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

Матија Ускоковић

**Примена AWS IAM и Cognito сервиса за контролу приступа**

ДИПЛОМСКИ РАД

- Основне академске студије -

Нови Сад, 2022.

|  |  |  |
| --- | --- | --- |
|  | Универзитет у Новом Сад, **Факултет техничких наука**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА** | Лист/Листова: |
|  |

*(Податке уноси предметни наставник - ментор)*

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

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | **Матија Ускоковић** | Број индекса: | **RA 76/2018** |
| Област: | **Безбедност у системима електронског пословања** | | |
| Ментор: | **проф. др Горан Сладић** | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:**

|  |
| --- |
| **Примена AWS IAM и Cognito сервиса за контролу приступа** |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
| Упознати се са AWS платформом са посебним освртом на AWS IAM и Cognito сервисе. Специфицирати и имплементирати веб апликацију базирану на AWS платформи у оквиру које ће контрола приступа бити реализована помоћу IAM и Cognito сервиса. Документовати решење. |

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

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

**САДРЖАЈ**

[1. УВОД 7](#_Toc113407017)

[2. Примена Амазонових сервиса 9](#_Toc113407018)

[2.1. Коришћени AWS сервиси 9](#_Toc113407019)

[2.1.1. Identity and Access Management (IAM) 9](#_Toc113407020)

[2.1.1.1. IAM за управљање корисницима и групама AWS налога 9](#_Toc113407021)

[2.1.1.2. IAM улоге и политике 10](#_Toc113407022)

[2.1.2. Амазон Cognito 10](#_Toc113407023)

[2.1.2.1. Атрибути корисника 12](#_Toc113407024)

[2.1.2.2. Подешавање лозинке 12](#_Toc113407025)

[2.1.2.3. Окидачи 13](#_Toc113407026)

[2.1.2.4. Клијентске апликације 13](#_Toc113407027)

[2.1.3. Lambda 14](#_Toc113407028)

[2.1.4. DynamoDB 15](#_Toc113407029)

[2.1.5. API Gateway 16](#_Toc113407030)

[2.1.6. CloudWatch 16](#_Toc113407031)

[3. Повезивање апликације са Амазоновим сервисима 17](#_Toc113407032)

[3.1. Имплементација аутентификације 17](#_Toc113407033)

[3.2. Имплементација ауторизације 19](#_Toc113407034)

[4. Апликација за организацију догађаја 21](#_Toc113407035)

[4.1. Нерегистровани корисник 21](#_Toc113407036)

[4.1.1. Регистрација корисника 24](#_Toc113407037)

[4.2. Обичан корисник 29](#_Toc113407038)

[4.3. Организатор догађаја 32](#_Toc113407039)

[4.4. Уређивач текстова 35](#_Toc113407040)

[4.5. Администратор 37](#_Toc113407041)

[5. ЗАКЉУЧАК 41](#_Toc113407042)

[ЛИТЕРАТУРА 43](#_Toc113407043)

[БИОГРАФИЈА 47](#_Toc113407044)

[КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА 49](#_Toc113407045)

[KEY WORDS DOCUMENTATION 51](#_Toc113407046)

# УВОД

Тема овог рада је коришћење Амазонових сервиса IAM (Identity and Access Management) [1] и Cognito [2] за аутентификацију и ауторизацију корисника као и имплементација тих сервиса кроз *web* апликацију за организацију догађаја. Апликација за организацију догађаја подржава различите улоге корисника где свака улога има посебан скуп функционалности доступан за ту улогу. Аутентификација је утврђивање идентитета корисника систему, а ауторизација је утврђивање права које идентификовани корисник има приликом коришћења система [3].

У другом поглављу ће бити описани сви Амазонови сервиси и технологије које су се користиле како би се изградила апликација за организацију догађаја. Посебан акценат ће бити на Амазоновим сервисима IAM и Cognito.

У трећем поглављу ће бити објашњен начин функционисања аутентификације и ауторизације код апликације за организацију догађаја. Такође, биће представљен ток комуникације апликације за организацију догађаја са Амазоновим сервисима, комуникација између сервиса као и управљање Амазоновим сервисима кроз AWS Менаџмент конзолу [4].

У четвртом поглављу ће *web* апликација за организацију догађаја бити детаљно разрађена. Биће описани корисници апликације, њихове улоге које имају у систему и могућност употребе апликације у зависности од улоге. Биће приказане слике са изгледом апликације и различитих форми које попуњавају корисници при њеном коришћењу, као и програмски кодови.

# Примена Амазонових сервиса

Апликација за организацију догађаја аутентификацију и ауторизацију корисника имплементира преко Амазон Cognito сервиса. Како би то било могуће неопходно је прво конфигурисати Амазон Cognito и прилагодити га одређеној апликацији, у овом случају апликацији за организацију догађаја. Да би се започело коришћење неког од сервиса неопходно је креирати AWS (Amazon Web Services) налог након чега се добија приступ AWS сервисима [5].

## Коришћени AWS сервиси

Поред споменутих сервиса за аутентификацију и ауторизацију, апликација је креирана и уз помоћ неких од осталих AWS сервиса, а то су: Lambda, DynamoDB, API Gateway i CloudWatch. Као што је имплементирана аутентификација и ауторизација на нивоу апликације за организацију догађаја, тако је исто имплементирана и у оквиру AWS-а како би се дефинисало који корисник у оквиру AWS налога сме да управља сервисима, креира и издаје дозволе за међусобно коришћење и комуникацију сервиса.

### 2.1.1. Identity and Access Management (IAM)

Овај сервис је један од основних сервиса који служи за управљање осталим сервисима из колекције AWS сервиса, као и управљању корисницима у оквиру AWS налога. Преко њега се дефинишу дозволе (енг. permissions) [6] које омогућавају специфицирање приступа AWS сервисима и ресурсима. Он се састоји од 4 компоненте, а то су: корисници [7], групе [8], улоге [9] и политике [10]. Дозволе је могуће присвајати корисницима, групама и улогама. Сваки ентитет односно ресурс у оквиру AWS-а (корисници, групе, улоге, инстанце сервиса итд.) има свој јединствени идентификатор односно ARN (Amazon Resource Name) [11] који служи да се уз помоћ тога тачно дефинишу дозволе за приступање једног ресурса другом [1].

2.1.1.1. IAM за управљање корисницима и групама AWS налога

Један AWS налог може имати произвољан број AWS корисника (где корисника може представљати нека особа или апликација) којима се могу давати различите дозволе за приступање различитим Амазоновим сервисима. Уз помоћ IAM сервиса се могу креирати ти корисници и могуће је сваком од њих дати посебна права приступа сервисима. Под правима приступа се подразумевају приступи за управљањем и коришћењем сервиса [7].

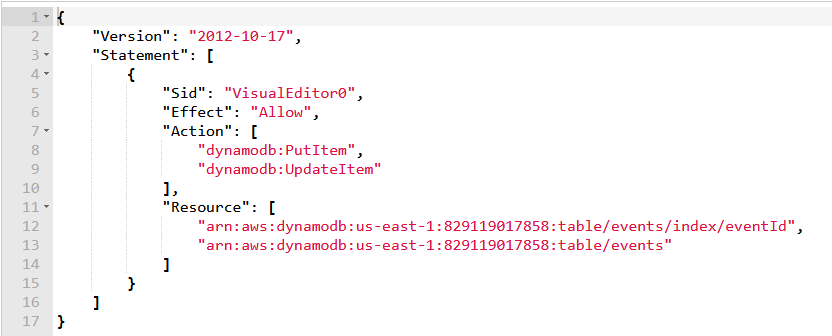
Групе се могу користити за груписање корисника где ће сви корисници који припадају одређеној групи имати заједничке дозволе и политике за приступ Амазоновим сервисима и ресурсима [8].

2.1.1.2. IAM улоге и политике

IAM улоге су IAM ентитети који су слични IAM корисницима и оне одређују шта неки ентитет може, а шта не може да ради у оквиру AWS-а. Међутим, уместо да улога буде јединствено повезана са једном особом, улога треба да буде додељена свакој особи којој је потребна. Такође, улога нема стандардне дугорочне креденцијале као што су лозинка или приступни кључеви повезани са њом. Уместо тога, када неки корисник преузме улогу, она му даје привремене безбедносне креденцијале за његову сесију улоге. Улоге могу да се користе како би се делегирао приступ корисницима, апликацијама или сервисима који обично немају приступ AWS ресурсима. Улога представља скуп дефинисаних политика [9].

