]);

    if ($validacija->fails()) {
        return response()->json(["message" => "Bad e-mail."], 500);
    }

    $updateData['email'] = $request->get('tbEmail');
}

// change password
if(!empty($request->get('tbPassword')))
{
    // update password if field is not empty
    if(!empty($request->get('tbPassword'))){
        $validacija = Validator::make($request->all(), [
            'tbPassword' => 'required|min:6',
        ]);
        $validacija->setAttributeNames([
            'tbPassword' => 'password',
        ]);

        if ($validacija->fails()) {
            return response()->json(["message" => "Bad password."], 500);
        }

        $updateData['password'] = md5($request->get('tbPassword'));
    }
}

// update roles
$userRoles = $request->get('userRoles');
$selectedRoles = [];

if(!empty($userRoles)){
    foreach($userRoles as $val){
        $selectedRoles[] = (int)$val;
    }
}

```

```

$userHasRolesDb = $users->getAllRoles($idUser);
$deleteRoles = [];
$newRoles = [];
$userHasRoles = [];

foreach($userHasRolesDb as $e){
    if($e->name){
        $userHasRoles[] = $e->idRole;
    }
}

// delete roles
foreach($userHasRoles as $idRole){
    if(!in_array($idRole, $selectedRoles)){
        $deleteRoles[] = $idRole;
    }
}

foreach($deleteRoles as $idRole){
    $users->deleteRole($idUser, $idRole);
}

// add roles
foreach($selectedRoles as $idRole){
    if(!in_array($idRole, $userHasRoles)){
        $newRoles[] = $idRole;
    }
}

foreach ($newRoles as $idRole)
{
    $users->addRole($idRole, $idUser);
}

// update avatar
if(!empty($request->file('fAvatarImage'))){
    $photo = $request->file('fAvatarImage');
    $extension = $photo->getClientOriginalExtension();
    $tmp_path = $photo->getPathName();

    $folder = 'images/avatars/';
    $file_name = $username . "_" . time() . "." . $extension;
    $new_path = public_path($folder) . $file_name;

    try {
        // insert avatar image
        File::move($tmp_path, $new_path);

        $updateData['avatarImagePath'] = 'images/avatars/' . $file_name;
    }
    catch(\Symfony\Component\HttpFoundation\File\Exception\FileException $ex)
    {
        \Log::error('File error!'.$ex->getMessage());
        return response()->json(["message" => "File error."], 500);
    }
    catch(\ErrorException $ex) {
        \Log::error('File error!'.$ex->getMessage());
        return response()->json(["message" => "Error."], 500);
    }
}

```

```

    }

    $updateData['isBanned'] = $request->has('chbIsBanned') ? 1 : 0;

    // update user
    $userId = $users->updateUser($idUser, $updateData);

    if(empty($userId))
    {
        return response()->json(["message" => "User not updated!"], 500);
    }

    return AdminController::getAll("users", "ajax.users");
}

public function gameCategories(Request $request) {
    return parent::update($request);
}

public function gameCriteria(Request $request) {
    return parent::update($request);
}

public function reports(Request $request) {
    return parent::update($request);
}

public function roles(Request $request) {
    return parent::update($request);
}

public function imageCategories(Request $request) {
    return parent::update($request);
}

public function platforms(Request $request) {
    return parent::update($request);
}

public function navigations(Request $request) {
    return parent::update($request);
}

public function pollquestions(Request $request) {
    return parent::update($request);
}

public function pollanswers(Request $request) {
    return parent::update($request);
}

public function setActivePollQuestion(Request $request) {
    $model = new Polls();
    $model->setActivePollQuestion($request->get("id"));

    return AdminController::getAll("pollquestions", "ajax.polls");
}
}

```


...

app\Http\Enums\ImageCategories.php

...

<?php

namespace App\Http\Enums;

class ImageCategories

{

public static \$COVER = 1;

public static \$AVATAR = 2;

public static \$TEASER = 3;

public static \$BADGE = 4;

public static \$SCREENSHOT = 5;

public static function getSavingFolderByEnum(\$category)

{

 \$result = 'images/unknown/';

 switch (\$category)

 {

 case null:

 \$result = 'downloads/';

 break;

 case ImageCategories::\$COVER:

 \$result = 'images/cover/';

 break;

 case ImageCategories::\$AVATAR:

 \$result = 'images/avatars/';

 break;

 case ImageCategories::\$TEASER:

 \$result = 'images/teasers/';

 break;

 case ImageCategories::\$BADGE:

 \$result = 'images/badges/';

 break;

 case ImageCategories::\$SCREENSHOT:

 \$result = 'images/screenshots/';

 break;

 }

 return \$result;

}

}

...

```
#### app\Http\Interfaces\IAuthorization.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Interfaces;
```

```
use Illuminate\Http\Request;
```

```
interface IAuthorization
```

```
{
```

```
    public function login(Request $request);
```

```
    public function logout(Request $request);
```

```
    public function register(Request $request);
```

```
}
```

```
...
```

```
#### app\Http\Interfaces\IBadge.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Interfaces;
```

```
use Illuminate\Http\Request;
```

```
interface IBadge
```

```
{
```

```
    public function get(Request $request, $gameId);
```

```
    public function add(Request $request, $gameId, $badgeId);
```

```
    public function remove(Request $request, $gameId, $badgeId);
```

```
}
```

```
...
```

```
#### app\Http\Interfaces\IContactUs.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Interfaces;
```

```
use Illuminate\Http\Request;

interface IContactUs
{
    public function pollVote(Request $request);
    public function postContact(Request $request);
}
    ...
```

```
#### app\Http\Interfaces\IGameJam.php
```

```
    ...
```

```
<?php
```

```
namespace App\Http\Interfaces;

use Illuminate\Http\Request;

interface IGameJam extends IGeneric
{
}
    ...
```

```
#### app\Http\Interfaces\IGameSubmission.php
```

```
    ...
```

```
<?php
```

```
namespace App\Http\Interfaces;

use Illuminate\Http\Request;

interface IGameSubmission extends IGeneric
{
}
    ...
```

```
#### app\Http\Interfaces\IGameSubmissionComment.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Interfaces;
```

```
use Illuminate\Http\Request;
```

```
interface IGameSubmissionComment
```

```
{
```

```
    public function get(Request $request, $gameId);
```

```
    public function add(Request $request, $gameId);
```

```
    public function edit(Request $request, $gameId, $commentId);
```

```
    public function remove(Request $request, $gameId, $commentId);
```

```
}
```

```
...
```

```
#### app\Http\Interfaces\IGeneric.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Interfaces;
```

```
use Illuminate\Http\Request;
```

```
interface IGeneric
```

```
{
```

```
    public function insert(Request $request);
```

```
    public function edit(Request $request, $id);
```

```
    public function delete(Request $request, $id);
```

```
}
```

```
...
```

```
#### app\Http\Interfaces\IProfile.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Interfaces;
```

```
use Illuminate\Http\Request;
```

```

interface IProfile
{
    public function edit(Request $request);
}
    ...

```

```

#### app\Http\Interfaces\ISearch.php
    ...

```

```

    <?php

```

```

namespace App\Http\Interfaces;

use Illuminate\Http\Request;

interface ISearch
{
    public function search(Request $request);
}
    ...

```

```

#### app\Http\Interfaces\Admin\IAdmin.php
    ...

```

```

    <?php

```

```

namespace App\Http\Interfaces\Admin;

use Illuminate\Http\Request;

interface IAdmin
{
    public function index($page = null);
    public function insert(Request $request);
    public function update(Request $request);
    public function delete(Request $request);
    public function getById(Request $request);
    public static function getAll($table, $view = null);
}
    ...

```

```
#### app\Http\Interfaces\Admin\IInsert.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Interfaces\Admin;
```

```
use Illuminate\Http\Request;
```

```
interface IInsert
```

```
{  
    public function users(Request $request);  
}
```

```
...
```

```
#### app\Http\Interfaces\Admin\IUpdate.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Interfaces\Admin;
```

```
use Illuminate\Http\Request;
```

```
interface IUpdate
```

```
{  
    public function users(Request $request);  
}
```

```
...
```

```
#### app\Http\Middleware\CheckIfAPILoggedIn.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use App\Http\Models\Users;
```

```
use Closure;
```

```
class CheckIfAPILoggedIn
```

```
{
```

```

/**
 * Handle an incoming request.
 *
 * @param \Illuminate\Http\Request $request
 * @param \Closure $next
 * @return mixed
 */
public function handle($request, Closure $next)
{
    //get auth token from cookie
    $authToken = $request->cookie('authToken');
    //
    // check if exist
    if(empty($authToken))
    {
        //user missing access token
        return response()->json(["error"=>["message"=>"Missing access token!"]],
403);
    }

    //
    //try to decode it
    try{
        $encrypter = app(\Illuminate\Contracts\Encryption\Encrypter::class);
        $token = $encrypter->decrypt($authToken);
    }
    catch (\Exception $e)
    {
        return response()->json(null, 500);
    }

    //
    // get user by token if exist
    $user = new Users();
    $userData = $user->getByAccessToken($token);

    if(empty($userData))
    {
        //user not authorized
        return response()->json(["error"=>["message"=>"User not authorized!"]],
401);
    }

    //
    // all good
    $request->attributes->add(["userInfo" => $userData]);
    return $next($request);
}

...

```

app\Http\Middleware\CheckIfLoggedIn.php

```

    ...

    <?php

namespace App\Http\Middleware;

use Closure;

class CheckIfLoggedIn
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if (!$request->session()->has('user'))
        {
            return back()->with('message', 'Please login to your account to access
this resource.');
```

```

#### app\Http\Middleware\CheckIfNotLoggedIn.php
```

```

    ...

    <?php

namespace App\Http\Middleware;

use Closure;

class CheckIfNotLoggedIn
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if ($request->session()->has('user'))
```



```

        {
            return back()->with('message', 'You are logged in.');
```

```
        }
        return $next($request);
    }
}
```

```

    ...

```

```

#### app\Http\Middleware\CheckRoleIfAdmin.php

```

```

    ...

```

```

    <?php

```

```

namespace App\Http\Middleware;

```

```

use App\Http\Models\Roles;

```

```

use Closure;

```

```

class CheckRoleIfAdmin

```

```

{

```

```

    /**

```

```

     * Handle an incoming request.

```

```

     *

```

```

     * @param \Illuminate\Http\Request $request

```

```

     * @param \Closure $next

```

```

     * @return mixed

```

```

     */

```

```

    public function handle($request, Closure $next)

```

```

    {

```

```

        $userRoles = $request->session()->get('roles');
```

```

        $canAccess = Roles::arrayOfRolesHasRoleByName($userRoles[0], 'admin');
```

```

        if($canAccess)

```

```

        {

```

```

            return $next($request);

```

```

        }

```

```

        return back()->with('message', 'You don\'t have role to access this
resource!');
    }
}

```

```

    ...

```

```
#### app\Http\Middleware\CheckRoleIfJamDeveloper.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use App\Http\Models\Roles;
```

```
use Closure;
```

```
class CheckRoleIfJamDeveloper
```

```
{  
    /**  
     * Handle an incoming request.  
     *  
     * @param \Illuminate\Http\Request $request  
     * @param \Closure $next  
     * @return mixed  
     */  
    public function handle($request, Closure $next)  
    {  
        $userRoles = $request->session()->get('roles');  
  
        $canAccess = Roles::arrayOfRolesHasRoleByName($userRoles[0], 'jamDeveloper');  
  
        if($canAccess)  
        {  
            return $next($request);  
        }  
  
        return back()->with('message', 'You don\'t have Jam Developer role to access  
this resource!');  
    }  
}
```

```
...
```

```
#### app\Http\Middleware\CheckRoleIfJamMaker.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use App\Http\Models\Roles;
```

```
use Closure;
```

```
class CheckRoleIfJamMaker
```

```
{  
    /**  
     * Handle an incoming request.
```

```

*
* @param \Illuminate\Http\Request $request
* @param \Closure $next
* @return mixed
*/
public function handle($request, Closure $next)
{
    $userRoles = $request->session()->get('roles');

    $canAccess = Roles::arrayOfRolesHasRoleByName($userRoles[0], 'jamMaker');

    if($canAccess)
    {
        return $next($request);
    }

    return back()->with('message', 'You don\'t have role to access this
resource!');
}
}

...

```

```

#### app\Http\Models\Comments.php

```

```

...

```

```

<?php

```

```

namespace App\Http\Models;

```

```

class Comment extends Generic

```

```

{
    public function __construct()
    {
        parent::__construct('comments', 'idComment');
    }
}

...

```

```

#### app\Http\Models\DownloadFiles.php

```

```

...

```

```

<?php

```

```

namespace App\Http\Models;

```

```

class DownloadFiles extends Generic
{
    public function __construct()
    {
        parent::__construct('downloadfiles', 'idDownloadFile');
    }
}

```

```

#### app\Http\Models\GameBadges.php

```

```

...

```

```

<?php

```

```

namespace App\Http\Models;

```

```

// idImage, idImageCategory, alt, path

```

```

class GameBadges extends Generic
{
    public function __construct()
    {
        parent::__construct('gamebadges', 'idGameBadges');
    }

    public function getByGameSubmissionId($gameId)
    {
        return \DB::table('gamesubmissions_badges')
            ->join('gamebadges', 'gamesubmissions_badges.idBadge', '=',
'gamebadges.idGameBadges')
            ->join('images', 'images.idImage', '=', 'gamebadges.idImage')
            ->where('gamesubmissions_badges.idGameSubmission', '=', $gameId)
            ->get();
    }

    public function getAllByGameId($idGame)
    {
        return \DB::table('gamesubmissions_badges')
            ->where('idGameSubmission', '=', $idGame)
            ->get();
    }
}

```

```

#### app\Http\Models\GameCategories.php

```

```

    ...

    <?php

namespace App\Http\Models;

class GameCategories extends Generic
{
    public function __construct()
    {
        parent::__construct('gamecategories', 'idGameCategory');
    }
}
    ...

```

```

#### app\Http\Models\GameCriteria.php

    ...