IAM политике су IAM ентитети који представљају скуп дозвола где су дефинисане могуће опције приступа одређеном AWS ресурсу односно ентитету. Политике се могу делегирати улогама, али и директно корисницима, групама и ресурсима [10].

Слика 2.1. приказује пример једне политике којом је тачно специфицирано које су операције дозвољене и над којим AWS ресурсом уз помоћ ARN-а.



Слика 2.1. Пример једне AWS политике

### Амазон Cognito

Амазон Cognito је сервис који омогућава креирање корисничких скупова (енг. user pools) [12]. Кориснички скуп је основна компонента Амазон Cognito сервиса и он представља базу корисника неке конкретне апликације. Важно је напоменути да се ови корисници разликују од корисника AWS налога. На кориснике AWS налога може се гледати као на организацију која се бави производњом софтверских решења и апликација, а корисници у оквиру корисничких скупова су корисници тих конкретних апликација. Тако је за потребе и кориснике апликације за организацију догађаја креиран кориснички скуп [7, 12].

Кориснички скуп подржава следеће функционалности [12]:

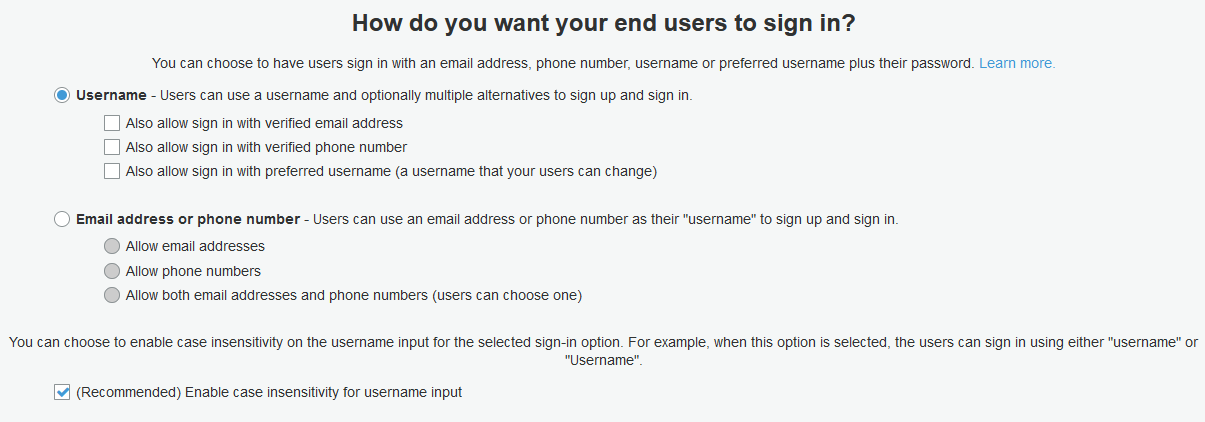
* Сервис за пријаву и регистрацију
* Унапред направљена и изменљива страница за пријаву и регистрацију корисника
* Пријава уз помоћ налога неких спољашњих система (*Facebook, Google, Amazon, Apple*)
* Управљање корисничким скупом и корисничким профилима
* Мултифакторска аутентификација (MFA) [13]
* Верификација преко телефона или преко мејла
* Произвољне процесе рада као и миграције преко *AWS Lambda triggers* (ослушкивачи одређених догађаја чијим окидањем се позива произвољна ламбда функција креирана преко AWS Lambda)

Преко корисничког скупа је могуће остварити аутентификацију и ауторизацију за неку апликацију. Самим пружањем форме за регистрацију и пријаву корисника АмазонCognitoобезбеђује аутентификацију корисника. Након успешне аутентификације корисника АмазонCognitoиздаје JSON web token (JWT) [14]. JWT је отворени стандард (*RFC 7519*) који дефинише компактан и самосталан начин за безбедно преношење информација између страна као *JSON* објекат. Ове информације могу бити верификоване и поуздане јер су дигитално потписане [15]. Уз помоћ издатог *JSON web token-a* могуће је имплементирати и ауторизацију над API-јем (Application programming interface) [16] неке апликације (у овом конкретном случају апликације за организацију догађаја).

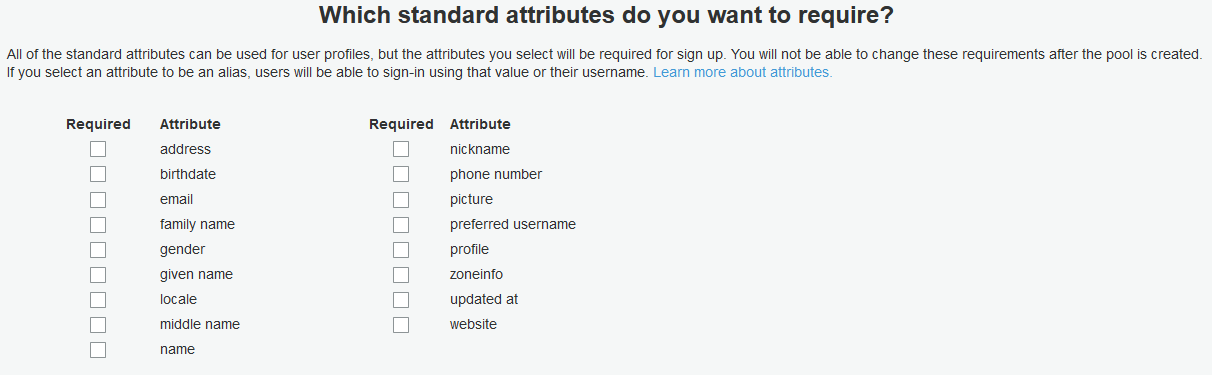
Функционалност Амазон Cognitoсервиса и корисничког скупа која омогућава имплементацију ауторизације је сврставање корисника у групе где се један корисник може налазити у више група. Информација о групама којима неки корисник припада је садржана у JWT-у што омогућава да се у API-у провери група којој корисник припада и на основу тога одреди да ли корисник има право да изврши функционалност дефинисану у наведеном API-у. Овакав модел аутентификације је RBAC (Role Based Access Control) [17] где свака Cognitoгрупа представља неку улогу и на основу те улоге може да се утврди да ли има право да затражену функционалност изврши или не.

2.1.2.1. Атрибути корисника

Приликом креирања корисничког скупа одређује се начин за регистрацију и пријаву корисника, као и стандардни атрибути корисника који су обавезни. Ова подешавања је могуће поставити само приликом креирања корисничких скупова и касније их није могуће мењати. На слици 2.2. су приказана понуђена подешавања за начин регистрације и пријаве корисника, а на слици 2.3. су приказани понуђени атрибути корисника где се бирају они који ће бити неопходни приликом регистрације на систем. Поред понуђених стандардних атрибута могуће је додавати и произвољне атрибуте [18].



Слика 2.2. Понуђена подешавања за начин регистрације и пријаве корисника



Слика 2.3. Понуђени стандардни атрибути корисника

2.1.2.2. Подешавање лозинке

Приликом креирања, а и након креирања корисничког скупа је могуће подешавање јачине лозинке. Атрибути лозинке које је могуће подешавати су:

* Минимална дужина лозинке
* Обавезни бројеви
* Обавезни специјални карактери
* Обавезна велика слова
* Обавезна мала слова

Амазоново препоручено подешавање лозинке је лозинка са: минималним бројем карактера 8, обавезна мала и велика слова, обавезни бројеви и специјални карактери [19].

2.1.2.3. Окидачи

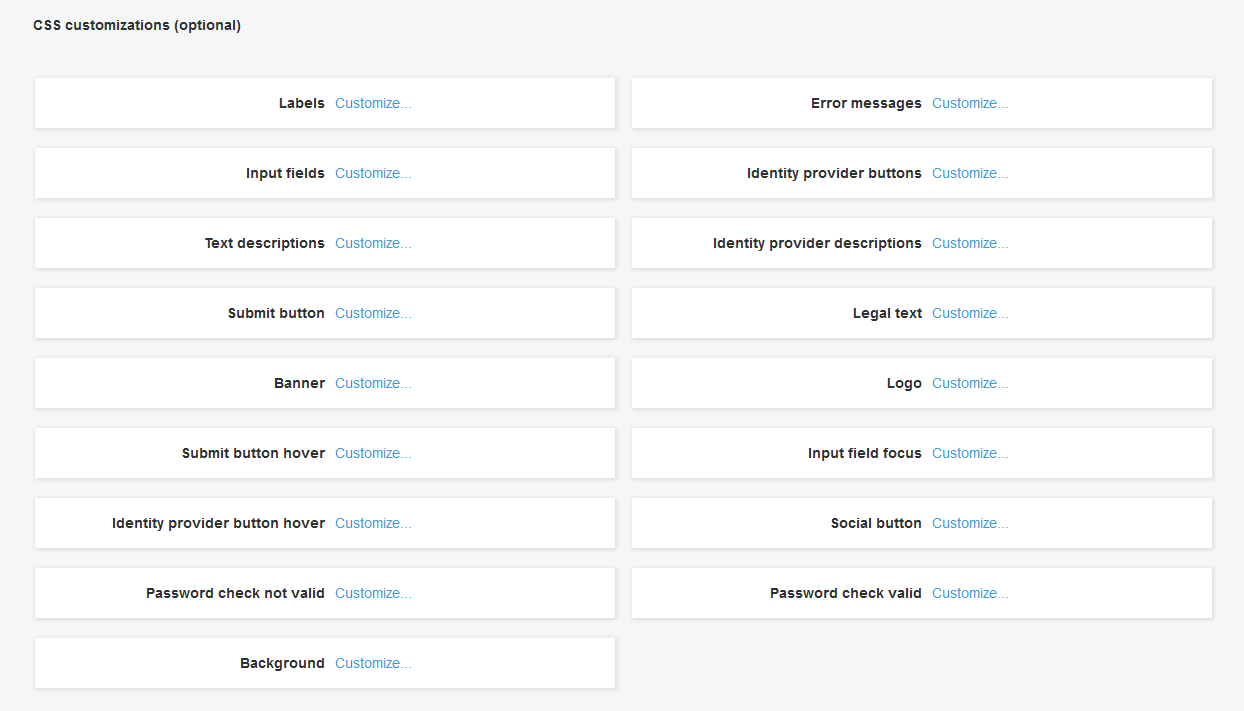
Окидачи су опција корисничког скупа који омогућавају да се Cognito повеже са одређеним Lambda функцијама. Постоје унапред одређени окидачи за које се дефинише покретање задате Lambda функције. Окидачи за које је могуће задати Lambda функцију су [20]:

* **Пред-регистрација** – окидач се покреће када корисник поднесе захтев за регистрацију
* **Пред-аутентификација** – окидач се покреће када корисник поднесе информације за аутентификацију
* **Прилагођена порука** – окидач се покреће пре слања верификационе или мултифакторске аутентификационе поруке
* **Пост-аутентификација** – окидач се покреће након што се корисник аутентификује
* **Пост-конфирмација** – окидач се покреће након што се корисник потврди
* **Дефинисање изазова за аутентификацију** – окидач се позива да би се покренуо прилагођени ток аутентификације
* **Креирање изазова за аутентификацију** – окидач се позива након „Дефинисање изазова за аутентификацију“ уколико је за тај окидач специфицирана Lambda функција
* **Верификација одговора на изазов за аутентификацију** – окидач се покреће да се верификује да ли је одговор од корисника за прилагођену аутентификацију валидан
* **Миграција корисника** – окидач се окида током операција пријаве и „заборављена-лозинка“
* **Пред генерисање токена** – окидач се окида пре него што се генерише токен

2.1.2.4. Клијентске апликације

У оквиру Cognito сервиса могуће је креирати клијентске апликације које ће користити одређени кориснички скуп, што значи да један кориснички скуп могу користити више различитих апликација. За сваку креирану клијентску апликацију у оквиру корисничког скупа се генерише идентификатор уз помоћ којег се та клијентска апликација јединствено идентификује корисничком скупу. Такође, за сваку клијентску апликацију се креира посебан домен за интерфејс за пријаву и регистрацију [21].

Изглед форме за пријаву и регистрацију је могуће мењати али је та измена ограничена. Могуће је додати лого слику или мењати CSS (Cascading Style Sheets) [22] одређених компоненти приказа. На слици 2.4. су приказане компоненте које је могуће мењати кроз CSS.



Слика 2.4. Компоненте приказа за пријаву и регистрацију које је могуће мењати кроз CSS

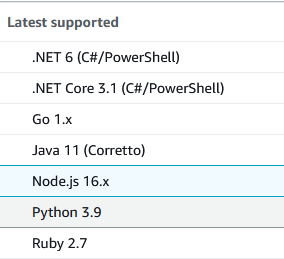
### 2.1.3. Lambda

AWS Lambda је сервис за креирање функција чији се код покреће без обезбеђивања или управљања серверима [23]. Функције је могуће покренути коришћењем Lambda API-а или их Lambda може покретати као одговор на догађаје из осталих AWS сервиса.

Кључне карактеристике Lambde [23]:

* Контрола паралелности и скалирања [24]
* Функције дефинисане као слике контејнера [25]
* Потписивање кода [26] – пружа поверљивост и интегритет за код који је примењен у Lambda функцијама
* Lambda екстензије [27] – екстензије за лакше интегрисање са различитим алатима за надгледање, видљивост, безбедност и управљање
* Нацрти функција – пружају примере кодова који показују како се Lambda користи са другим AWS сервисима или апликацијама трећих страна
* Приступ бази података
* Приступ фајл систему

На слици 2.5. су приказане најновије верзије програмских језика које тренутно подржава AWS Lambda.



Слика 2.5. Најновије верзије које тренутно подржава AWS Lambda (27.08.2022.)

За потребе апликације за организацију догађаја коришћене су Lambda функције писане у Node.js 16.x и Python 3.9 програмским језицима.

### 2.1.4. DynamoDB

DynamoDB је Амазон сервис NoSQL (Not SQL) базе података. NoSQL базе података су наменски направљене за специфичне моделе података и имају флексибилне шеме за иградњу модерних апликација. Постоји више типова NoSQL база података а то су: кључ-вредност оријентисана база података, база података оријентисана по колонама, документ-оријентисана и граф оријентисана база података. DynamoDB је кључ-вредност оријентисана база података што значи да су подаци смештени у кључ-вредност паровима. Овај сервис омогућава креирање табела где је свака специфично намењена за неки модел података [28].

### 2.1.5. API Gateway

API Gateway је Амазонов сервис који омогућава креирање различитих типова API-а за апликације. Конкретно је за апликацију за организацију догађаја креиран REST (Representational State Transfer) API [29]. REST API Gateway представља колекцију ресурса и метода које су повезане са Lambda функцијама.

Уз помоћ API Gateway-a је омогућен приступ функционалностима које нуди апликација за организацију догађаја. Он дефинише URL-ове (Uniform Resource Locator) [30] који представљају адресе уз помоћ којих се јединствено одређује неки ресурс на мрежи, у овом случају ресурси апликације за организацију догађаја.

Поред тога што се у оквиру API Gateway-a повезују Lambda функције, такође га је могуће повезати и са Амазон Cognito сервисом. Повезивањем са Амазон Cognito сервисом остварује се могућност аутентификације корисника који покушавају да добаве ресурсе преко траженог URL-a. Аутентификација се спроводи тако што се у захтеву који се прослеђује на API Gateway проследи и корисников JWT чију валидност затим проверава Cognito сервис и уколико је JWT валидан захтев се прослеђује до траженог ресурса [31].

### 2.1.6. CloudWatch

CloudWatch је сервис за надгледање и праћење рада Амазонових сервиса, као и апликација које се налазе на AWS-u. CloudWatch се може користити за прикупљање и праћење метрика, а такође се могу и креирати различити аларми који прате одређену метрику и шаљу обавештења или аутоматски уносе измене у ресурсе које надгледају [32].

За развој апликације за организацију догађаја овај сервис је омогућио праћење извршавања Lambda функција и увид у грешке које су се дешавале током развоја што је омогућило бржи и несметани развој апликације.