    <?php

namespace App\Http\Models;

// idGameCriteria, name, description
class GameCriteria extends Generic
{
    public function __construct()
    {
        parent::__construct('gamecriteria', 'idGameCriteria');
    }

    //
    // methods
    //

    public function insert($name, $description)
    {
        $insertData = [
            'name' => $name,
            'description' => $description
        ];
        return parent::insertGetId($insertData);
    }
}
    ...

```

```
#### app\Http\Models\GameJams.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Models;
```

```
// idGameJam title description idCoverImage startDate endDate votingEndDate content  
lockSubmissionAfterSubmitting  
// othersCanVote isBlocked idUserCreator numOfViews  
class GameJams extends Generic
```

```
{  
    public function __construct()  
    {  
        parent::__construct('gamejams', 'idGameJam');  
    }  
  
    //  
    // methods  
    //  
  
    public function insert($title, $description, $coverImage, $startDate, $endDate,  
$votingEndDate, $content, $lockSubmissionAfterSubmitting, $othersCanVote,  
$idUserCreator)  
    {  
        $insertData = [  
            'title' => $title,  
            'description' => $description,  
            'idCoverImage' => $coverImage,  
            'startDate' => $startDate,  
            'endDate' => $endDate,  
            'votingEndDate' => $votingEndDate,  
            'content' => $content,  
            'lockSubmissionAfterSubmitting' => $lockSubmissionAfterSubmitting,  
            'othersCanVote' => $othersCanVote,  
            'isBlocked' => 0,  
            'idUserCreator' => $idUserCreator,  
            'numOfViews' => 0,  
            'createdAt' => time(),  
        ];  
        return parent::insertGetId($insertData);  
    }  
  
    public function getAll() {  
        return \DB::table($this->tableName)  
            ->join('users', 'gamejams.idUserCreator', '=', 'users.idUser')  
            ->join('images', 'gamejams.idCoverImage', '=', 'images.idImage')  
            ->select("gamejams.*", "users.username", "images.path as cover")  
            ->get();  
    }  
  
    public function getAllWhereVotingEndDateNotFinished()  
    {  
        return \DB::table($this->tableName)  
            ->select(["idGameJam", "title", "startDate", "endDate", "votingEndDate"])  
            ->where('votingEndDate', '>', time())  
            ->get();  
    }  
}
```

```

}

public function getAllUsersGameJams($userId, $offset = 0, $limit = 6)
{
    $return = [];

    $result = \DB::table($this->tableName)
        ->join('users', 'gamejams.idUserCreator', '=', 'users.idUser')
        ->join('images', 'gamejams.idCoverImage', '=', 'images.idImage')
        ->where('gamejams.idUserCreator', '=', $userId)
        ->select("*", \DB::raw("(SELECT count(*) FROM gamejams_participants WHERE
gamejams_participants.idGameJam = gamejams.idGameJam) as countJoined"));

    $return["count"] = $result->count();

    $result->offset($offset)
        ->limit($limit);
    $return["result"] = $result->get();

    return $return;
}

public function getFilteredGameJams($filter, $offset = 0, $limit = 6)
{
    $return = [];

    $result = \DB::table($this->tableName)
        ->join('users', 'idUserCreator', '=', 'users.idUser')
        ->join('images', 'idCoverImage', '=', 'images.idImage')
        ->select("*", \DB::raw("(SELECT count(*) FROM gamejams_participants WHERE
gamejams_participants.idGameJam = gamejams.idGameJam) as countJoined, (SELECT
count(*) FROM gamesubmissions WHERE gamesubmissions.idGameJam = gamejams.idGameJam)
as countSubmissions"));

    // in progress
    if ($filter === "progress") {
        $count = $result->where("startDate", "<", time())
            ->where("endDate", ">", time())->count();

        $result->offset($offset)
            ->limit($limit)
            ->where("startDate", "<", time())
            ->where("endDate", ">", time());
    } // upcoming
    else if ($filter === "upcoming") {
        $count = $result->where("startDate", ">", time())->count();

        $result->offset($offset)
            ->limit($limit)
            ->where("startDate", ">", time());
    }

    $return["count"] = $count;
    $return["result"] = $result->get();

    return $return;
}

public function getOne($id)

```

```

{
    $gameJam = \DB::table($this->tableName)
        ->join('users', 'gamejams.idUserCreator', '=', 'users.idUser')
        ->join('images', 'gamejams.idCoverImage', '=', 'images.idImage')
        ->select('*')
        ->where('gamejams.idGameJam', '=', $id)
        ->first();

    $gameJam->{"participants"} = $this->getParticipants($id);
    $gameJam->{"criteria"} = $this->getCriteria($id);
    $gameJam->{"submissions"} = $this->getSubmissions($id);
    $gameJam->{"countSubmissions"} = $this->countByJoinedId("gamesubmissions",
$id);

    return $gameJam;
}

public function getSubmissions($id) {
    return \DB::table('gamesubmissions')
        ->join('images', 'gamesubmissions.idTeaserImage', '=', 'images.idImage')
        ->join('users', 'gamesubmissions.idUserCreator', '=', 'users.idUser')
        ->select('*')
        ->where('idGameJam', '=', $id)
        ->get();
}

public function getParticipants($id) {
    return \DB::table('gamejams_participants')
        ->join('users', 'gamejams_participants.idUser', '=', 'users.idUser')
        ->select('*')
        ->where('gamejams_participants.idGameJam', '=', $id)
        ->get();
}

public function getCriteria($id)
{
    return \DB::table('gamejams_criterias')
        ->join('gamecriteria', 'gamejams_criterias.idCriteria', '=',
'gamecriteria.idGameCriteria')
        ->select('*')
        ->where('gamejams_criterias.idGameJam', '=', $id)
        ->get();
}

public function insertCriteria($idGameJam, $idCriteria)
{
    return \DB::table('gamejams_criterias')
        ->insert([
            'idGameJam' => $idGameJam,
            'idCriteria' => $idCriteria,
        ]);
}

public function deleteCriteria($idGameJam, $idGameCriteria){
    return \DB::table('gamejams_criterias')
        ->where("idGameJam", "=", $idGameJam)
        ->where("idCriteria", "=", $idGameCriteria)
        ->delete();
}

```



```

public function increaseViews($id)
{
    \DB::statement("UPDATE gamejams SET numOfViews = numOfViews + 1 WHERE
idGameJam = " . $id);
}

public function joinUserToGameJam($idUser, $idGameJam) {
    return \DB::table('gamejams_participants')
        ->insertGetId(["idUser" => $idUser, "idGameJam" => $idGameJam]);
}

public function removeUserFromGameJam($idUser, $idGameJam) {
    return \DB::table('gamejams_participants')
        ->where('idUser', '=', $idUser)
        ->where('idGameJam', '=', $idGameJam)
        ->delete();
}

public function userAlreadyJoined($idUser, $idGameJam) {
    return \DB::table('gamejams_participants')
        ->select('*')
        ->where('idUser', '=', $idUser)
        ->where('idGameJam', '=', $idGameJam)
        ->exists();
}

public function userOwnsGameJam($idUser, $idGameJam) {
    return \DB::table('gamejams')
        ->select('*')
        ->where('idUserCreator', '=', $idUser)
        ->where('idGameJam', '=', $idGameJam)
        ->exists();
}

public function getAllSearched($queryString, $offset = 0, $limit = 6) {
    return \DB::table($this->tableName)
        ->join('images', 'gamejams.idCoverImage', '=', 'images.idImage')
        ->select(["images.alt", "images.path", "gamejams.title",
"gamejams.description", "gamejams.idGameJam"])
        ->where('gamejams.title', 'like', '%' . $queryString . '%')
        ->orWhere('gamejams.description', 'like', '%' . $queryString . '%')
        ->offset($offset)
        ->limit($limit)
        ->get();
}

public function countAllSearched($queryString)
{
    return \DB::table($this->tableName)
        ->where('gamejams.title', 'like', '%' . $queryString . '%')
        ->orWhere('gamejams.description', 'like', '%' . $queryString . '%')
        ->count();
}
}
...

```

```
#### app\Http\Models\GameSubmissions.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Models;
```

```
// idGameSubmission, idGameJam, idTeaserImage, idCoverImage, idUserCreator,  
description, createdAt, editedAt, numOfViews, numOfDownloads, isBlocked,  
numberOfVotes, sumOfVotes
```

```
class GameSubmissions extends Generic
```

```
{  
    public function __construct()  
    {  
        parent::__construct('gamesubmissions', 'idGameSubmission');  
    }  
  
    //  
    // methods  
    //  
  
    public function insert()  
    {  
  
    }  
  
    public function getAll() {  
        return \DB::table($this->tableName)  
            ->join('users', 'gamesubmissions.idUserCreator', '=', 'users.idUser')  
            ->join('gamejams', 'gamesubmissions.idGameJam', '=',  
'gamejams.idGameJam')  
            ->join('images', 'gamesubmissions.idCoverImage', '=', 'images.idImage')  
            ->select(\DB::raw("gamesubmissions.*, gamejams.title as gameJam,  
(gamesubmissions.sumOfVotes / gamesubmissions.numberOfVotes) as rating,  
users.username, images.path as cover"))  
            ->get();  
    }  
  
    public function get($offset = 0, $limit = 9, $sort)  
    {  
        $sort["name"] = "gamesubmissions." . $sort["name"];  
  
        $games = \DB::table($this->tableName)  
            ->join('users', 'gamesubmissions.idUserCreator', '=', 'users.idUser')  
            ->join('images', 'gamesubmissions.idTeaserImage', '=', 'images.idImage')  
            ->select(\DB::raw('*', (gamesubmissions.sumOfVotes /  
gamesubmissions.numberOfVotes) as rating'))  
            ->offset($offset)  
            ->limit(9)  
            ->orderBy($sort["name"] === "gamesubmissions.rating" ? "rating" :  
$sort["name"], $sort["direction"])  
            ->get();  
  
        foreach ($games as $game) {
```

```

        $game->{"categories"} = $this->getCategories($game->idGameSubmission);
    }

    return $games;
}

public function getCategories($id)
{
    return \DB::table('gamesubmissions_categories')
        ->join('gamecategories', 'gamesubmissions_categories.idCategory', '=',
'gamecategories.idGameCategory')
        ->select('*')
        ->where('gamesubmissions_categories.idGameSubmission', '=', $id)
        ->get();
}

public function getCategoriesIds($id)
{
    $categories = $this->getCategories($id);
    $result = [];
    foreach($categories as $c)
    {
        $result[] = $c->idCategory;
    }

    return $result;
}

public function getByUserAndGameJam($idUser, $idGameJam)
{
    return \DB::table($this->tableName)
        ->select('*')
        ->where('idUserCreator', '=', $idUser)
        ->where('idGameJam', '=', $idGameJam)
        ->first();
}

public function getAllSearched($queryString, $offset = 0, $limit = 6)
{
    return \DB::table($this->tableName)
        ->join('images', 'gamesubmissions.idCoverImage', '=', 'images.idImage')
        ->select(["images.alt", "images.path", "gamesubmissions.title",
"gamesubmissions.description", "gamesubmissions.idGameSubmission"])
        ->where('gamesubmissions.title', 'like', '%' . $queryString . '%')
        ->orWhere('gamesubmissions.description', 'like', '%' . $queryString .
'%')
        ->offset($offset)
        ->limit($limit)
        ->get();
}

public function countAllSearched($queryString)
{
    return \DB::table($this->tableName)
        ->where('gamesubmissions.title', 'like', '%' . $queryString . '%')
        ->orWhere('gamesubmissions.description', 'like', '%' . $queryString .
'%')
        ->count();
}

```

```

public function increaseViews($id)
{
    return \DB::table($this->tableName)
        ->where('idGameSubmission', '=', $id)
        ->increment('numOfViews', 1);
}

public function getOne($id)
{
    return \DB::table($this->tableName)
        ->join('users', 'gamesubmissions.idUserCreator', '=', 'users.idUser')
        ->join('images', 'gamesubmissions.idCoverImage', '=', 'images.idImage')
        ->where('idGameSubmission', '=', $id)
        ->first();
}

public function getScreenShots($id)
{
    return \DB::table('gamesubmissions_screenshots')
        ->join('images', 'gamesubmissions_screenshots.idImage', '=',
'images.idImage')
        ->where('gamesubmissions_screenshots.idGameSubmission', '=', $id)
        ->get();
}

public function getDownloadFiles($id)
{
    return \DB::table('gamesubmissions_downloadfiles')
        ->select(['gamesubmissions_downloadfiles.*', 'downloadfiles.*',
'platforms.classNameForIcon'])
        ->join('downloadfiles', 'gamesubmissions_downloadfiles.idDownloadFile',
'=', 'downloadfiles.idDownloadFile')
        ->join('platforms', 'downloadfiles.idPlatform', '=',
'platforms.idPlatform')
        ->where('gamesubmissions_downloadfiles.idGameSubmission', '=', $id)
        ->get();
}

public function getGamePlatformId($id)
{
    return \DB::table('gamesubmissions_downloadfiles')
        ->select('downloadfiles.idPlatform')
        ->join('downloadfiles', 'gamesubmissions_downloadfiles.idDownloadFile',
'=', 'downloadfiles.idDownloadFile')
        ->where('gamesubmissions_downloadfiles.idGameSubmission', '=', $id)
        ->first();
}

public function saveRelationshipWithDownloadFile($idGameSubmission, $idFile)
{
    return \DB::table('gamesubmissions_downloadfiles')
        ->insert([
            "idGameSubmission" => $idGameSubmission,
            "idDownloadFile" => $idFile,
        ]);
}

public function removeRelationshipWithDownloadFile($idGame)

```

```

{
    return \DB::table('gamesubmissions_downloadfiles')
        ->where('idGameSubmission', '=', $idGame)
        ->delete();
}

public function addBadge($idGame, $idBadge, $idUser)
{
    //insert and return value
    $id = \DB::table('gamesubmissions_badges')
        ->insertGetId([
            "idGameSubmission" => $idGame,
            "idBadge" => $idBadge,
            "idUser" => $idUser,
        ]);

    return \DB::table('gamesubmissions_badges')
        ->join('gamebadges', 'gamesubmissions_badges.idBadge', '=',
'gamebadges.idGameBadges')
        ->join('images', 'images.idImage', '=', 'gamebadges.idImage')
        ->where('gamesubmissions_badges.idGameSubmissionsBadge', '=', $id)
        ->first();
}

public function getOneGameSubmissionBadgeById($id)
{
    return \DB::table('gamesubmissions_badges')
        ->where('idGameSubmissionsBadge', '=', $id)
        ->first();
}

public function removeBadge($idBadge)
{
    return \DB::table('gamesubmissions_badges')
        ->where('idGameSubmissionsBadge', '=', $idBadge)
        ->delete();
}

public function getAllUsersGameSubmissions($userId, $offset = 0, $limit = 6)
{
    $return = [];

    $games = \DB::table($this->tableName)
        ->join('users', 'gamesubmissions.idUserCreator', '=', 'users.idUser')
        ->join('images', 'gamesubmissions.idTeaserImage', '=', 'images.idImage')
        ->select(\DB::raw('* , (gamesubmissions.sumOfVotes /
gamesubmissions.numberOfVotes) as rating'))
        ->where("gamesubmissions.idUserCreator", "=", $userId);

    $return["count"] = $games->count();

    $games->offset($offset)
        ->limit($limit);
    $return["result"] = $games->get();

    foreach ($return["result"] as $game) {
        $game->{"categories"} = $this->getCategories($game->idGameSubmission);
    }
}

```

```

        return $return;
    }

    public function getAllUsersGameSubmissionsWins($userId, $offset = 0, $limit = 6)
    {
        $return = [];