# Повезивање апликације са Амазоновим сервисима

Како би се успешно имплементирала аутентификација и ауторизација користећи Амазонове сервисе неопходно их је међусобно повезати. Након што се сервиси међусобно повежу потребно је повезати и те сервисе са апликацијом, у овом случају апликацијом за управљање догађаја.

Прво је неопходно имплементирати аутентификацију, а након тога и ауторизацију.

## Имплементација аутентификације

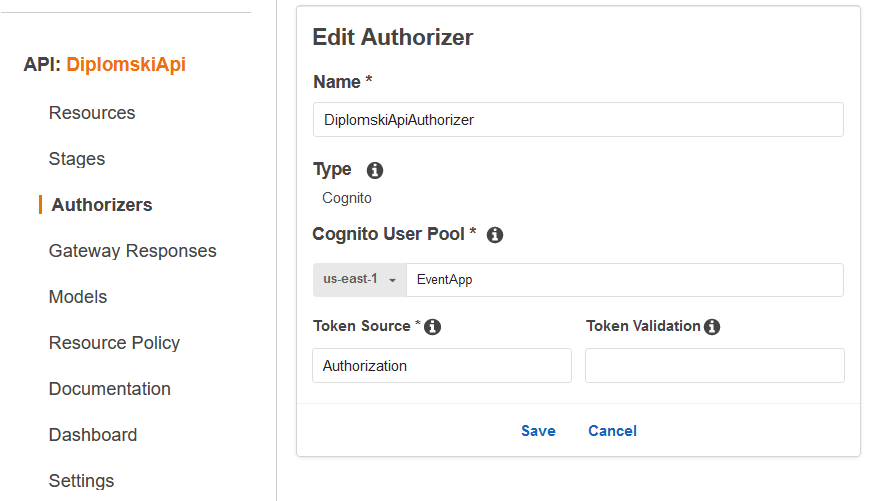
Главна компонента за реализацију аутентификације је Cognito сервис. Када се корисник успешно пријави на апликацију Cognito сервис креира сесију и враћа идентификациони токен, приступни токен и освежавајући токен.

**Идентификациони токен** је JWT који садржи тврдње о идентитету аутентификованог корисника као што су: име, адреса, мејл адреса и број телефона. Те информације о идентитету се могу користити унутар апликације [33].

**Приступни токен** је JWT који садржи тврдње о аутентификованом кориснику, листу корисничких група и листу опсега. Сврха токена за приступ је да ауторизује API операције у контексту корисника у оквиру корисничког скупа [34].

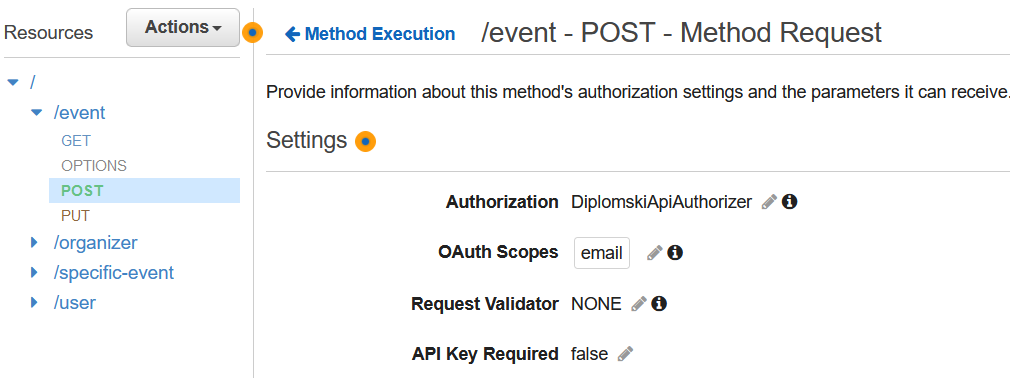
**Освежавајући токен** је токен који се користи за добијање новог идентификационог или приступног токена [35]. Докле год неки корисник у оквиру апликације има важећи освежавајући токен који није истекао њиме може да добија нове идентификационе и приступне токене без да се поново улогује на апликацију.

Да би аутентификација била успешно реализована неопходно је Cognito сервис повезати са API Gateway-ом. Повезивање се врши из конзоле за управљање API Gateway-ом који садржи опцију за креирање овлашћења (енг. Authorizers) [36]. Овлашћења служе за контролу приступа API-у и идентификацију позиваоца односно аутентификацију корисника. На слици 3.1. је приказан пример једног овлашћења.



Слика 3.1. Пример једног овлашћења

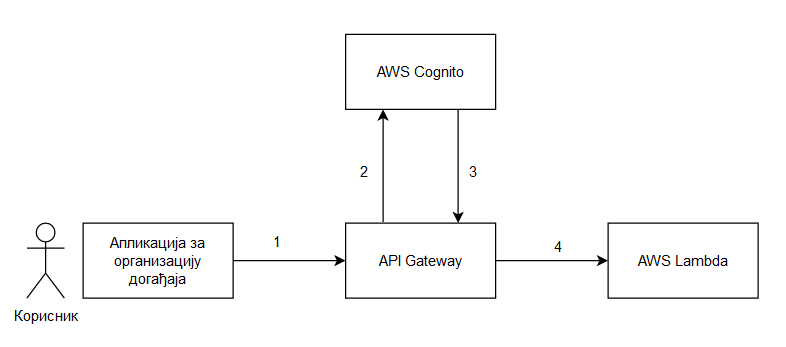
Како би се креирало овлашћење неопходно је унети његов назив, тип овлашћења, кориснички скуп за који ће овлашћење важити, извор токена који представља назив заглавља уз који ће се слати JWT и валидацију токена у виду регуларног израза. Након што се креира одговарајуће овлашћење оно се може примењивати на жељене ресурсе и методе које су имплементиране у оквиру API Gateway-а. На слици 3.2. је приказано подешавање овлашћења над методом за креирање догађаја чиме је постигнута аутентификација корисника.



Слика 3.2. Пример примене овлашћења над методом за креирање догађаја

У подешавању овлашћења неопходно је навести овлашћење које се користи. Додатно је могуће навести листу опсега које је неопходно да садржи прослеђени JWT како би аутентификација била успешна. Такође могуће је навести валидатор HTTP (HyperText Transfer Protocol) [37] захтева да садржи одговарајуће параметре и одговарајуће тело захтева.

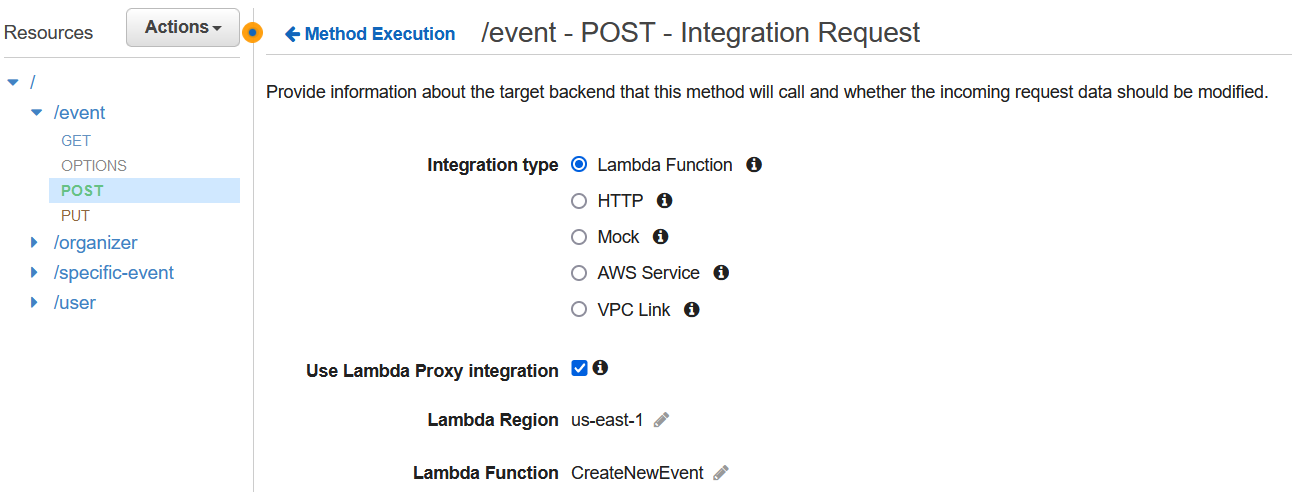
Након што се корисник успешно аутентификује позив се пропушта до одговарајуће Lambda функције у оквиру које се даље врши ауторизација уколико је то неопходно. Пример успешног тока аутентификације корисника је приказано на слици 3.3.