        $games = \DB::table($this->tableName)
            ->join('users', 'gamesubmissions.idUserCreator', '=', 'users.idUser')
            ->join('images', 'gamesubmissions.idTeaserImage', '=', 'images.idImage')
            ->select(\DB::raw('*', (gamesubmissions.sumOfVotes /
gamesubmissions.numberOfVotes) as rating'))
            ->where("gamesubmissions.idUserCreator", "=", $userId)
            ->where("gamesubmissions.isWinner", '=', '1');

        $return["count"] = $games->count();

        $games->offset($offset)
            ->limit($limit);
        $return["result"] = $games->get();

        foreach ($return["result"] as $game) {
            $game->{"categories"} = $this->getCategories($game->idGameSubmission);
        }

        return $return;
    }

    public function saveGameCategories($model)
    {
        return \DB::table('gamesubmissions_categories')
            ->insertGetId($model);
    }

    public function removeAllCategories($idGame){
        return \DB::table('gamesubmissions_categories')
            ->where('idGameSubmission', '=', $idGame)
            ->delete();
    }

    public function userOwnsGameSubmission($idUser, $idGameSubmission)
    {
        //todo
    }

    public function gameIdByDownloadFileId($id)
    {
        return \DB::table('gamesubmissions_downloadfiles')
            ->select('*')
            ->where('idDownloadFile', '=', $id)
            ->first();
    }

    public function getDownloadFileIdByGameId($id)
    {
        return \DB::table('gamesubmissions_downloadfiles')
            ->select('*')

```

```

        ->where('idGameSubmission', '=', $id)
        ->first();
    }
}
...

```

app\Http\Models\GameSubmission_Comment.php

...

<?php

namespace App\Http\Models;

class GameSubmission_Comment extends Generic

```

{
    public function __construct()
    {
        parent::__construct('gamesubmissions_comments', 'idGameSubmissionComment');
    }

```

public function getByGameSubmissionId(\$gameId)

```

{
    return \DB::table($this->tableName)
        ->select([
            'gamesubmissions_comments.*',
            'comments.*',
            'users.idUser',
            'users.username',
            'users.avatarImagePath',
        ])
        ->join('comments', 'comments.idComment', '=',
'gamesubmissions_comments.idComment')
        ->join('users', 'users.idUser', '=', 'comments.idUserCreator')
        ->where('gamesubmissions_comments.idGameSubmission', '=', $gameId)
        ->get();
}

```

public function getOneById(\$commentId)

```

{
    return \DB::table($this->tableName)
        ->select([
            'gamesubmissions_comments.*',
            'comments.*',
            'users.idUser',
            'users.username',
            'users.avatarImagePath',
        ])
        ->join('comments', 'comments.idComment', '=',
'gamesubmissions_comments.idComment')
        ->join('users', 'users.idUser', '=', 'comments.idUserCreator')
        ->where('gamesubmissions_comments.idGameSubmissionComment', '=',
$commentId)

```

```

        ->first();
    }
}
...

```

```
#### app\Http\Models\Generic.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Models;
```

```
use Illuminate\Support\Facades\DB;
```

```
class Generic
```

```

{
    protected $tableName;
    private $idName;

    public function __construct($tableName = null, $idName = null)
    {
        $this->tableName = $tableName;
        $this->idName = !empty($idName) ? $idName : $this->getTableIdColumnName();
    }

    public function getAll()
    {
        return DB::table($this->tableName)
            ->select('*')
            ->get();
    }

    public function getById($id)
    {
        return DB::table($this->tableName)
            ->select('*')
            ->where($this->idName, '=', $id)
            ->first();
    }

    public function insertGetId($model)
    {
        return DB::table($this->tableName)
            ->insertGetId($model);
    }

    public function insertGetRow($model)
    {
        $id = $this->insertGetId($model);
        return $this->getById($id);
    }
}

```



```

public function update($id, $data)
{
    return DB::table($this->tableName)
        ->where($this->idName, '=', $id)
        ->update($data);
}

public function delete($id)
{
    return DB::table($this->tableName)
        ->where($this->idName, '=', $id)
        ->delete();
}

public function deleteMultiple($ids) {
    return DB::table($this->tableName)
        ->whereIn($this->idName, $ids)
        ->delete();
}

public function count()
{
    return DB::table($this->tableName)
        ->count();
}

public function countByJoinedId($tableName, $id)
{
    return \DB::table($tableName)
        ->where($this->idName, '=', $id)
        ->count();
}

public function exist($id)
{
    return \DB::table($this->tableName)
        ->where($this->idName, $id)
        ->exists();
}

public function getTableColumnNames()
{
    return DB::getSchemaBuilder()->getColumnListing($this->tableName);
}

public function getTableIdColumnName()
{
    return DB::getSchemaBuilder()->getColumnListing($this->tableName)[0];
}

public function increment($id, $columnName)
{
    return \DB::table($this->tableName)
        ->where($this->idName, '=', $id)
        ->increment($columnName, 1);
}
}
...

```

```
#### app\Http\Models\ImageCategories.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Models;
```

```
class ImageCategories extends Generic
```

```
{
```

```
    public function __construct()
```

```
    {
```

```
        parent::__construct('imagecategories', 'idImageCategory');
```

```
    }
```

```
}
```

```
...
```

```
#### app\Http\Models\Images.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Models;
```

```
// idImage, idImageCategory, alt, path
```

```
use App\Http\Enums\ImageCategories;
```

```
use Illuminate\Support\Facades\File;
```

```
class Images extends Generic
```

```
{
```

```
    public function __construct()
```

```
    {
```

```
        parent::__construct('images', 'idImage');
```

```
    }
```

```
//
```

```
// methods
```

```
//
```

```
    public function insert($category, $alt, $path)
```

```
    {
```

```
        $insertData = [
```

```
            'idImageCategory' => $category,
```

```
            'alt' => $alt,
```

```
            'path' => $path,
```

```

        'createdAt' => time(),
    ];

    return parent::insertGetId($insertData);
}

public function insertScreenShots($idGameSubmission, $idScreenShot)
{
    return \DB::table('gamesubmissions_screenshots')
        ->insert([
            "idGameSubmission"=>$idGameSubmission,
            "idImage"=> $idScreenShot,
        ]);
}

public function removeScreenShots($idGame)
{
    return \DB::table('gamesubmissions_screenshots')
        ->where('idGameSubmission', '=', $idGame)
        ->delete();
}

public function saveFileInFolder($category, $file)
{
    $extension = $file->getClientOriginalExtension();
    $tmp_path = $file->getPathName();

    $folder = ImageCategories::getSavingFolderByEnum($category);
    $file_name = time() . rand(1, 10) . "." . $extension;
    $new_path = public_path($folder) . $file_name;

    try {
        File::move($tmp_path, $new_path);
        return $folder.$file_name;
    } catch (\ErrorException $ex) {
        return null;
    }
}

public function saveImageAndGetId($category, $alt, $photo)
{
    $filePath = $this->saveFileInFolder($category, $photo);
    if($filePath == null){
        return null;
    }

    return $this->insert($category, $alt, $filePath);
}
}

...

```

app\Http\Models\Navigations.php

```

    ...

    <?php

namespace App\Http\Models;

class Navigations extends Generic
{
    public function __construct()
    {
        parent::__construct('navigations', 'idNavigation');
    }

    public function getAllSortedByPosition()
    {
        return \DB::table('navigations')
            ->select('*')
            ->orderBy('position','asc')
            ->get();
    }
}
    ...

```

```

#### app\Http\Models\Platform.php

```

```

    ...

    <?php

namespace App\Http\Models;

class Platform extends Generic
{
    public function __construct()
    {
        parent::__construct('platforms', 'idPlatform');
    }
}
    ...

```

```

#### app\Http\Models\Platforms.php

```

```

    ...

```

```

        <?php

namespace App\Http\Models;

class Platforms extends Generic
{
    public function __construct()
    {
        parent::__construct('platforms', 'idPlatform');
    }
}
    ...

```

```

#### app\Http\Models\PollAnswers.php
    ...

```

```

        <?php

namespace App\Http\Models;

use Illuminate\Support\Facades\DB;

class PollAnswers extends Polls {

    public function __construct() {
        parent::__construct('pollanswers', 'idPollAnswer');
    }

}
    ...

```

```

#### app\Http\Models\PollQuestions.php
    ...

```

```

        <?php

namespace App\Http\Models;

use Illuminate\Support\Facades\DB;

class PollQuestions extends Polls {

    public function __construct() {
        parent::__construct('pollquestions', 'idPollQuestion');
    }
}

```

```
    }  
}  
    ...
```

```
#### app\Http\Models\Polls.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Models;
```

```
//  
// this class is responsible for tables:  
// [pollQuestions, pollAnswers, pollVotes]  
//
```

```
use Illuminate\Support\Facades\DB;
```

```
class Polls extends Generic
```

```
{  
    public function __construct($tableName = null, $idName = null) {  
        parent::__construct($tableName === null ? "pollquestions" : $tableName,  
$idName === null ? "idPollQuestion" : $idName);  
    }
```

```
    public function getAll() {  
        $questions = DB::table('pollquestions')  
            ->select('*')  
            ->get();
```

```
        foreach($questions as $question) {  
            $question->{"answers"} = $this->getAnswersByQuestionId($question->  
idPollQuestion);  
        }
```

```
        return $questions;  
    }
```

```
    public function getAnswersByQuestionId($id) {  
        return DB::table('pollanswers')  
            ->select('*')  
            ->where('idPollQuestion', '=', $id)  
            ->get();  
    }
```

```
    public function getByIdAnswer($id) {  
        return DB::table('pollanswers')  
            ->select('*')  
            ->where("idPollAnswer", $id)  
            ->first();  
    }
```

```

public function getActivePollQuestion()
{
    return DB::table('pollquestions')
        ->select('*')
        ->where('active', '=', 0)
        ->first();
}

public function setActivePollQuestion($id) {
    DB::table('pollquestions')
        ->update(['active' => 0]);

    return DB::table('pollquestions')
        ->where('idPollQuestion', $id)
        ->update(['active' => 1]);
}

public function pollVote($userId, $idPollQuestion, $idPollAnswer)
{
    try {
        DB::table('pollvotes')
            ->insert([
                'idUserVoter' => $userId,
                'idPollQuestion' => $idPollQuestion,
                'idPollAnswer' => $idPollAnswer,
                'createdAt' => time(),
            ]);

        DB::table('pollAnswers')
            ->where('idPollAnswer', '=', $idPollAnswer)
            ->increment('numberOfVotes', 1);

        return true;
    }
    catch (\Illuminate\Database\QueryException $e)
    {
        //duplicate entry [ user already voted]
        return false;
    }

    return false;
}
}
...

```

app\Http\Models\Reports.php

...

<?php

```

namespace App\Http\Models;

class Reports extends Generic
{
    public function __construct()
    {
        parent::__construct('reports', 'idReport');
    }

    public function getAll() {
        return \DB::table($this->tableName)
            ->join("users", "reports.idUserCreator", "=", "users.idUser")
            ->select("*", "users.username")
            ->where("solved", 0)
            ->orderBy("idReport")
            ->get();
    }

    public function userHasReportedGame($idGame, $idUser){
        return \DB::table($this->tableName)
            ->where('idUserCreator', '=', $idUser)
            ->where('idReportObject', '=', $idGame)
            ->exists();
    }
}

```

```

#### app\Http\Models\Roles.php

```

```

    ...

```

```

    <?php

```

```

/**
 * Created by PhpStorm.
 * User: Urukalo
 * Date: 5/26/2018
 * Time: 4:29 PM
 */

```

```

namespace App\Http\Models;

class Roles extends Generic
{
    public function __construct()
    {
        parent::__construct('roles', 'idRole');
    }

    public function getAllAvailable()
    {
        return \DB::table('roles')
            ->select('*')
    }
}

```



```

        ->where('isAvailableForUser','=', 1) //bool => true
        ->get();
    }

    public static function arrayOfRolesHasRoleByName($roles, $roleName)
    {
        if(!empty($roles))
        {
            foreach($roles as $role)
            {
                if($role->name == $roleName)
                {
                    return true;
                }
            }
        }

        return false;
    }
}
...

```

```
#### app\Http\Models\Statistics.php
```

```
...
```

```
<?php
```

```
namespace App\Http\Models;
```

```
class Statistics
```

```

{
    private $limit = 7;

    public function getForChart($tableName, $tableNameTime = 'createdAt'){
        return \DB::table($tableName)
            ->select(\DB::raw('FROM_UNIXTIME(`'.$tableNameTime.'`, \'%Y-%m-%d\') as
ndate, count(*) as data_count'))
            ->groupBy('ndate')
            ->limit($this->limit)
            ->get();
    }

    public function getTableCount($tableName){
        return \DB::table($tableName)
            ->count();
    }
}
...

```

```

#### app\Http\Models\Users.php
...

<?php

namespace App\Http\Models;

class Users extends Generic
{
    public function __construct()
    {
        parent::__construct('users', 'idUser');
    }

    //
    // methods
    //

    public function insert($email, $username, $password, $isBanned = 0, $avatar =
null)
    {
        $timeCreatedAt = time();

        $avatar = empty($avatar) ? 'https://api.adorable.io/avatars/285/' . $email :
$avatar;

        $insertData = [
            'email' => $email,
            'username' => $username,
            'password' => md5($password),
            'createdAt' => $timeCreatedAt,
            'updatedAt' => $timeCreatedAt,
            'avatarImagePath' => $avatar,
            'isBanned' => $isBanned,
        ];
        return parent::insertGetId($insertData);
    }

    public function updateUser($idUser, $updateData){
        $updateData['updatedAt'] = time();

        return parent::update($idUser, $updateData);
    }

    public function getByUsernameOrEmailAndPassword($usernameEmail, $password)
    {
        $result = \DB::table($this->tableName)
            ->select('*')
            ->where('password', '=', md5($password))
            ->where(function ($query) use ($usernameEmail) {
                $query->where('username', '=', strtolower($usernameEmail))
                    ->orWhere('email', '=', strtolower($usernameEmail));
            })
            ->first();
    }
}

```

```

        return $result;
    }

    public function getAll() {
        $users = \DB::table($this->tableName)
            ->select("*")
            ->get();

        foreach ($users as $user) {
            $user->{"roles"} = $this->getAllRoles($user->idUser);
        }

        return $users;
    }

    public function getById($id) {
        $user = \DB::table($this->tableName)
            ->select("*")
            ->where("idUser", "=", $id)
            ->first();

        $user->{"roles"} = $this->getAllRoles($id);

        return $user;
    }

    public function getByUsername($username)
    {
        return \DB::table($this->tableName)
            ->select('*')
            ->where('username', '=', $username)
            ->first();
    }

    public function getIdByUsername($username)
    {
        return \DB::table($this->tableName)
            ->select('idUser')
            ->where('username', '=', $username)
            ->first();
    }

    public function addRole($idRole, $idUser)
    {
        return \DB::table('users_roles')
            ->insert([
                'idUser' => $idUser,
                'idRole' => $idRole,
            ]);
    }

    public function deleteRole($userId, $roleId){
        return \DB::table('users_roles')
            ->where("idUser", "=", $userId)
            ->where("idRole", "=", $roleId)
            ->delete();
    }

    public function getAllRoles($userId)

```

```

    {
        return \DB::table('users_roles')
            ->join('roles', 'users_roles.idRole', '=', 'roles.idRole')
            ->select('*')
            ->where('users_roles.idUser', '=', $userId)
            ->get();
    }

    public function getByAccessToken($token)
    {
        return \DB::table($this->tableName)
            ->select('*')
            ->where('accessToken', '=', $token)
            ->first();
    }

    public function updateAccessToken($idUser, $token)
    {
        return \DB::table($this->tableName)
            ->where('idUser', '=', $idUser)
            ->update([
                'accessToken'=> $token
            ]);
    }

    public function removeAccessToken($idUser)
    {
        return \DB::table($this->tableName)
            ->where('idUser', '=', $idUser)
            ->update([
                'accessToken'=> null
            ]);
    }
}

```

app\Http\Models\Utilities.php

...

<?php

namespace App\Http\Models;

class Utilities

```

{
    private static function DateTimeFormater($format, $time = null)
    {
        if ($time == null) $time = time();
        return date($format, $time);
    }
}

```

```

public static function PrintDate($time = null)
{
    return self::DateTimeFormatter("Y-m-d", $time);
}

public static function PrintDateTime($time = null)
{
    return self::DateTimeFormatter("Y-m-d h:i:s", $time);
}

public static function FormatBytes($size, $precision = 2)
{
    $base = log($size, 1024);
    $suffixes = array('', 'KB', 'MB', 'GB', 'TB');

    return round(pow(1024, $base - floor($base)), $precision) . ' ' .
    $suffixes[floor($base)];
}
}

```

2. JavaScript

```

#### public\js\admin.js

...

$(document).ready(function () {
$(".modal-close, #modal-confirm-no").on("click", function () {
    $(".modal-box").animate({
        opacity: 0
    }, 150, function () {
        $(".modal-box").css("display", "none");
    });
});

$(".modal-box").on("click", function (e) {
    if ($(e.target).attr("class") !== "modal-box modal-update-box") return;

    $(".modal-box").animate({
        opacity: 0
    }, 150, function () {
        $(".modal-box").css("display", "none");
    });
});

$("#hide-form-errors").on("click", function () {
    $(this).parent().hide();
});

$(".menu-open span").on("click", function () {
    $(this).toggleClass("click");
    if $(".menu").css("opacity") == "1" {
        $(".menu-container").animate({"left": -237}, 300, function () {
            $(".menu").css("opacity", "0");
        });
    }
});

```

```

    });
    $(".menu-open").css("width", "50px");
    $(".menu-open span").css("left", "10px");
    $(".main-content").animate({"margin-left": "50px"}, 450);
  }
  else {
    $(".menu").css("opacity", "1");
    $(".menu-container").animate({"left": 0}, 400);
    $(".menu-open").css("width", "36px");
    $(".menu-open span").css("left", "7px");
    $(".main-content").animate({"margin-left": "283px"}, 400);
  }
});

// delete
$("#content").on("click", ".main-table .data-delete a", function (e) {
  e.preventDefault();

  var resetElements = [];

  if($(this).attr("data-poll-type") === "question") {
    var pollQuestionId = $(this).attr("data-id");
    $(".inner-table-wrap:visible").each(function () {
      if($(this).attr("data-poll-question-id") !== pollQuestionId) {
        resetElements.push([".inner-table-wrap", $(".inner-table-
wrap").index($(this)), $(this).attr("data-poll-question-id")]);
      }
    });
    deleteData(pollQuestionId, "pollquestions", resetElements);
  }
  else if($(this).attr("data-poll-type") === "answer") {
    $(".inner-table-wrap:visible").each(function () {
      resetElements.push([".inner-table-wrap", $(".inner-table-
wrap").index($(this)), $(this).attr("data-poll-question-id")]);
    });
    deleteData($(this).attr("data-id"), "pollanswers", resetElements);
  }
  else{
    deleteData($(this).attr("data-id"));
  }

  return false;
});

// delete selected
$("#deleteSelected").on("click", function (e) {
  e.preventDefault();

  var ids = [];

  $(".input.chb-select-row").filter(function (i) {
    this.checked ? ids.push($(this).attr("data-id")) : null;
  });

  if(ids.length > 0){
    deleteData(ids);
  }

  return false;
});

```

```

});

// select all checkboxes
$("#content").on("change", ".main-table #chbSelectAll", function () {
    $("input.chb-select-row").prop("checked", this.checked);
});

var deleteData = function (ids, table = null, resetElements = null) {
    var _ids = Array.isArray(ids) ? ids : [ids];

    var url = delete_url;
    var tableName = table === null ? base_table_name : table;
    var csrfToken = $('meta[name="csrf-token"]').attr('content');
    var viewName = base_view_name;

    $.ajax({
        url: url,
        method: "DELETE",
        data: {
            _token: csrfToken,
            viewName: viewName,
            ids: _ids,
            tableName: tableName
        },
        beforeSend: function(data) {
            $("#loading-overlay").css("display", "block");
        },
        success: function (data) {
            $("#content").html(data);

            if(resetElements !== null) {
                for(var i = 0; i < resetElements.length; i++){
                    var $element = $(resetElements[i][0] + ":eq(" +
resetElements[i][1] + ")");
                    $element.show();
                    $(".table-poll-question-row[data-id='" + resetElements[i][2]
+ "']").find(".expand-poll-question").addClass("click");
                }
            }

        },
        complete: function() {
            $("#loading-overlay").css("display", "none");
        },
        error: function (error) {
            console.log(error.message);
        }
    });
};

});

slamjam.dashboard = (function(){

    function formatDate(date) {
        var dd = date.getDate();
        var mm = date.getMonth() + 1;
        var yyyy = date.getFullYear();
        if (dd < 10) {
            dd = '0' + dd
        }
    }

```

```

        if (mm < 10) {
            mm = '0' + mm
        }
        date = yyyy + '-' + mm + '-' + dd;
        return date
    }

function last7Days() {
    var result = [];
    for (var i = 0; i < 7; i++) {
        var d = new Date();
        d.setDate(d.getDate() - i);
        result.push({
            date: formatDate(d),
            value: 0,
        })
    }

    return result.reverse();
}

function combineRealWith7Days(realData) {
    var aLast7Days = last7Days();
    // console.log(aLast7Days);
    // console.log(realData);
    // console.log("");

    return aLast7Days.map(function (item) {
        var index = realData.findIndex(function(row){ return item.date ===
row.ndate; });
        if(index > -1){
            item.value += realData[index].data_count;
        }

        return item.value;
    });
}

function makeChart(selector, realData) {
    $(selector).sparkline(realData, {
        type: 'line',
        height: '20',
        width: '100%',
        lineColor: '#8EB8E5',
        fillColor: '#292c35',
        spotColor: '#DB504A',
    });

    /*var myvalues = [0, 0, 0, 2, 4, 12, 4, 8];
    $('.dynamicsparkline').each(function () {
        var $this = $(this);
        $this.sparkline(myvalues, {
            type: 'line',
            height: '20',
            width: '100%',
            lineColor: '#8EB8E5',
            fillColor: '#292c35',

```



```

        spotColor: '#DB504A',
    });
});*/
}

```

```

function initChartView(chartData, metadata) {
    chartData.map(function (item) {
        var statPropName = Object.keys(item)[0];
        var chartMeta = metadata[statPropName];

        if(chartMeta){
            makeChart(
                "#" + chartMeta["chartSelector"], //this is selector: #users
                combineRealWith7Days(item[statPropName])// this is array data:
[0, 2, 3,]
            );
        }
    });
}

```

```

function initCountView(countData, metadata) {
    countData.map(function (item) {
        var statPropName = Object.keys(item)[0];
        var countMeta = metadata[statPropName];

        if(countMeta){
            $("#" + countMeta["countSelector"]).text(item[statPropName]);
        }
    });
}

```

```

function initDashboard() {
    var statisticMetaData = {
        "users": { countSelector: "stats_count_users", chartSelector:
"stats_chart_users"},
        "gameJams": { countSelector: "stats_count_gamejams", chartSelector:
"stats_chart_gamejams"},
        "games": { countSelector: "stats_count_games", chartSelector:
"stats_chart_games"},
        "reports": { countSelector: "stats_count_reports", chartSelector:
"stats_chart_reports"},
        "comments": { countSelector: "stats_count_comments", chartSelector:
"stats_chart_comments"},
        "downloadFiles": { countSelector: "stats_count_downloadfiles",
chartSelector: "stats_chart_downloadfiles"},
        "images": { countSelector: "stats_count_images", chartSelector:
"stats_chart_images"},
        "polls": { countSelector: "stats_count_polls", chartSelector:
"stats_chart_polls"},
    };

    $.ajax({
        url: base_url_api + '/admin/statistics/chart/all',
        dataType: 'json',
    });
}

```

```

        success: function (data) {
            initChartView(data, statisticMetaData);
        },
        error: function (err) {
            console.log(err);
        }
    });

$.ajax({
    url: base_url_api + '/admin/statistics/count/all',
    dataType: 'json',
    success: function (data) {
        initCountView(data, statisticMetaData);
    },
    error: function (err) {
        console.log(err);
    }
});

}

return {
    initDashboard: initDashboard,
}
})();
...

```

```

#### public\js\main.js

```

```

...

/*
 * All about game jams
 **/
slamjam.common = (function () {

    function _createUrl(url, noApi) {
        if (noApi) return base_url + url;
        return base_url_api + url;
    }

    function _ajax(options) {
        //
        // add default options for ajax
        var defaultOptions = {
            data: null,
            dataType: 'json',
            url: base_url,
        };

        //
        // ajax call
        $.ajax(Object.assign({}, defaultOptions, options, {
            //

```

```

        // default function that can handle things before or after
        //
        beforeSend: function () {
            //
            // start loader
            _startLoader();
        },
        success: function (data, textStatus, jqXHR) {
            if (typeof options.success === 'function') {
                options.success(data, textStatus, jqXHR);
            }
        },
        error: function (jqXHR, textStatus, errorThrown) {
            if (typeof options.error === 'function') {
                options.error(jqXHR, textStatus, errorThrown);
            }
        },
        complete: function () {
            _stopLoader();

            if (typeof options.complete === 'function') {
                options.complete();
            }
        }
    }
    }));
}

/*
 *   This is Loading handler
 * */

var loaderPosition = 0;
var $loader = null;

function _startLoader() {
    if ($loader === null) {
        //init loader selector
        $loader = $("#loading-overlay");
    }

    loaderPosition++;
    if ($loader) {
        $loader.css('display', 'block');
    }
}

function _stopLoader() {
    --loaderPosition;

    if (loaderPosition <= 0) {
        loaderPosition = 0;
        if ($loader !== null) {
            $loader.css('display', 'none');
        }
    }
}

function _confirmBox($element) {

```

```

    $element.on("click", function () {
        $("#modal-confirm-yes").attr("href", $element.attr("data-url"));
        $(".modal-info").css("display", "block");
        $(".modal-confirm").css({"display": "block"});
        $(".modal-info").animate({
            opacity: 1
        }, 150);
        $(".modal-info-text").html($element.attr("data-text"));
        return false;
    });
}

function _getDateComponents(date) {
    const days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"];
    const months = ["January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"];

    var dayName = days[date.getDay()];
    var monthName = months[date.getMonth()];

    var hours = date.getHours();
    var minutes = date.getMinutes();
    var ampm = hours >= 12 ? 'PM' : 'AM';
    hours = hours % 12;
    hours = hours ? hours : 12;
    hours = hours < 10 ? '0' + hours : hours;
    minutes = minutes < 10 ? '0' + minutes : minutes;
    var strTime = hours + ':' + minutes + ' ' + ampm;

    return {
        time: strTime,
        day: date.getDate(),
        dayName: dayName,
        monthName: monthName
    };
}

return {
    //return what others need to use from common
    ajax: _ajax,
    createURL: _createURL,

    startLoader: _startLoader,
    stopLoader: _stopLoader,

    confirmBox: _confirmBox,
    getDateComponents: _getDateComponents
};
})();

/*
*
* */
slamjam.error = (function () {

    var $selector = null;
    var _enumList = {};
    ["ERROR", "SUCCESS", "INFO", "WARNING"].map(function (item) {

```

```

    _enumList[item] = item;
  });

  var _errorTypes = {
    "ERROR": "danger",
    "SUCCESS": "success",
    "INFO": "info",
    "WARNING": "warning",
  };

  function _initSelector(type) {
    if ($selector === null) {
      // add first,
      $selector = $(".modal-info-text");
    }
  }

  function _print(message, type) {
    //if (_enumList[type] === undefined) return;
    _initSelector(type);

    var msg = `<div>${ message.replace(/\</g, '&lt;').replace(/\>/g, '&gt;')}
  }</div>`;
    $selector.prepend(msg);

    $('.modal-info').css({"display": "block", "opacity": 1});
  }

  return {
    print: _print,
    enumList: _enumList,
  }
})();

/*
 * All about games page and games handling
 */
slamjam.games = (function () {
  //todo

  function _initGamesPage() {
    $('body').on('click', '#pagination-games li a', function (e) {

      e.preventDefault();

      $('.games-container').css('opacity', '0.5');
      slamjam.common.startLoader();

      var url = $(this).attr('href');

      getGames(url);
      //window.history.pushState("", "", url);
    });

    $("#gamesSorter").on('change', function () {
      var value = $(this).val();

      $('.games-container').css('opacity', '0.5');
    });
  }
})();

```

```

slamjam.common.startLoader();

var url = window.location.href;
var newUrl;

var regSort = /([?&]sort)=(^[#&]*)/g;

if (!/[?&]page=/.test(window.location.search)) {
    newUrl = url + "?page=1";
}

if (/[?&]sort=/.test(window.location.search)) {
    newUrl = url.replace(regSort, "$1=" + value);
}
else {
    newUrl += "&sort=" + value;
}

getGames(newUrl);
//window.history.pushState({state:'new'}, "", newUrl);
});

function getGames(url) {
    $.ajax({
        url: url
    }).done(function (data) {
        slamjam.common.stopLoader();
        $('.games-container').css('opacity', '1');
        $('.games-container').html(data);
    }).fail(function () {
        alert('Failed to load games.');
```

```

$( '.owl-carousel' ).owlCarousel({
  loop: false,
  margin: 10,
  autoplay: true,
  stagePadding: 50,
  autoplayHoverPause: true,
  responsive: {
    0: {
      items: 1
    },
    600: {
      items: 2
    },
    1000: {
      items: 4
    }
  }
});