Слика 3.3. Пример успешног тока аутентификације корисника апликације за организацију догађаја

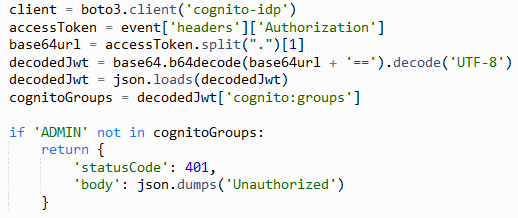
## Имплементација ауторизације

Ауторизација је имплементирана уз помоћ Cognito група. Сваки пут када улоговани корисник упути захтев ка API Gateway-у у оквиру његовог приступног токена се налази информација о групи којој корисник припада. Након што захтев корисника прође успешно аутентификацију имамо гаранцију да је његов JWT исправан и тада у оквиру Lambda функције можемо проверити да ли JWT има захтевану групу. На слици 3.4. је приказано подешавање за позивање Lambda функције када се затражи одговарајући ресурс са одговарајућом методом, где је за ауторизацију јако битно означити опцију „Use Lambda Proxy integration“. Споменута опција омогућава да након успешне аутентификације API Gateway проследи захтев са свим информацијама на Lambda функцију.



Слика 3.4. Пример подешавања повезаности ресурса API Gateway-а са Lambda функцијом

Након што Lambda функција добије информације из захтева на основу њих може утврдити да ли корисник који је упутио захтев има одговарајућу групу односно улогу да изврши тражену функционалност. На слици 3.5. је приказан пример провере администраторске улоге корисника у Python програмском језику.



Слика 3.5. Пример ауторизације у Lambda функцији

# Апликација за организацију догађаја

Апликација за огранизацију догађаја, која је тема овог рада, за складиштење корисничких информација, као и за аутентификацију и ауторизацију корисника, користи Cognito сервис, а за складиштење података о догађајима користи DynamoDB. Апликација је имплементирана у Angular развојном окружењу [38]. Са наведеним сервисима апликација комуницира преко API Gateway-а који позива одговарајуће Lambda функције како би се извршиле захтеване операције.

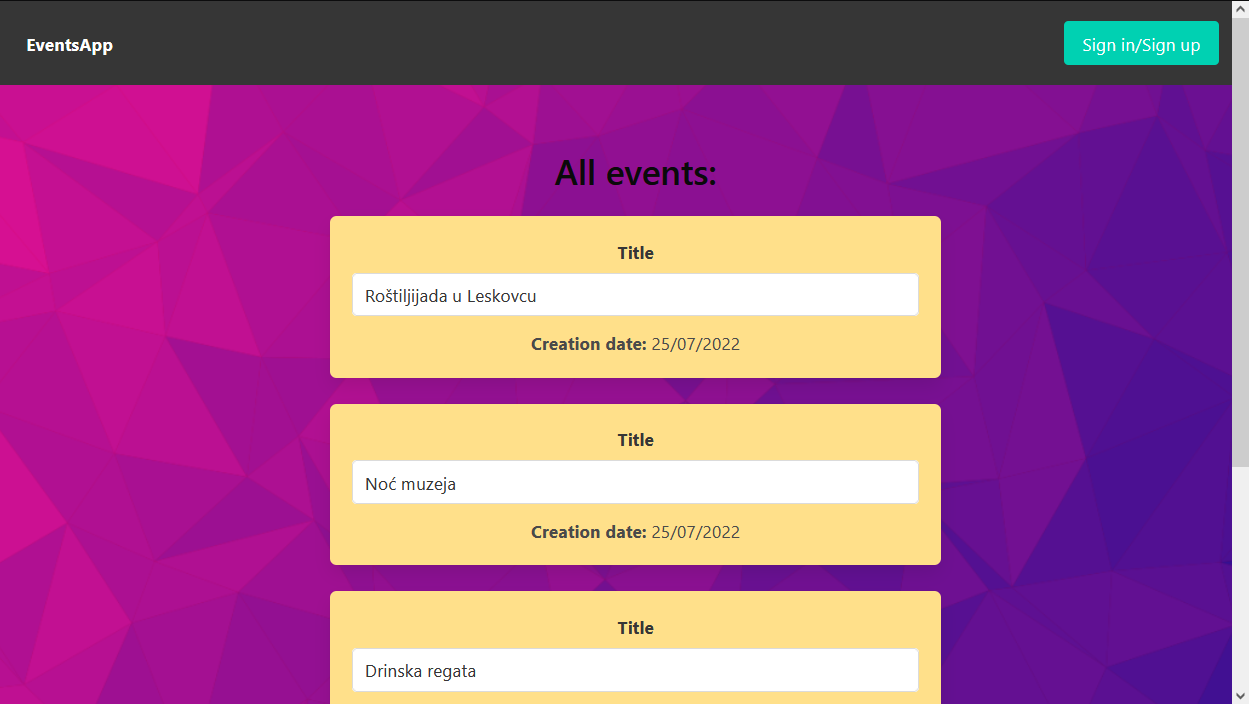
Апликација за организацију догађаја има пет врста корисника:

* Нерегистровани корисник
* Обичан корисник (Cognito група ***USER***)
* Организатор догађаја (Cognito група ***ORGANIZER***)
* Уређивач текстова (Cognito група ***COPYWRITER***)
* Администратор система (Cognito група ***ADMIN***)

## 4.1. Нерегистровани корисник

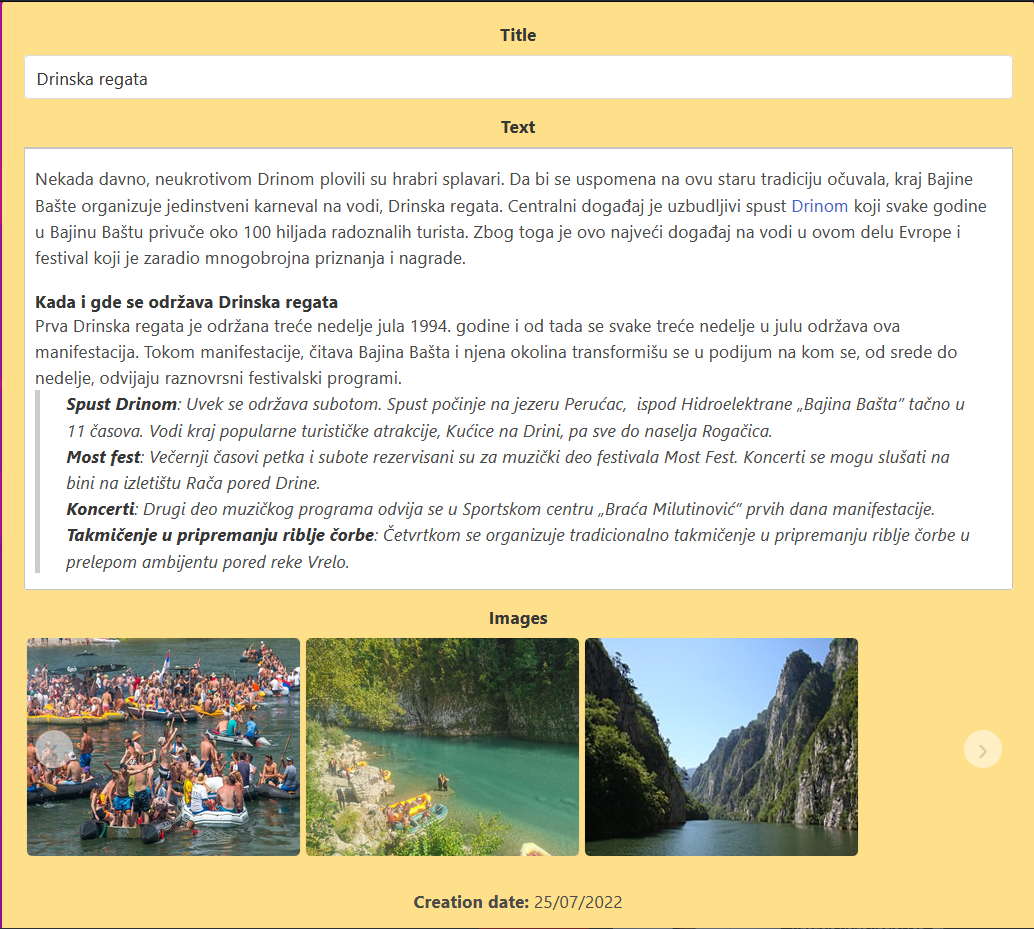
Нерегистровани корисник има увид у почетну страницу где су приказани сви креирани догађаји са основним информацијама и кликом на неки од догађаја му се отвара детаљан приказ тог догађаја.