$( '.item a' ).magnificPopup({
  type: 'image',
  mainClass: 'mfp-fade'
});

}

return {
  initGamesPage: _initGamesPage,
  initOneGamePage: _initOneGamePage,
}

})();

/*
 * All about badges
 */
slamjam.badges = (function () {

  function _renderBadge(badgeData) {
    var removeHtml = `<a href="javascript:void(0)" class="remove-badge-a" data-id="${badgeData.idGameSubmissionsBadge}">
      <i class="fa fa-times"></i>
    </a>`;

    var result = `<div class="col-md-4 col-xs-6 col-sm-4 relative-badge">
      
      ${window.__user && badgeData.idUser == __user.idUser ?
removeHtml : ''}
    </div>`;

    return result;
  }

  function _initBadgesOnGamesPage() {

    slamjam.common.ajax({
      url: slamjam.common.createURL(`/games/${idGameSubmission}/badges`),
      success: function (data) {
        if (data && data.length) {

```

```

        var badgesHtml = data.map(function (item) {
            return _renderBadge(item);
        });

        $("#badgesRenderedList").html(badgesHtml.join(''));
    } else {
        $("#badgesRenderedList").html("<i>There is currently no badges
for this game.</i>");
    }
},
error: function (error) {
    var message = "Getting game badges has failed.";
    try {
        message = error.responseJSON.error.message;
    } catch (e) {
        //todo
    }

    slamjam.error.print(message, slamjam.error.enumList.ERROR)
}
});

//init button
$("#btnAddBadge").on('click', function () {

    var badgeId = $("#gamesBadgesList").val();
    if (badgeId == null) return;

    slamjam.common.ajax({
        url:
slamjam.common.createURL(`/games/${idGameSubmission}/badges/${badgeId}`),
        method: "POST",
        success: function (data) {
            var badge = _renderBadge(data);
            var $parent = $("#badgesRenderedList");
            if ($parent.find("i").length) {
                $parent.html(badge);
            } else {
                $parent.append(badge);
            }
        },
        error: function (error) {
            var message = "Posting game badge has failed.";
            try {
                message = error.responseJSON.error.message;
            } catch (e) {
                //todo
            }

            slamjam.error.print(message, slamjam.error.enumList.ERROR)
        }
    });
});

$("#badgesRenderedList").on("click", ".remove-badge-a", function () {
    var $this = $(this);
    var badgeId = $this.data('id');
    if (badgeId) {
        slamjam.common.ajax({

```



```

        url:
slamjam.common.createURL(`/games/${idGameSubmission}/badges/${badgeId}`),
        method: "DELETE",
        success: function (data) {
            $this.parent().remove();

            var $parent = $("#badgesRenderedList");
            if ($parent && $parent.children().length === 0) {
                $parent.html("<i>There is currently no badges for this
game.</i>");
            }
        },
        error: function (error) {
            var message = "Removing game badge has failed.";
            try {
                message = error.responseJSON.error.message;
            } catch (e) {
                //todo
            }

            slamjam.error.print(message, slamjam.error.enumList.ERROR)
        }
    });
}

});
}

return {
    initBadgesOnGamesPage: _initBadgesOnGamesPage,
};
})();

/*
 * All about game jams
 */
slamjam.gameJam = (function () {
    //todo

    function _initChart() {
        // create a dataset with items
        // note that months are zero-based in the JavaScript Date object, so month 3
is April

        function _mapChartData(data) {

            var parsedData = data.map(function (item) {
                return {
                    id: item.idGameJam,
                    group: item.idGameJam, // so items are in one line
                    content: item.title + "<span> - Submissions start!</span>",
                    start: new Date(Number(item.startDate) * 1000),
                    end: new Date(Number(item.endDate) * 1000),
                    link: "game-jams/" + item.idGameJam,
                    className: "chart-game-jam-bar chart-game-jam-bar-blue"
                };
            });

            for (var i = 0; i < data.length; i++) {
                var item = data[i];

```

```

        parsedData.push({
            group: item.idGameJam, // so items are in one line
            content: item.title + "<span> - Voting starts!</span>",
            start: new Date(Number(item.endDate) * 1000),
            end: new Date(Number(item.votingEndDate) * 1000),
            link: "game-jams/" + item.idGameJam,
            className: "chart-game-jam-bar chart-game-jam-bar-red"
        });
    }

    return new vis.DataSet(parsedData);
}

function _mapChartGroups(data) {
    var parsedData = data.map(function (item) {
        return {
            id: item.idGameJam,
            content: "",
            //className: "", // call some type of generator that will return
same css class for some value
        };
    });

    return new vis.DataSet(parsedData);
}

//
// DOCS: http://visjs.org/docs/timeline/
//
function _createChart(data) {
    var timelineDuration = 7; // this is presented in days
    var date = new Date();

    var container = document.getElementById('visualization');
    var options = {
        maxHeight: 400,
        editable: false,
        clickToUse: false,
        zoomable: false,
        selectable: false,
        orientation: 'both', // add both up and down labels
        start: date,
        end: (new Date()).setDate(date.getDate() + timelineDuration),
        template: function (data, x, y) {
            return `\${data.content}</a>`;
        },
        margin: {
            item: {
                horizontal: 0
            }
        },
        stack: false
    };

    var items = \_mapChartData\(data\);
    var groups = \_mapChartGroups\(data\);

    var timeline = new vis.Timeline\(container\);
    timeline.setOptions\(options\);

```

```

        timeline.setItems(items);
        timeline.setGroups(groups);
        timeline.moveTo(date);
    }

    slamjam.common.ajax({
        url: slamjam.common.createURL('/game-jams/chart'),
        success: function (data) {
            if (data) {
                _createChart(data);
            } else {
                $("#no-chart-game-jam").removeClass("hide");
            }
        },
        error: function (error) {
            slamjam.error.print("Fetching game jams for chart has failed.",
slamjam.error.enumList.ERROR)
        }
    });
}

function _initGameJamItems() {
    $('body').on('click', '.pagination-game-jams li a', function (e) {
        e.preventDefault();

        var page = $(this).attr("data-page");
        var gameJamsType = $(this).attr("data-type");
        var gameJamsClass = "";

        if (gameJamsType === "inProgress") {
            gameJamsClass = ".game-jams-in-progress-container";
            $(gameJamsClass).css('opacity', '0.5');
        }
        else {
            gameJamsClass = ".game-jams-upcoming-container";
            $(gameJamsClass).css('opacity', '0.5');
        }

        slamjam.common.startLoader();

        getGameJams(page, gameJamsType, gameJamsClass);
        //window.history.pushState("", "", url);
    });

    function getGameJams(page, gameJamsType, gameJamsClass) {
        $.ajax({
            data: {
                page: page,
                gameJamsType: gameJamsType
            }
        }).done(function (data) {
            slamjam.common.stopLoader();
            $(gameJamsClass).css('opacity', '1');
            $(gameJamsClass).html(data);
        }).fail(function () {
            alert('Failed to load game jams.');
```

```

    }

    return {
        initChart: _initChart,
        initGameJamItems: _initGameJamItems,
    }
})();

/*
 *   All about game jams
 **/
slamjam.search = (function () {

    function _initPage() {
        $('body').on('click', '.pagination-game-jams-search li a', function (e) {
            e.preventDefault();

            var page = $(this).attr("data-page");
            var gameJamsType = $(this).attr("data-type");

            var gameClass = {
                gameJams: "#load-search-game-jams",
                gameSubmissions: "#load-search-game-submission",
            }
            $(gameClass[gameJamsType]).css('opacity', '0.5');

            slamjam.common.startLoader();

            getGameJams(page, gameJamsType, gameClass[gameJamsType]);
        });

        function getGameJams(page, gameJamsType, gameClass) {
            $.ajax({
                data: {
                    page: page,
                    type: gameJamsType
                }
            }).done(function (data) {
                slamjam.common.stopLoader();

                $(gameClass).css('opacity', '1');
                $(gameClass).parent().replaceWith(data);

            }).fail(function () {
                alert('Failed to load game jams.');
```

```

function _renderComment(data) {
    var isCreator = window.__user && data.idUser == __user.idUser;

    var removeHtml = `<a href="javascript:void(0)" class="remove-comment-a" data-
id="${data.idGameSubmissionComment}">
        <i class="fa fa-times"></i>
    </a>`;

    var editHtml = `<a href="javascript:void(0)" class="edit-comment-a" data-
id="${data.idGameSubmissionComment}">
        <i class="fa fa-edit"></i>
    </a>`;

    var result = `<li>
        <div class="comment-main-level">
            <!-- Avatar -->
            <div class="comment-avatar"></div>
            <!-- Contenedor del Comentario -->
            <div class="comment-box">
                <div class="comment-head">
                    <h6 class="comment-name
${idGameSubmissionUserCreatorId == data.idUser ? 'by-author' : (isCreator ? 'by-me' :
'')} "><a href="${slamjam.common.createURL('/user/' + data.username,
true)}">${data.username}</a></h6>
                    <span>${timeagoInstance.format(data.editedAt
+ "000")}</span>

                    ${isCreator ? (removeHtml + editHtml) : ''}
                </div>
                <div class="comment-content">
                    ${data.text}
                </div>
            </div>
        </div>
    </li>`;

    return result;
}

function _getCommentsAndRenderView() {
    slamjam.common.ajax({
        url: slamjam.common.createURL(`/games/${idGameSubmission}/comments`),
        success: function (data) {
            if (data && data.length) {
                var commentsHtml = data.map(function (item) {
                    return _renderComment(item);
                });

                $("#comments-list").html(commentsHtml.join(''));
            } else {
                $("#comments-list").html("<i>There is currently no comment for
this game.</i>");
            }
        },
        error: function (error) {
            var message = "Getting game comments has failed.";
            try {
                message = error.responseJSON.error.message;
            }
        }
    });
}

```

```

        } catch (e) {
            //todo
        }

        slamjam.error.print(message, slamjam.error.enumList.ERROR)
    }
    });
}

function _initAddCommentBiding() {
    //
    //edit binding
    $("#btnAddComment").on("click", function () {

        var comment = $("#comment").val();

        if (comment) {
            slamjam.common.ajax({
                url:
slamjam.common.createURL(`/games/${idGameSubmission}/comments`),
                method: "POST",
                data: {text: comment},
                success: function (data) {
                    if (data) {
                        var commentsHtml = _renderComment(data);
                        $("#comments-list").append(commentsHtml);
                    }

                    //reset form
                    $("#comment").val("");
                },
                error: function (error) {
                    var message = "Posting game comment has failed.";
                    try {
                        message = error.responseJSON.error.message;
                    } catch (e) {
                        //todo
                    }

                    slamjam.error.print(message, slamjam.error.enumList.ERROR)
                }
            });
        }
    });
}

function _initRemoveCommentBiding() {
    $("#comments-list").on("click", ".remove-comment-a", function () {
        var $this = $(this);
        var commentId = $this.data("id");

        if (commentId) {
            slamjam.common.ajax({
                url:
slamjam.common.createURL(`/games/${idGameSubmission}/comments/${commentId}`),
                method: "DELETE",
                success: function (data) {
                    // remove comment

```

```

        $this.closest('li').remove();
    },
    error: function (error) {
        var message = "Removing game comment has failed.";
        try {
            message = error.responseJSON.error.message;
        } catch (e) {
            //todo
        }

        slamjam.error.print(message, slamjam.error.enumList.ERROR)
    }
});
}

});
}

function _initUpdateCommentBiding() {
    $("#comments-list").on("click", ".edit-comment-a", function () {
        var commentid = $(this).data("id");
        var $text = $(this).parent().next();

        if (commentid && $text.length) {
            var editHtml = `<div class="input-group">
                <textarea type="text" class="form-control resize-
vertical">${$text.text().trim()}</textarea>
                <a href="javascript:void(0)" data-id="${commentid}" class="input-
group-addon update-comment-a"><i class="fa fa-check"></i></a>
            </div>`;

            // insert update html
            $text.html(editHtml);
        }
    });

    $("#comments-list").on("click", ".update-comment-a", function () {

        var commentId = $(this).data("id");
        var commentText = $(this).prev().val();
        var $parent = $(this).parent();

        if (commentId && commentText) {

            slamjam.common.ajax({
                url:
slamjam.common.createURL(`/games/${idGameSubmission}/comments/${commentId}`),
                method: "PATCH",
                data: {text: commentText},
                success: function (data) {
                    $parent.html(commentText);
                },
                error: function (error) {
                    var message = "Removing game comment has failed.";
                    try {
                        message = error.responseJSON.error.message;
                    } catch (e) {
                        //todo
                    }
                }
            });
        }
    });
}

```

```

        slamjam.error.print(message, slamjam.error.enumList.ERROR)
    }
    });
}

function _initGameSubmissionComments() {
    //init timeago instance
    timeagoInstance = timeago();

    _getCommentsAndRenderView();

    _initAddCommentBiding();
    _initRemoveCommentBiding();

    _initUpdateCommentBiding();
}

return {
    initGameSubmissionComments: _initGameSubmissionComments,
};
})();

/*
 * Downloads
 * */
slamjam.downloads = (function () {
    var $gameNumberOfDownloads = null;

    function _findDownloadElement() {
        if ($gameNumberOfDownloads === null) {
            $gameNumberOfDownloads = $("#gameNumOfDownloads");
        }
        return $gameNumberOfDownloads;
    }

    function _increment() {
        var $el = _findDownloadElement();

        var value = $el.text();
        if (!isNaN(value)) {
            $el.text((Number(value) + 1));
        }
    }

    return {
        increment: _increment,
    }
})();

/*
 * Validation
 * */
slamjam.validation = (function () {
    // add validation rules to validator
    $('form[data-toggle="validator"]').validator({
        custom: {

```



```

// make it so it works with multiple files

//custom file size validation
filesize: function ($el) {
    var maxKilobytes = $el.data('filesize') * 1024;

    if ($el[0].files[0] && $el[0].files[0].size > maxKilobytes) {
        return "File must be smaller than " + $el.data('filesize') + "
kB."
    }
},
//custom file type validation
filetype: function ($el) {
    var acceptableTypes = $el.data('filetype').split(',');
    var file = $el[0].files[0];

    if (file && acceptableTypes.indexOf(file.type) === -1) {
        return "Invalid file type"
    }
},
"datetime-gt": function ($el) {
    //
    // this is not that robust
    // no time for now
    //
    var datetimeformat = $el.data('datetime-gt');
    var value = $el.val();
    var ONE_DAY = 86400;

    if (value) {
        var isDef = false;

        switch (datetimeformat) {
            case 'now':
                //check if time is gt then Date.now, if not return false
                break;
            case 'one-day':
                if (new Date(value).getTime() < (Date.now() + ONE_DAY)) {
                    return "Date need to be at least 1 day from now.";
                }
                break;
            default:
                isDef = true;
                break;
        }

        //
        //this is default case
        if (isDef) {
            var fromEl = $(datetimeformat).val();

            if (fromEl && new Date(value).getTime() < (new
Date(fromEl).getTime() + ONE_DAY)) {
                return "Date need to be at least 1 day from time
before.";
            }
        }
    }
}
}

```

```

        },
        //add more if needed
    }
});

return {};
})();

```

3. CSS

```
#### public\css\admin-dashboard.css
```

```

    ...