На слици 4.1. је приказана почетна страница нерегистрованог корисника.



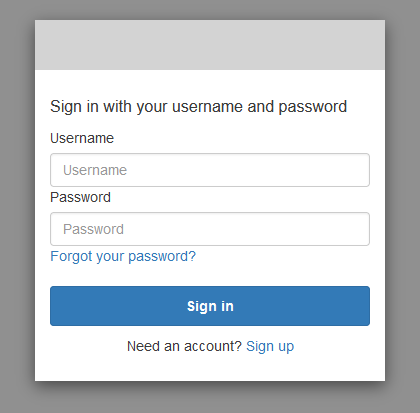
Слика 4.1. Почетна страница нерегистрованог корисника

На слици 4.2. је приказан детаљан приказ догађаја.



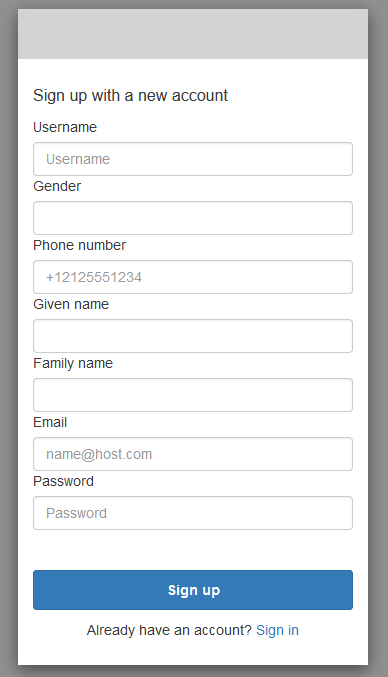
Слика 4.2. Детаљан приказ догађаја

Такође, кориснику је на почетној страници доступно и дугме за пријаву или регистрацију на систем. На слици 4.3. приказана је форма коју нуди Cognito сервис за пријаву корисника.



На слици 4.4. приказана је форма коју нуди Cognito за регистрацију корисника.

Слика 4.3. Cognito форма за пријаву корисника



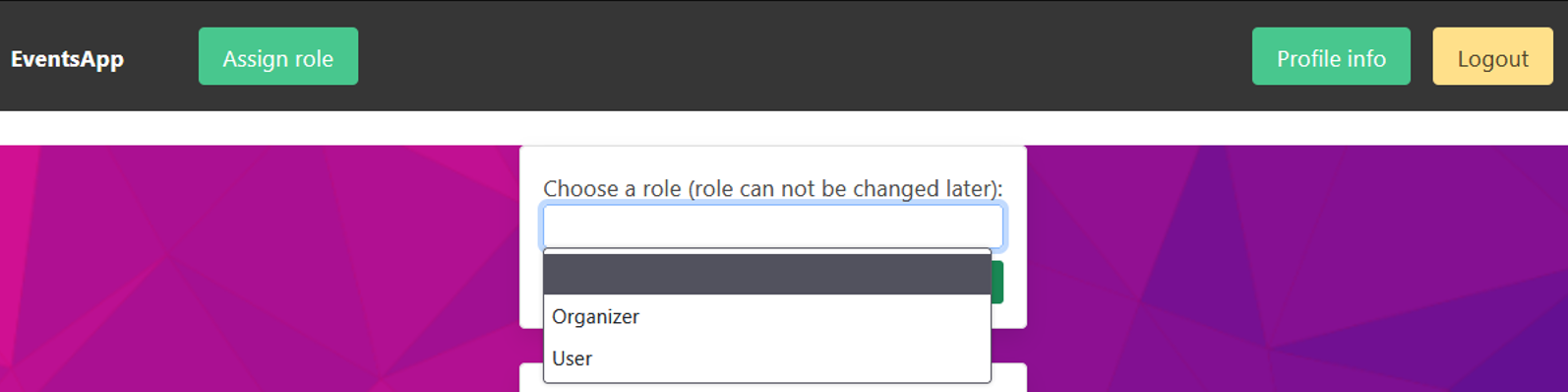
Слика 4.4. Cognito форма за регистрацију корисника

### 4.1.1. Регистрација корисника

Попуњавањем форме за регистрацију корисника и кликом на „Sign up“ (слика 4.4.) Cognito сервис шаље верификациони мејл на мејл адресу која је наведена приликом попуњавања форме. Да би корисник могао да се пријави на апликацију за организацију догађаја неопходно је да кликне на линк за верификацију који је добио путем мејла.

Након прве пријаве на систем корисник добија две нове опције а то су опција да одреди своју улогу (коју касније није могуће изменити) и приказ својих информација. Улоге које новорегистровани корисник може одабрати су: обичан корисник и организатор. Преостале две улоге, администратор и уређивач текста, су улоге које се могу поставити само кроз Cognito сервис и то су корисници који се баве управљањем апликације.

На слици 4.5. приказана је опција када корисник може да одреди своју улогу.



Слика 4.5. Опција бирања улоге у систему

Одређивањем улоге упућује се HTTP захтев на API Gateway који тај захтев прослеђује на Lambda функцију која се бави сврставањем корисника у одговарајућу Cognito групу. Код за сврставање корисника у одговарајућу Cognito групу је писан у NodeJS-u и приказан је у листингу 4.1. У коду из листинга 4.1. се на почетку добавља инстанца Cognito провајдера уз помоћ којег се позивају жељене методе за управљање Cognito сервисом, затим се добавља приступни токен из захтева уз помоћ којег је могуће одредити да ли корисник већ има додељену улогу и уколико нема послаће се захтев на Cognito сервис да му се тражена улога додели.

Листинг 4.1. Код за сврставање корисника у одговарајућу Cognito групу

var AWS = require('aws-sdk');

exports.handler = (event, context, callback) => {

var cognitoIdentityServiceProvider = new AWS.CognitoIdentityServiceProvider();

var parsedJwt = parseJwt(event.headers.Authorization);

if (alreadyAssigned(parsedJwt)) {

return badRequestResponse();

}

var userName = parsedJwt['username'];

let body = JSON.parse(event.body);

let role = body["role"];

var params = {

GroupName: role, //user gets added to this group

UserPoolId: "us-east-1\_mOGgp8jYV",

Username: userName

};

cognitoIdentityServiceProvider.adminAddUserToGroup(params, function(err, data) {

if (err) {

callback(err);

}

callback(null, event);

});

};

Сваки регистровани корисник има могућност да мења своје информације као и лозинку. За обе измене се преко Lambda функција позивају функције које нуди Cognito сервис. Те Lambda функције су писане у Python програмском језику и у листингу 4.2. је приказана Lambda функција за измену корисничких информација односно атрибута. У коду за измену корисничких атрибута се на почетку добавља инстанца Cognito провајдера, приступни токен и тело захтева са новим корисничким информацијама које ће се искористити да се упути захтев на Cognito сервис.

Листинг 4.2. Код за измену атрибута корисника

import json

import boto3

import base64

def lambda\_handler(event, context):

client = boto3.client('cognito-idp')

accessToken = event['headers']['Authorization']

base64url = accessToken.split(".")[1]

decodedJwt = base64.b64decode(base64url + '==').decode('UTF-8')

decodedJwt = json.loads(decodedJwt)

username = decodedJwt['username']

body = json.loads(event['body'])

givenName = body['givenName']

familyName = body['familyName']

phoneNumber = body['phoneNumber']

gender = body['gender']

try:

response = client.admin\_update\_user\_attributes(

UserPoolId='us-east-1\_mOGgp8jYV',

Username=username,

UserAttributes=[

{

'Name': 'given\_name',

'Value': givenName

},

{

'Name': 'family\_name',

'Value': familyName

},

{

'Name': 'phone\_number',

'Value': phoneNumber

},

{

'Name': 'gender',

'Value': gender

},

]

)

except Exception as e:

return {

'statusCode': 400,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps(str(e))

}

else:

return {

'statusCode': 200,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps('Successfuly changed attributes!')