    .card {
display: -ms-flexbox;
display: flex;
-ms-flex-direction: column;
flex-direction: column;
min-width: 0;
word-wrap: break-word;
background-clip: border-box;

border-radius: 12px;
box-shadow: 0 6px 10px -4px rgba(0,0,0,.15);
/*background-color: #fff;*/
color: #e4e4e4;
margin-bottom: 20px;
position: relative;
border: 0 none;
transition: transform .3s cubic-bezier(.34,2,.6,1),box-shadow .2s ease;
}

.card-stats .card-body{
padding: 15px 15px 0;
}

.card-stats .card-body .numbers{
text-align: right;
font-size: 2em;
}

.card-stats .card-body .numbers .card-category{
color: #cacaca;
font-size: 16px;
line-height: 1.4em;
}

.card-stats .card-body .numbers p{
margin-bottom: 0;
}

.card-stats .card-footer{
padding: 0 15px 15px;
}

```

```

.card-stats .card-footer .stats{
    color: #cacaca;
}

.card-stats .icon-big{
    font-size: 3em;
    min-height: 64px;
}
.card-stats .icon-big i{
    line-height: 59px;
}

.card-footer a {
    color: #647992;
}

.card-footer a:hover {
    color: #aladbb;
}

hr.sick {
    border-top: 1px solid #323540;
    border-bottom: 1px solid #1a1c23;
}
    ...

```

```

#### public\css\admin.css
    ...

```

```

    html, body, div, span, h1, h2, h3, h4, h5, h6, p, pre,
a, abbr, strong, b, u, i, ol, ul, li, form, label,
table, tbody, tr, th, td, input, li a {
    margin: 0;
    padding: 0;
}

```

```

html, body {
    background-color: #292c35;
    width: 100%;
    height: 100%;
    font-family: 'Roboto', sans-serif;
    color: #fff;
    font-size: 16px;
}

```

```

table {
    border-collapse: collapse;
    border-spacing: 0;
}

```

```

.float-left {
    float: left;
}

```

```

}

.float-right {
    float: right;
}

.strike-through {
    text-decoration: line-through;
}

.main-container {
    min-height: 500px;
}

.menu-container {
    position: fixed;
    height: 100%;
    left: 0;
    z-index: 5001;
}

.menu {
    padding: 1.5em 1em 1.5em 0.8em;
    margin: 0 -20px 0 0;
    float: left;
    height: 100%;
    width: 257px;
    left: 0;
    background-color: #1f2229;
    z-index: 5000;
    position: relative;
    overflow-y: scroll;
}

.menu-open {
    float: right;
    height: 100%;
    width: 36px;
    background-color: #16181d;
    border-right: 1px solid #1d1f25;
    box-shadow: 4px 0 12px #1a1d24;
    z-index: 5001;
    position: relative;
    transition: .25s width;
}

.menu-open span{
    height: 17px;
    display: block;
    width: 24px;
    position: relative;
    top: 36px;
    left: 7px;
    border-bottom: 3px solid transparent;
    cursor: pointer;
    opacity: 0.8;
    transition: top .25s, border-bottom-color .25s, width .25s, height .25s, left
    .25s, opacity .2s;
}

```

```
.menu-open span:hover {
  opacity: 1;
}

.menu-open span:before {
  content: "";
  display: block;
  border-bottom: 3px solid #fff;
  margin-bottom: 5px;
  position: relative;
  transition: transform .25s ease-in-out, top .25s ease-in-out;
  outline: 1px solid transparent;
  transform: rotate(45deg);
  top: 9px;
}

.menu-open span:after {
  content: "";
  display: block;
  border-bottom: 3px solid #fff;
  margin-bottom: 5px;
  position: relative;
  transition: transform .25s ease-in-out, top .25s ease-in-out;
  outline: 1px solid transparent;
  transform: rotate(-45deg);
  top: 1px;
}

.menu-open span.click {
  top: 38px;
  border-bottom-color: #fff;
  width: 30px;
  height: 19px;
}

.menu-open span.click:before {
  transform: rotate(0deg);
  top: -2px;
}

.menu-open span.click:after {
  transform: rotate(0deg);
  top: -1px;
}

.title {
  font-family: 'Anton', sans-serif;
  letter-spacing: 4px;
  font-size: 2.5em;
  margin-bottom: 0;
  padding-left: 0.2em;
  padding-bottom: 0;
}

.title a {
  color: #d5dcf1;
  text-decoration: none;
  outline: 0;
}
```

```

        transition: .2s color;
    }

    .title a:hover {
        color: #f4f7ff;
    }

    .sub-title {
        color: #6f7890;
        margin-top: 0;
        margin-bottom: 2em;
        font-size: 0.8em;
        letter-spacing: 1px;
        padding-left: 0.8em;
        padding-top: 0;
    }

    .dashboards-title {
        font-size: 0.9em;
        color: #51586a;
        padding: 0 0.8em;
        margin-bottom: 0.6em;
        letter-spacing: 1px;
        font-size: 0.8em;
    }

    .admin-log-out {
        font-size: 1.3em;
        color: #fff;
        position: absolute;
        bottom: 15px;
        text-align: center;
        display: inline-block;
        width: 100%;
        opacity: 0.8;
        transition: .2s opacity;
    }

    .admin-log-out:hover {
        color: #fff;
        opacity: 1;
    }

    .admin-menu-list {
        list-style-type: none;
        padding-bottom: 0.8em;
    }

    .admin-menu-list li a {
        display: block;
        text-decoration: none;
        color: #cfd2da;
        padding: 0.5em 0.8em;
        border-radius: 0.4em;
        font-size: 0.9em;
        margin: 0.1em 0;
        outline: 0;
        transition: background-color .15s, color .15s;
    }

```

```

.admin-menu-list li.active a {
    background-color: #1997c6;
    color: #fff;
}

.admin-menu-list li.active a:hover {
    background-color: #21a1cf;
}

.admin-menu-list li a:hover {
    color: #fff;
}

.main-content {
    padding: 1.8em 2em 1.5em 1.5em;
    margin-left: 283px;
    background-color: #292c35;
    position: relative;
}

.loading-overlay {
    display: none;
    position: fixed;
    z-index: 30000;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.4);
    opacity: 1;
}

.loading-overlay img {
    width: 100px;
    height: 100px;
    top: 50%;
    left: 50%;
    position: relative;
    transform: translate(-50%, -50%);
}

.main-content h2 {
    padding-bottom: 0.8em;
    font-size: 2em;
}

.content {
    border-top: 1px solid #6b7183;
}

.main-commands {
    padding: 0 0 0.5em 0.25em;
}

.main-commands a, .inner-table tbody tr td a.add-new-answer-btn {
    background-color: #23262e;
    color: #d1d5e2;
    border: 1px solid #d1d5e2;
    border-radius: 0.4em;
    display: inline-block;
}

```

```

padding: 0.4em 0.6em 0.2em 0.5em;
text-decoration: none;
width: 2.2em;
position: relative;
transition: .2s all;
box-shadow: 2px 2px 5px #1e2129;
outline: 0;
}

.main-commands a:hover, .inner-table tbody tr td a.add-new-answer-btn:hover {
color: #fafbfd;
border: 1px solid #fafbfd;
}

.main-commands i {
font-size: 1.2em;
vertical-align: top;
padding-right: 0.5em;
}

.no-data-found {
font-style: italic;
padding-top: 25px;
font-size: 1.2em;
}

.main-table, .inner-table {
width: 100%;
max-width: 100%;
margin-bottom: 1em;
background-color: transparent;
white-space: nowrap;
box-shadow: 5px 4px 8px #1f2129;
}

.main-table a {
color: #67a7d8;
transition: .15s color;
text-decoration: none;
}

.main-table a:hover {
color: #b6dfff;
}

.main-table tbody tr {
transition: .15s background-color;
}

.main-table tbody tr:nth-child(odd) {
background-color: #2d313b;
}

.poll-table tbody tr.table-odd-row {
background-color: #292c35;
}

.main-table tbody tr:not(.table-poll-answers):hover {
background-color: #353a46;
}

```



```

}

.main-table tbody tr td.text-center, .main-table thead tr th.text-center {
    text-align: center;
}

.main-table tr th, .main-table tr td {
    color: #e8eaf0;
    text-align: left;
    border: none;
    padding-right: 1em;
}

.main-table tr th {
    padding-top: 1em;
    padding-bottom: 1em;
    font-weight: bold;
    border-bottom: 2px solid #434857;
    letter-spacing: 0;
}

.main-table tr td {
    padding-top: 0.5em;
    padding-bottom: 0.5em;
    border-top: 1px solid #434857;
    font-size: 0.9em;
}

.main-table thead tr th:nth-child(1), .main-table tbody tr td:nth-child(1) {
    width: 45px;
    padding-left: 0.8em;
}

.main-table thead tr th:nth-child(2), .main-table tbody tr td:nth-child(2) {
    padding-right: 0.5em;
    padding-left: 0.3em;
    width: 50px;
}

.main-table tbody tr td.table-cell-id {
    color: #67a7d8;
}

.main-table tr td.data-date {
    width: 120px;
    font-size: 0.8em;
}

.main-table tr td.data-user-image img {
    width: 48px;
    height: 48px;
    border-radius: 50%;
    border: 2px solid #424857;
    box-shadow: 2px 2px 5px #1f2129;
}

.main-table tr .data-edit, .main-table tr .data-delete {
    width: 3.5em;
}

```

```
.main-table tr td.data-edit {
    padding-left: 0.4em;
}

.data-edit a, .data-delete a {
    font-size: 1.6em;
    color: #67a7d8;
    transition: .2s color;
    outline: 0;
}

.data-edit a:hover, .data-delete a:hover {
    color: #b6dfff;
}

.data-user-role i {
    color: #c0d1df;
    font-size: 1.7em;
    padding: 0 0.2em;
    transition: .2s color;
    cursor: pointer;
}

.data-user-role i:hover {
    color: rgb(221, 230, 250);
}

.data-cover {
    width: 80px;
    height: 50px;
}

.inner-table-wrap {
    padding: 0 0.8em;
}

.inner-table {
    width: 100%;
}

.inner-table tbody tr {
    border: none;
}

.inner-table tbody tr td:nth-child(1) {
    color: #67a7d8;
    padding-left: 1.6em;
}

.inner-table tbody tr td:nth-child(2) {
    width: 90%;
    padding-left: 1.8em;
}

.main-table tbody tr.table-poll-answers {
    background-color: #292c35;
}
```

```

.main-table tbody tr.table-poll-answers > td {
    padding-top: 0;
    padding-bottom: 0;
    transition: .2s padding;
}

.table-cell-70 {
    width: 70px;
}

.inner-table-wrap {
    display: none;
}

.inner-table tbody tr:nth-child(1) td {
    border-top: none;
}

.expand-poll-question {
    cursor: pointer;
    display: block;
    width: 25px;
    height: 25px;
}

.expand-poll-question:before {
    content: "";
    display: block;
    border-bottom: 3px solid #fff;
    margin-bottom: 5px;
    position: relative;
    transition: transform .25s ease-in-out, top .25s ease-in-out;
    outline: 1px solid transparent;
    transform: rotate(45deg);
    top: 13px;
    left: 0px;
    width: 15px;
}

.expand-poll-question:after {
    content: "";
    display: block;
    color: #fff;
    border-bottom: 3px solid #fff;
    margin-bottom: 5px;
    position: relative;
    transition: transform .25s ease-in-out, top .25s ease-in-out;
    outline: 1px solid transparent;
    transform: rotate(-45deg);
    top: 5px;
    left: 9px;
    width: 15px;
}

.expand-poll-question.click:before {
    transform: rotate(-45deg);
}

.expand-poll-question.click:after {

```

```

        transform: rotate(45deg);
    }

    .inner-table tbody tr.add-new-answer td {
        padding-top: 1.1em;
        padding-bottom: 0;
        padding-left: 1.4em;
    }

    .inner-table tbody tr.add-new-answer td form {
        display: inline-block;
        width: 40%;
    }

    .inner-table tbody tr.add-new-answer td:nth-child(2) {
        padding: 0;
    }

    .inner-table tbody tr td a.add-new-answer-btn i {
        font-size: 1.2em;
    }

    .inner-table tbody tr td a.add-new-answer-btn {
        padding-left: 0.7em;
        font-size: 1.1em;
        width: 2.5em;
        margin-top: 0em;
        margin-left: 0.5em;
        padding-top: 0.5em;
        padding-bottom: 0.3em;
        display: inline-block;
        transform: translateY(1px);
    }

    /* ----- Checkboxes ----- */

    .checkbox-cell {
        display: block;
        position: relative;
        cursor: pointer;
        width: 100%;
        -webkit-user-select: none;
        -moz-user-select: none;
        -ms-user-select: none;
        user-select: none;
    }

    .checkbox-cell input {
        position: absolute;
        opacity: 0;
        cursor: pointer;
    }

    .checkmark {
        position: absolute;
        top: -10px;
        left: 50%;
    }

```

```

    transform: translateX(-11px);
    height: 22px;
    width: 22px;
    background-color: #eee;
}

.checkbox-cell:hover input ~ .checkmark {
    background-color: #eee;
}

.checkbox-cell input:checked ~ .checkmark {
    background-color: #549ed6;
}

.checkmark:after {
    content: "";
    position: absolute;
    display: none;
}

.checkbox-cell input:checked ~ .checkmark:after {
    display: block;
}

.checkbox-cell .checkmark:after {
    left: 8px;
    top: 4px;
    width: 7px;
    height: 12px;
    border: solid white;
    border-width: 0 3px 3px 0;
    -webkit-transform: rotate(45deg);
    -ms-transform: rotate(45deg);
    transform: rotate(45deg);
}

/* ----- */

/* ----- Modal ----- */

.modal-box {
    display: none;
    opacity: 0;
    background-color: rgba(0, 0, 0, 0.3);
    height: 100%;
    width: 100%;
    position: fixed;
    z-index: 20000;
    padding: 0;
    margin: 0;
    top: 0;
    left: 0;
    border-radius: 5px;
    color: #222;
}

.modal-box-inner {
    width: 440px;
    position: absolute;

```

```
    top: 50%;
    left: 50%;
    background-color: #fff;
    transform: translate(-50%, -53%);
    box-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);
}