}

У листингу 4.3. је приказана Lambda функција за измену лозинке корисника. У коду за измену лозинке се на почетку добавља инстанца Cognito провајдера, приступни токен и тело захтева са информацијама о старој и новој лозинци након чега се те информације прослеђују Cognito сервису позивом методе провајдера за измену лозинке.

Листинг 4.3. Код за измену лозинке корисника

import json

import boto3

def lambda\_handler(event, context):

client = boto3.client('cognito-idp')

accessToken = event['headers']['Authorization']

body = json.loads(event['body'])

previousPassword = body['previousPassword']

newPassword = body['newPassword']

try:

response = client.change\_password(

PreviousPassword= previousPassword,

ProposedPassword= newPassword,

AccessToken= accessToken

)

except Exception as e:

return {

'statusCode': 400,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps(str(e))

}

else:

return {

'statusCode': 200,

'headers': {

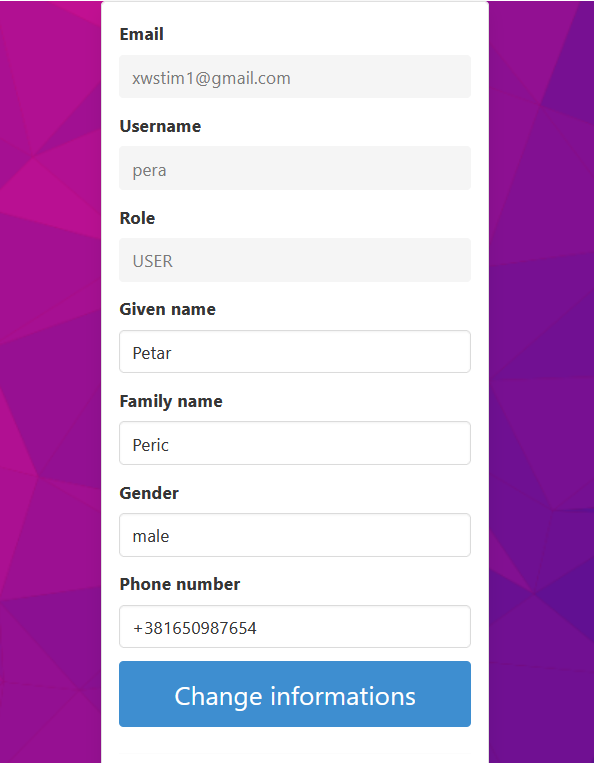
'Access-Control-Allow-Origin': '\*'

},

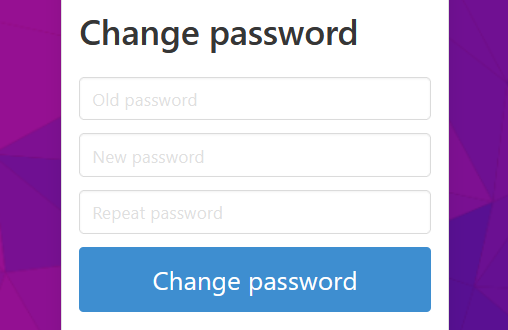
'body': json.dumps('Successfuly changed password!')

}

На слици 4.6. је приказана форма, из апликације за организацију догађаја, за измену информација корисника, а на слици 4.7. форма за измену лозинке.



Слика 4.6. Форма за измену информација корисника

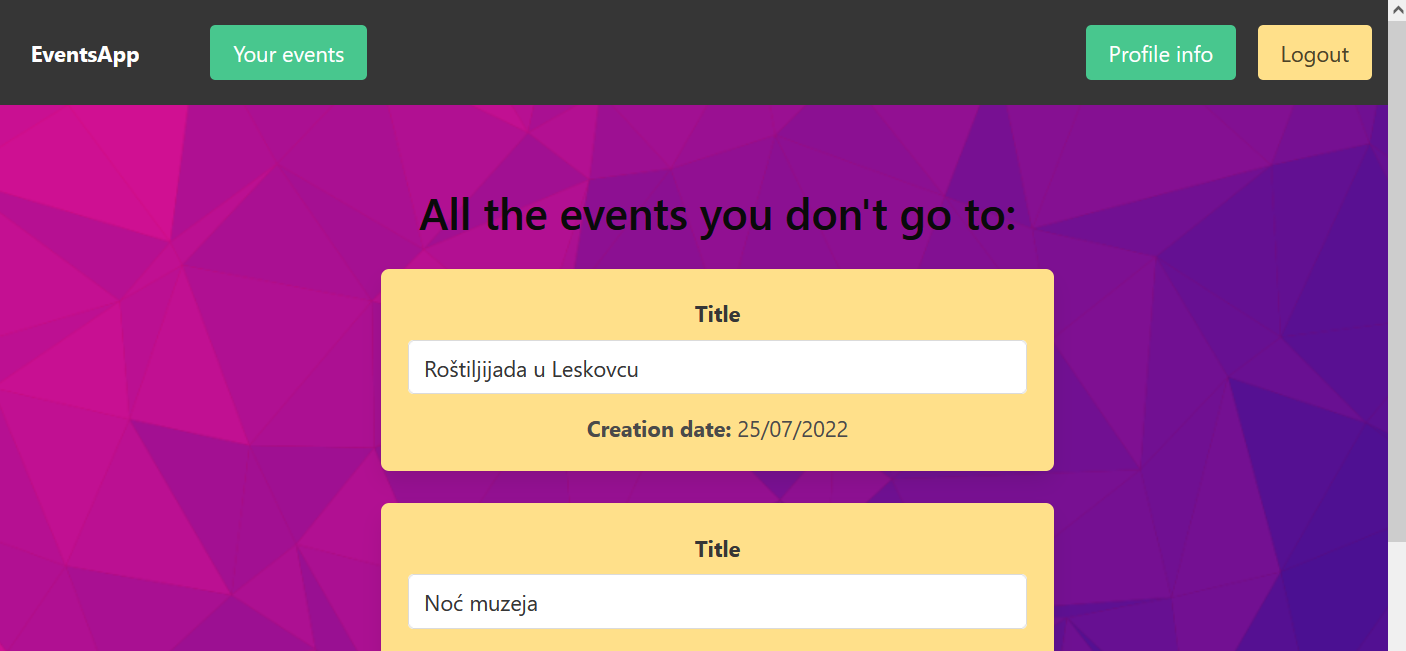


Слика 4.7. Форма за измену лозинке корисника

## Обичан корисник

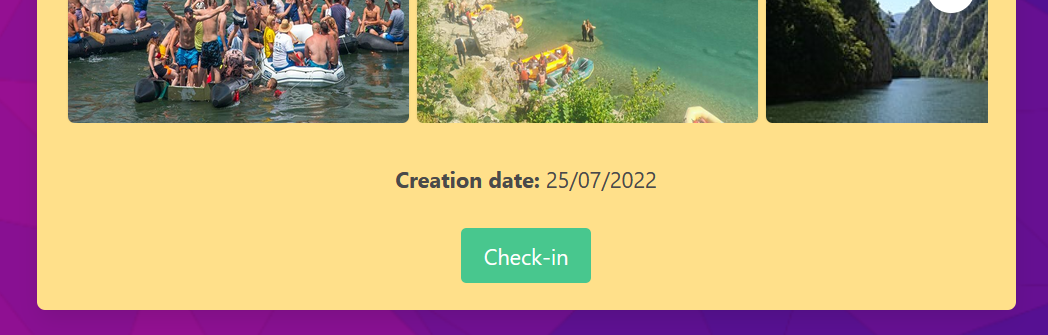
Приликом бирања улоге у систему, уколико корисник одабере „User“ постаје обичан корисник. Обичан корисник има додатну функционалност која му омогућава да се пријави на одабране догађаје, као и да се са пријављених догађаја одјави.

На почетној страници су му излистани догађаји на којима није пријављен, а кликом на дугме „Your events“ (слика 4.8.) добија приказ свих догађаја за које се пријавио.