.modal-box-header {
    background-color: #0f1114;
    height: 30px;
}

.modal-box-content {
    padding: 25px 30px 5px;
}

.modal-close {
    width: 40px;
    height: 30px;
    float: right;
    text-align: center;
    cursor: pointer;
    transition: .1s background-color;
}

.modal-close:hover {
    background-color: #e81123;
}

.modal-close i {
    color: #fff;
    font-size: 22px;
    padding-top: 4px;
}

.modal-confirm {
    display: block;
    float: right;
    padding: 0 8px 6px 0;
}

.modal-confirm a {
    padding-left: 20px;
    padding-right: 20px;
    transition: .15s background-color;
}

#modal-confirm-yes {
    background-color: #57a1da;
}

#modal-confirm-yes:hover {
    background-color: #3680b8;
}

.data-form h3 {
    font-size: 1.4rem;
    padding-bottom: 15px;
    text-align: center;
}
```

```

}

.control-title {
  font-size: 0.9rem;
  padding-bottom: 0.2rem;
}

.modal-box-content label {
  font-weight: normal;
  font-size: 0.9rem;
}

.modal-box-content input[type='checkbox'] {
  margin-right: 10px;
}

.form-group p {
  padding-bottom: 5px;
  font-size: 0.9rem;
}

.checkbox {
  padding-top: 0;
  padding-bottom: 2px;
  margin-top: 0;
  margin-bottom: 3px;
}

form p {
  font-size: 0.9rem;
  font-weight: 700;
}

#form-errors {
  display: none;
  font-size: 0.8rem;
  padding-top: 0.6rem;
  padding-bottom: 0.6rem;
}

/* ----- */

/* ----- Tooltip ----- */

.has-tooltip {
  position: relative;
}

.has-tooltip:hover .tooltip {
  opacity: 1;
  visibility: visible;
  top: -43px;
}

.tooltip {
  font-size: 0.8rem;
  color: #fff;
  font-family: 'Roboto', sans-serif;
  font-weight: 300;
}

```

```

    text-align: center;
    position: absolute;
    top: -56px;
    left: -39px;
    padding: 6px 15px;
    width: 105px;
    display: block;
    background-color: #1a1d24;
    box-shadow: 1px 1px 3px #242630;
    padding: 8px 0;
    border-radius: 7px;
    transform: translateZ(0);
    opacity: 0;
    visibility: hidden;
    transition: .2s all;
    pointer-events: none;
    z-index: 10000;
}

.tooltip:after {
    content: "";
    width: 0;
    height: 0;
    border-right: 7px solid transparent;
    border-left: 7px solid transparent;
    border-top: 7px solid #1a1d24;
    position: absolute;
    top: 33px;
    left: 49px;
}

.tooltip-small {
    left: -16px;
    width: 60px;
}

.tooltip-small:after {
    left: 26px;
}

/* ----- */

/* ----- Edit user ----- */

.user-edit-file {
    height: 140px;
    font-size: 14px;
    line-height: 1.5;
    margin-top: 5px;
    color: #555;
    background-color: #fff;
    background-image: none;
    border: 1px solid #ccc;
    border-radius: 4px;
    -webkit-box-shadow: inset 0 1px 1px rgba(0,0,0,.075);
    box-shadow: inset 0 1px 1px rgba(0,0,0,.075);
    position: relative;
}

```



```

input.user-edit-upload-file-control {
  display: block;
  height: 140px;
  width: 100%;
  position: relative;
  z-index: 10;
  opacity: 0;
  cursor: pointer;
  border: 1px solid #000;
}

.drag-and-drop-overlay {
  position: absolute;
  top: 16px;
  left: 0;
  display: block;
  width: 100%;
  text-align: center;
  opacity: 1;
  transition: .2s opacity;
}

.drag-and-drop-overlay i{
  color: skyblue;
  font-size: 64px;
}

.drag-and-drop-overlay p {
  font-size: 18px;
  margin-top: 15px;
}

.user-edit-file img {
  width: 90px;
  height: 90px;
  border-radius: 50%;
  border: 3px solid #b9c1db;
  position: absolute;
  top: 50%;
  left: 50%;
  margin-top: -60px;
  margin-left: -45px;
  z-index: 9;
  opacity: 0;
  transition: 2s opacity;
}

.image-filename {
  position: absolute;
  bottom: 10px;
  width: 100%;
  text-align: center;
  opacity: 0;
  transition: 1s opacity;
}

.is-banned-form {
  width: 150px;
}

```

```
.is-banned-form label {
    cursor: pointer;
}

/* ----- */
    \ \ \
```

```
#### public\css\game.css
```

```
    \ \ \
```

```
        .comments-container {
margin: 60px auto 15px;
width: 768px;
}

.comments-container h1 {
    font-size: 36px;
    color: #283035;
    font-weight: 400;
}

.comments-container h1 a {
    font-size: 18px;
    font-weight: 700;
}

.comments-list {
    margin-top: 30px;
    position: relative;
}

/**
 * Lineas / Detalles
 -----*/
.comments-list:before {
    content: '';
    width: 2px;
    height: 100%;
    background: #c7cacb;
    position: absolute;
    left: 32px;
    top: 0;
}

.comments-list:after {
    content: '';
    position: absolute;
    background: #c7cacb;
    bottom: 0;
    left: 27px;
    width: 7px;
```

```

    height: 7px;
    border: 3px solid #dee1e3;
    -webkit-border-radius: 50%;
    -moz-border-radius: 50%;
    border-radius: 50%;
}

.reply-list:before, .reply-list:after {display: none;}
.reply-list li:before {
    content: '';
    width: 60px;
    height: 2px;
    background: #c7caca;
    position: absolute;
    top: 25px;
    left: -55px;
}

.comments-list li {
    margin-bottom: 15px;
    display: block;
    position: relative;
}

.comments-list li:after {
    content: '';
    display: block;
    clear: both;
    height: 0;
    width: 0;
}

.reply-list {
    padding-left: 88px;
    clear: both;
    margin-top: 15px;
}

/**
 * Avatar
 -----*/
.comments-list .comment-avatar {
    width: 65px;
    height: 65px;
    position: relative;
    z-index: 99;
    float: left;
    border: 3px solid #FFF;
    -webkit-border-radius: 4px;
    -moz-border-radius: 4px;
    border-radius: 4px;
    -webkit-box-shadow: 0 1px 2px rgba(0,0,0,0.2);
    -moz-box-shadow: 0 1px 2px rgba(0,0,0,0.2);
    box-shadow: 0 1px 2px rgba(0,0,0,0.2);
    overflow: hidden;
}

.comments-list .comment-avatar img {
    width: 100%;

```

```

        height: 100%;
    }

    .reply-list .comment-avatar {
        width: 50px;
        height: 50px;
    }

    .comment-main-level:after {
        content: '';
        width: 0;
        height: 0;
        display: block;
        clear: both;
    }
/**
 * Caja del Comentario
 -----*/
    .comments-list .comment-box {
        width: 80%;
        float: right;
        position: relative;
        -webkit-box-shadow: 0 1px 1px rgba(0,0,0,0.15);
        -moz-box-shadow: 0 1px 1px rgba(0,0,0,0.15);
        box-shadow: 0 1px 1px rgba(0,0,0,0.15);
    }

    .comments-list .comment-box:before, .comments-list .comment-box:after {
        content: '';
        height: 0;
        width: 0;
        position: absolute;
        display: block;
        border-width: 10px 12px 10px 0;
        border-style: solid;
        border-color: transparent #FCFCFC;
        top: 8px;
        left: -11px;
    }

    .comments-list .comment-box:before {
        border-width: 11px 13px 11px 0;
        border-color: transparent rgba(0,0,0,0.05);
        left: -12px;
    }

    .reply-list .comment-box {
        width: 85%;
    }

    .comment-box .comment-head {
        background: #FCFCFC;
        padding: 10px 12px;
        border-bottom: 1px solid #E5E5E5;
        overflow: hidden;
        -webkit-border-radius: 4px 4px 0 0;
        -moz-border-radius: 4px 4px 0 0;
        border-radius: 4px 4px 0 0;
    }

```

```
.comment-box .comment-head i {
    float: right;
    margin-left: 14px;
    position: relative;
    top: 2px;
    color: #A6A6A6;
    cursor: pointer;
    -webkit-transition: color 0.3s ease;
    -o-transition: color 0.3s ease;
    transition: color 0.3s ease;
}

.comment-box .comment-head i:hover {
    color: #03658c;
}

.comment-box .comment-name {
    color: #283035;
    font-size: 14px;
    font-weight: 700;
    float: left;
    margin-right: 10px;
}

.comment-box .comment-name a {
    color: #283035;
}

.comment-box .comment-head span {
    display: inline-block;
    color: #999;
    font-size: 13px;
    margin-top: 9px;
}

.comment-box .comment-content {
    background: #FFF;
    padding: 12px;
    font-size: 15px;
    color: #595959;
    -webkit-border-radius: 0 0 4px 4px;
    -moz-border-radius: 0 0 4px 4px;
    border-radius: 0 0 4px 4px;
}

.comment-box .comment-name.by-author, .comment-box .comment-name.by-author a {color: #03658c;}
.comment-box .comment-name.by-author:after {
    content: 'autor';
    background: #03658c;
    color: #FFF;
    font-size: 12px;
    padding: 3px 5px;
    font-weight: 700;
    margin-left: 10px;
    -webkit-border-radius: 3px;
    -moz-border-radius: 3px;
    border-radius: 3px;
}
```

```

.comment-box .comment-name.by-me, .comment-box .comment-name.by-me a {color:
#8900d0;}
.comment-box .comment-name.by-me:after {
    content: 'me';
    background: #8900d0;
    color: #FFF;
    font-size: 12px;
    padding: 3px 5px;
    font-weight: 700;
    margin-left: 10px;
    -webkit-border-radius: 3px;
    -moz-border-radius: 3px;
    border-radius: 3px;
}

/** =====
 * Responsive
 =====*/
@media only screen and (max-width: 766px) {
    .comments-container {
        width: 480px;
    }

    .comments-list .comment-box {
        width: 80%;
    }

    .reply-list .comment-box {
        width: 80%;
    }
}

```

```

#### public\css\main.css

```

```

...

```

```

    * {
margin: 0;
padding: 0;
}

```

```

/* Search input */

```

```

#tbSearch {
    transition: .2s width;
    width: 150px;
}

```

```

@media only screen and (min-width: 960px) {
    #tbSearch.expand-width {
        width: 260px;
    }
}

```

```

    }

    #tbSearch.shrink-width {
        width: 150px;
    }
}

@media only screen and (max-width: 768px) {
    #tbSearch {
        width: 100%;
    }
}

/* ----- */

.nav-title {
    margin-right: 6px;
}

.navbar {
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.4);
    background-color: rgb(28, 28, 28);
}

.slam-jam-logo {
    width: 40px;
    height: 40px;
    float: right;
    margin-top: -8px;
    margin-left: 10px;
}

.float-left {
    float: left;
}

.float-right {
    float: right;
}

.custom-footer {
    padding: 15px 0 25px;
    background-color: rgb(25, 25, 25);
    color: #eee;
}

.carousel-item {
    height: 65vh;
    min-height: 300px;
    background: no-repeat center center scroll;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    background-size: cover;
}

.portfolio-item {
    margin-bottom: 30px;
}

```

```
a.host-own-game-jam, button.host-own-game-jam {
  width: 100%;
  margin: 25px 0;
}

@media screen and (max-width: 768px) {
  a.host-own-game-jam, button.host-own-game-jam {
    margin-top: 20px;
  }
}

#visualization {
  box-sizing: border-box;
  width: 100%;
  min-height: 100px;
  height: 100%;
  margin-bottom: 40px;
}

div.no-padding {
  padding: 0;
}

p.margin-bottom-5 {
  margin-bottom: 5px;
}

h2.margin-bottom-40 {
  margin-bottom: 40px;
}

h3.no-game-jam {
  padding-top: 30px;
  padding-bottom: 40px;
}

p.p-joind-submissions {
  margin-top: 10px;
  margin-bottom: 0px;
  font-size: small;
}

.main-container {
  min-height: 600px;
  position: relative;
}

.show {
  visibility: visible !important;
}

header {
  margin-bottom: 30px;
  margin-top: 30px;
}

.card-img-top {
  width: 100%;
}
```



```
        height: 100%;
    }

/* ----- */

/* Registration - login */

.auth-box-body {
    width: 500px;
    margin: 0 auto;
}

.auth-box-msg {
    text-align: center;
}

.auth-title {
    font-size: 20px;
    padding: 20px 0;
}

/* ----- */

/* About */

.about-title {
    padding: 20px 0;
}

.about-content {
    width: 800px;
}

.bg-white {
    background-color: white;
}

/* ----- */

/* User profile */

.user-profile {
    padding-top: 35px;
    width: 700px;
    margin-left: 50%;
    transform: translateX(-50%);
}

.table-user-information tr td {
    padding: 15px 0 !important;
}

/* ----- */

/* Edit profile */

.user-edit-content {
    width: 500px;
    margin: 0 auto;
}
```

```

}

.user-edit-file input {
    height: auto;
}

.user-edit-title {
    font-size: 20px;
    padding: 20px 0;
}

.btn-update-profile {
    margin-top: 15px;
}

.user-edit-control {
    position: relative;
}

.user-edit-roles {
    padding: 5px 0 0;
    margin-bottom: 0;
}

.user-edit-file {
    height: 260px;
    font-size: 14px;
    line-height: 1.42857143;
    color: #555;
    background-color: #fff;
    background-image: none;
    border: 1px solid #ccc;
    border-radius: 4px;
    -webkit-box-shadow: inset 0 1px 1px rgba(0, 0, 0, .075);
    box-shadow: inset 0 1px 1px rgba(0, 0, 0, .075);
    position: relative;
}

input.user-edit-upload-file-control {
    display: block;
    height: 260px;
    width: 100%;
    position: relative;
    z-index: 10;
    opacity: 0;
    cursor: pointer;
    border: 1px solid #000;
}

.drag-and-drop-overlay {
    position: absolute;
    top: 68px;
    left: 0;
    display: block;
    width: 100%;
    text-align: center;
    opacity: 1;
    transition: .2s opacity;
}

```

```

.drag-and-drop-overlay i {
    color: skyblue;
    font-size: 64px;
}

.drag-and-drop-overlay p {
    font-size: 22px;
    margin-top: 15px;
}

.user-edit-file img {
    width: 200px;
    height: 200px;
    border-radius: 50%;
    border: 5px solid rgba(0, 0, 0, 0.25);
    position: absolute;
    top: 50%;
    left: 50%;
    margin-top: -113px;
    margin-left: -100px;
    z-index: 9;
    opacity: 0;
    transition: 2s opacity;
}

.image-filename {
    position: absolute;
    bottom: 3px;
    width: 100%;
    text-align: center;
    opacity: 0;
    transition: 1s opacity;
}

/* ----- */

textarea.no-resize {
    resize: none;
}

textarea.resize-vertical {
    resize: vertical;
}

footer.custom-footer {
    margin-top: 40px;
}

#clockdiv {
    font-family: sans-serif;
    color: #fff;
    display: inline-block;
    font-weight: 100;
    text-align: center;
    font-size: 32px;
}

#clockdiv > div {

```

```

padding: 10px;
border-radius: 3px;
background: #00BF96;
display: inline-block;
}

#clockdiv div > span {
padding: 15px;
border-radius: 3px;
background: #00816A;
display: inline-block;
}

.countdown-timer-title {
padding: 10px 0 15px;
}

/* ----- */

.smalltext {
padding-top: 5px;
font-size: 16px;
}

.one-game-jam-content {
min-height: 400px;
}

.reg-role-description {
display: block;
margin-left: 20px;
color: #aaa;
}

.gray-color-social a {
color: #4e555b;
margin-right: 5px;
}

#social-fb:hover {
color: #3B5998;
}

#social-yt:hover {
color: red;
}

#social-tw:hover {
color: #4099FF;
}

#social-gp:hover {
color: #d34836;
}

#social-em:hover {
color: #f39c12;
}

```

```
#social-pr: hover {
    color: #bd081c;
}

.list-no-style {
    list-style: none;
}

.split-right {
    -webkit-box-ordinal-group: 0;
    -webkit-order: -1;
    -ms-flex-order: -1;
    order: -1;
    color: #333;
    background: #fff;
    background-image: url("../images/about-background2.png");
    background-size: cover;
    background-position: center center;
    background-repeat: no-repeat;
}

.split-left {
    background: #7b5a9e;
    color: #fff;
    -webkit-box-ordinal-group: 0;
    -webkit-order: -1;
    -ms-flex-order: -1;
    order: -1;
    background-image: url("../images/trianglify-guran-background.png");
    background-size: cover;
    background-position: center center;
    background-repeat: no-repeat;
}

.split-left, .split-right {
    height: 305px;
}

.split-image img {
    border: 5px solid rgba(0, 0, 0, 0.2);
    width: 100%;
    height: auto;
    margin: 2em auto;
    border-radius: 50%;
    max-width: 100px;
    display: block;
}

.user-nav-avatar {
    border: 5px solid rgba(255, 255, 255, 0.2);
    width: 48px;
    border-radius: 30%;
}

.avatar-li {
    padding-top: 1px;
    margin-left: 10px;
}
```

```
.loading-overlay {
    display: none;
    position: fixed;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
    z-index: 1000;
}

.game-cover-image-wrap {
    height: 300px;
    position: relative;
    overflow: hidden;
}

.game-cover-image {
    background-repeat: no-repeat;
    background-size: cover;
    background-position: center center;
    height: 550px;
    position: absolute;
    width: 100%;
}

.game-header {
    background-color: #111;
    padding: 20px 0;
    color: #fff;
}

.game-header span {
    display: inline-block;
}

.game-content {
    margin: 15px auto;
}

.game-content-left {
    min-height: 200px;
}

.game-content-right {
    min-height: 200px;
}

.game-comments {
    min-height: 100px;
}

/* Games page */

.games-container {
    transition: .1s opacity;
}

.games-content {
```

```

        padding: 25px 0 0;
    }

.games-title {
    margin: 40px 0;
}

span.game-category {
    border: 1px solid #555;
    border-radius: 3px;
    display: inline-block;
    padding: 2px 6px;
    font-size: 12px;
    color: #555;
}

.games-content-item {
    min-height: 270px;
    margin-bottom: 15px;
    padding: 0 10px;
}

.pagination > li > a {
    border-color: #999;
}

@media only screen and (max-width: 768px) {
    .games-content-item {
        margin-bottom: 30px;
    }
}

@media only screen and (max-width: 1200px) {
    .games-content-item {
        min-height: 250px;
    }
}

/* ----- */

.main-container.one-game-container {
    width: 100%;
    padding: 0;
    margin: 0;
}

.navbar.one-game-container {
    margin-bottom: 0;
}

.game-jam-header-title {
    margin-top: 0;
}

.game-jam-join-buttons {
    text-align: center;
    padding-top: 30px;
    height: 100px;
}

```

```
.game-jam-join-button-holder {
  display: inline-block;
  vertical-align: bottom;
}

.game-jam-button-block {
  display: inline-block;
  vertical-align: top;
}

.game-jam-join-button, .game-jam-leave-button, .game-jam-add-button {
  padding: 10px 50px;
  margin-top: 0;
  border-radius: 6px;
  font-size: 20px;
  color: #FFF;
  text-decoration: none;
  border: none;
  margin-left: 5px;
  display: inline-block;
}

.game-jam-join-button:hover, .game-jam-leave-button:focus, .game-jam-leave-
button:hover, .game-jam-add-button:hover {
  border-bottom-width: 0;
  margin-top: 3px;
  color: #fff;
  text-decoration: none;
  outline: none;
}

.game-jam-join-button {
  background-color: #52A0FD;
  border-bottom: 3px solid #2980B9;
}

.game-jam-leave-button {
  background-color: #ea5455;
  border-bottom: 3px solid #ff1111;
}

.game-jam-add-button {
  background-color: #52A0FD;
  border-bottom: 3px solid #2980B9;
}

.game-jam-remove-block {
  position: absolute;
}

.alert-parent {
  position: absolute;
  width: 100%;
  margin: 0 auto;
  z-index: 10000;
}

.alert-parent > .alert {
```



```

    margin: 0 auto;
    width: 80%;
}

.pagination-game-jams {
    margin-left: 25px;
}

.game-jam-participant-tab-row {
    padding-top: 20px;
}

.game-jam-submission-tab-row {
    padding-bottom: 15px;
}

.game-jam-participant-tab-row a, .game-jam-submission-tab-row a {
    display: inline-block;
    padding: 5px 10px;
    font-size: 16px;
}

.game-jam-participant-tab-row img {
    width: 40px;
    height: 40px;
    object-fit: cover;
}

.game-jam-submission-tab-row img {
    width: 160px;
    height: 70px;
    object-fit: cover;
    vertical-align: bottom;
}

.game-jam-submission-tab-row-info {
    display: inline-block;
}

.game-jam-submission-tab-row-info p {
    font-size: 14px;
    padding-left: 10px;
}

.game-jam-dates {
    padding-bottom: 15px;
    height: 167px;
}

.game-jam-date {
    font-size: 1em;
    display: inline-block;
    position: relative;
    width: 110px;
    height: 110px;
    background-color: #fff;
    margin: 5px 7px 10px;
    border-radius: 6px;
}

```

```

    box-shadow: 0 1px 0 #bdbdbd, 0 2px 0 #fff, 0 3px 0 #bdbdbd, 0 4px 0 #fff, 0 5px 0
    #bdbdbd, 0 0 0 1px #bdbdbd;
    cursor: pointer;
    transition: .2s height;
}

.game-jam-date:hover {
    height: 130px;
}

.game-jam-date:hover .game-jam-date-time {
    opacity: 1;
}

.game-jam-date * {
    display: block;
    width: 100%;
    font-size: 1em;
    font-weight: bold;
    font-style: normal;
    text-align: center;
}

.game-jam-date strong {
    position: absolute;
    top: 0;
    padding: 0.4em 0;
    color: #fff;
    border-bottom: 1px dashed #eee;
    border-radius: 6px 6px 0 0;
}

.game-jam-date-time {
    font-size: 1em;
    position: absolute;
    margin-top: -5px;
    opacity: 0;
    transition: .2s opacity;
}

.game-jam-date:nth-child(1) strong {
    background-color: #007bff;
    box-shadow: 0 2px 0 #007bff;
}

.game-jam-date:nth-child(2) strong {
    background-color: #ffc107;
    box-shadow: 0 2px 0 #ffc107;
}

.game-jam-date:nth-child(3) strong {
    background-color: #dc3545;
    box-shadow: 0 2px 0 #dc3545;
}

.game-jam-date em {
    position: absolute;
    bottom: 0.3em;
    color: #337ab7;
}

```

```

}

.game-jam-date:nth-child(1) em {
    color: #0062ce;
}

.game-jam-date:nth-child(2) em {
    color: #e6b00e;
}

.game-jam-date:nth-child(3) em {
    color: #eb192e;
}

.game-jam-date span {
    width: 100%;
    font-size: 2.8em;
    letter-spacing: -0.05em;
    padding-top: 0.8em;
    color: #2f2f2f;
}

.game-jam-date:hover .game-jam-date-tooltip {
    opacity: 1;
    visibility: visible;
    top: -37px;
}

.game-jam-date-tooltip {
    font-size: 12px;
    color: #fff;
    font-family: 'Roboto Condensed', sans-serif;
    font-weight: 300;
    text-align: center;
    position: absolute;
    top: -52px;
    left: 3px;
    padding: 6px 15px;
    width: 105px;
    display: block;
    background-color: #222;
    box-shadow: 1px 1px 3px #333;
    padding: 8px 0;
    border-radius: 7px;
    transform: translateZ(0);
    opacity: 0;
    visibility: hidden;
    transition: .2s all;
    pointer-events: none;
}

.game-jam-date:nth-child(1) .game-jam-date-tooltip, .game-jam-date:nth-child(2)
.game-jam-date-tooltip {
    width: 80px;
    left: 16px;
}

.game-jam-date:nth-child(1) .game-jam-date-tooltip:after, .game-jam-date:nth-child(2)
.game-jam-date-tooltip:after {

```

```

    left: 37px;
}

.game-jam-date-tooltip:after {
    content: "";
    width: 0;
    height: 0;
    border-right: 7px solid transparent;
    border-left: 7px solid transparent;
    border-top: 7px solid #222;
    position: absolute;
    top: 32px;
    left: 49px;
}

/*
    generic tooltip
*/

.tooltip-hover-trigger {
    position: relative;
}

.tooltip-hover-trigger:hover .generic-tooltip {
    opacity: 1;
    visibility: visible;
    top: -37px;
    margin: 0 50%;
    transform: translate(-50%, 0);
}

.generic-tooltip {
    font-size: 12px;
    color: #fff;
    font-family: 'Open Sans Condensed', sans-serif;
    font-weight: 300;
    text-align: center;
    position: absolute;
    top: -52px;
    left: 3px;
    padding: 6px 15px;
    /*width: 105px;*/
    width: 80%;
    display: block;
    background-color: #222;
    box-shadow: 1px 1px 3px #333;
    padding: 8px;
    border-radius: 7px;
    transform: translateZ(0);
    opacity: 0;
    visibility: hidden;
    transition: .2s all;
    pointer-events: none;
    margin: 0 50%;
    transform: translate(-50%, 0);
}

.generic-tooltip:after {
    content: "";

```

```

    width: 0;
    height: 0;
    border-right: 7px solid transparent;
    border-left: 7px solid transparent;
    border-top: 7px solid #222;
    position: absolute;
    top: 32px;
    left: 49px;
}

/*
    /generic tooltip
*/

.nav-tabs {
    font-size: 18px;
}

.nav-tabs li a {
    outline: none;
}

.tab-content {
    padding: 20px;
}

.game-jam-description, .game-criteria {
    padding: 35px 0 15px;
}

.game-criteria ul {
    list-style-type: none;
}

.game-criteria ul li {
    background-color: #337ab7;
    display: inline;
    padding: 5px 16px;
    font-size: 12px;
    margin-right: 2px;
    font-weight: 700;
    line-height: 1;
    color: #fff;
    text-align: center;
    white-space: nowrap;
    vertical-align: baseline;
    border-radius: 3px;
}

/* Modal */

.modal-info {
    display: none;
    opacity: 0;
    background-color: rgba(0, 0, 0, 0.3);
    height: 100%;
    width: 100%;
    position: fixed;
    z-index: 100000;
}

```

```

padding: 0;
margin: 0;
top: 0;
left: 0;
border-radius: 5px;
}

.modal-info-inner {
width: 360px;
position: relative;
top: 50%;
left: 50%;
background-color: #fff;
transform: translate(-50%, -50%);
box-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);
}

.modal-info-header {
background-color: #222;
height: 30px;
}

.modal-info-content {
padding: 45px 15px 60px;
text-align: center;
}

.modal-close {
width: 40px;
height: 30px;
float: right;
text-align: center;
cursor: pointer;
transition: .1s background-color;
}

.modal-close:hover {
background-color: #e81123;
}

.modal-close i {
color: #fff;
font-size: 22px;
padding-top: 4px;
}

.modal-confirm {
display: none;
position: absolute;
bottom: 5px;
right: 9px;
}

.modal-confirm a {
padding-left: 20px;
padding-right: 20px;
}

/* ----- */

```

```

/*
    Game submission
*/

.gs-min-height-100 {
    min-height: 100px;
}

.carousel-wrap {
    margin: 20px auto;
    padding: 0 5%;
    width: 80%;
    position: relative;
}

/* fix blank or flashing items on carousel */
.owl-carousel .item {
    position: relative;
    z-index: 100;
    -webkit-backface-visibility: hidden;
}

/* end fix */
.owl-nav > div {
    margin-top: -26px;
    position: absolute;
    top: 50%;
    color: #cdcbcd;
}

.owl-nav i {
    font-size: 52px;
}

.owl-nav .owl-prev {
    left: -30px;
}

.owl-nav .owl-next {
    right: -30px;
}

/*****/
.remove-badge-a {
    position: absolute;
    top: 0;
    right: 0;
}

.relative-badge {
    position: relative;
}

.vis-item.chart-game-jam-bar {
    cursor: pointer;
}

.vis-item.chart-game-jam-bar:hover span {

```

```

        display: inline;
    }

    .vis-item.chart-game-jam-bar span {
        display: none;
    }

    .vis-item.chart-game-jam-bar-red {
        background-color: #ffdddd;
    }

    .download-row-game-submission{
        border: 1px solid #ddd;
        border-radius: 10px;
        background-color: #eee;
        margin: 0 40px 0 20px;
    }


    .panel-heading .accordion-toggle:after {
        /* symbol for "opening" panels */
        font-family: 'Glyphicons Halflings'; /* essential for enabling glyphicon */
        content: "\e114"; /* adjust as needed, taken from bootstrap.css */
        float: right; /* adjust as needed */
        color: grey; /* adjust as needed */
    }
    .panel-heading .accordion-toggle.collapsed:after {
        /* symbol for "collapsed" panels */
        content: "\e080"; /* adjust as needed, taken from bootstrap.css */
    }

```


9. Literatura

- Jon Duckett, Web Design with HTML, CSS, JavaScript and jQuery Set 1st Edition, 2014
- Larry Ullman, PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide (5th Edition) 5th Edition, 2017
- Riwanto Megsinarso, Step By Step Bootstrap 3: A Quick Guide to Responsive Web Development Using Bootstrap 3, 2014
- Matt Stauffer, Laravel: Up and Running: A Framework for Building Modern PHP Apps 1st Edition, 2016