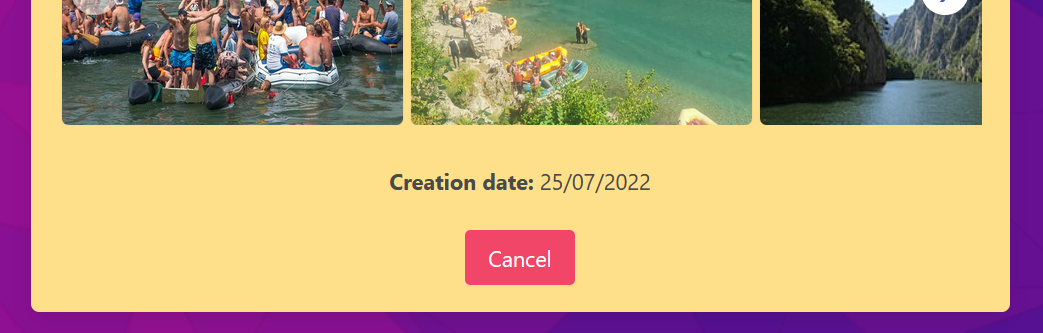


Слика 4.8. Почетна страница обичног корисника

Уколико обичан корисник уђе у детаљан приказ догађаја за који није пријављен при дну догађаја му се прикаже дугме да се за тај догађај пријави (слика 4.9.), а уколико већ јесте пријављен тада има опцију да се са тог догађаја одјави (слика 4.10.).



Слика 4.9. Опција обичног корисника да се пријави на догађај



Слика 4.10. Опција обичног корисника да се одјави са догађаја

Lambda функција која врши пријаву на одређени догађај је приказана у листингу 4.4. и писана је у NodeJS-u.

Листинг 4.4. Код за пријављивање корисника на догађај

const AWS = require('aws-sdk');

let dynamodb = new AWS.DynamoDB.DocumentClient();

exports.handler = async (event) => {

var parsedJwt = parseJwt(event.headers.Authorization);

var roles = parsedJwt['cognito:groups'];

var userId = parsedJwt['sub'];

if (!roles.includes("USER")) {

return unauthorizedResponse();

}

let body = JSON.parse(event.body);

let eventId = body["eventId"];

if (eventId == "" || eventId == undefined) {

return badRequestResponse();

}

let params = {

TableName:'eventConfirmations',

Item: {

'userId': userId,

'eventId': eventId

}

};

// Using await, make sure object writes to DynamoDB table before continuing execution

await dynamodb.put(params).promise();

return okResponse();

};

Lambda функција која служи за одјаву корисника са одређеног догађаја је приказана у листингу 4.5. и писана је у NodeJS-u.

Листинг 4.5. Код за одјављивање корисника са догађаја

const AWS = require('aws-sdk');

let dynamodb = new AWS.DynamoDB.DocumentClient();

exports.handler = async (event) => {

var parsedJwt = parseJwt(event.headers.Authorization);

var roles = parsedJwt['cognito:groups'];

var userId = parsedJwt['sub'];

if (!roles.includes("USER")) {

return unauthorizedResponse();

}

let body = JSON.parse(event.body);

let eventId = body["eventId"];

if (eventId == "" || eventId == undefined) {

return badRequestResponse();

}

let params = {

TableName:'eventConfirmations',

Key: {

'userId': userId,

'eventId': eventId

}

};

await dynamodb.delete(params).promise();

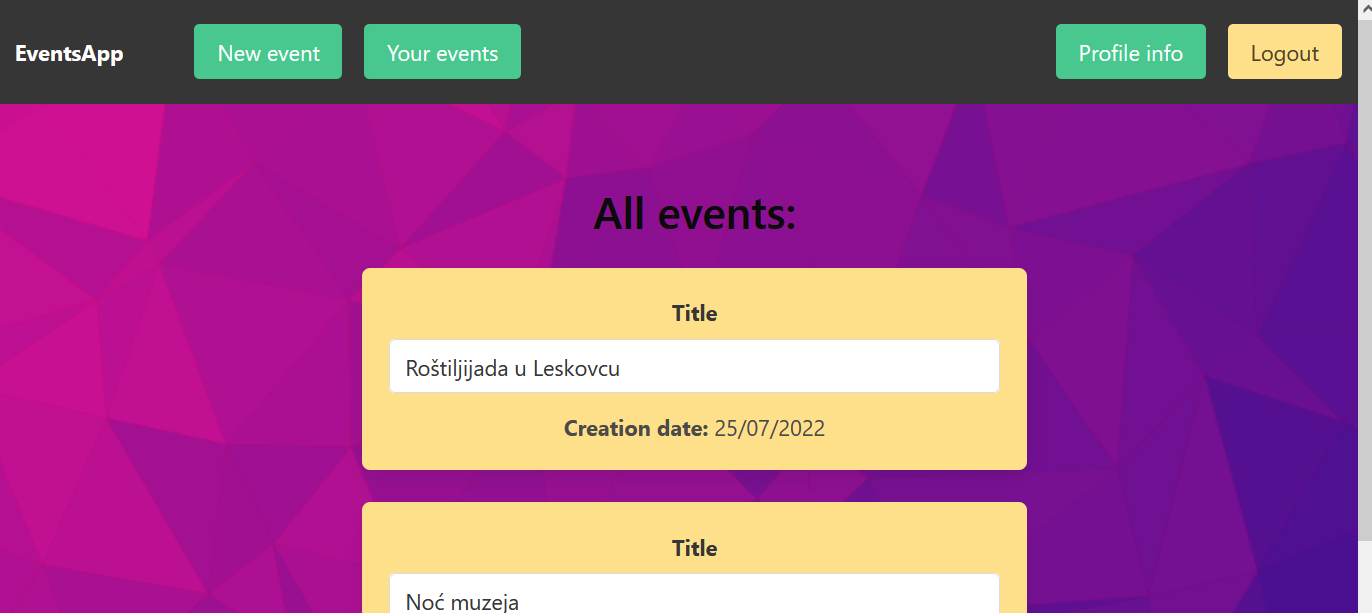
return okResponse();

};

На почетку функције из листинга 4.4. и 4.5. се добавља приступни токен из захтева из којег се даље извуче Cognito група којој корисник припада. Након добављања Cognito групе врши се ауторизација односно провера да ли корисник припада групи обичног корисника. Уколико ауторизација прође успешно даље се користе функције које нуди DynamoDB сервис где у случају пријављивања на догађај (листинг 4.5.) уписује информацију о томе да се корисник пријавио на одређени догађај, а у случају одјављивања са догађаја (листинг 4.6.) брише информацију о томе да се корисник пријавио на одређени догађај.

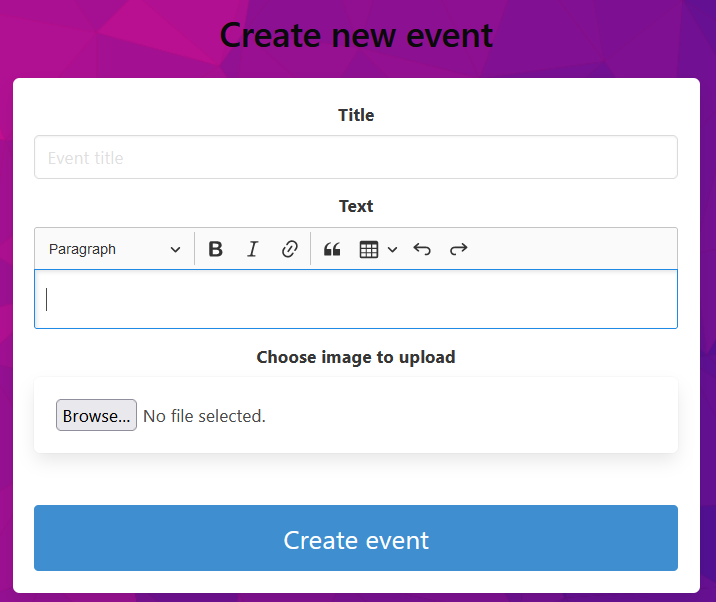
## Организатор догађаја

Приликом бирања улоге у систему, уколико корисник одабере „Organizer“ постаје организатор. Организатор је улога која креира догађаје на које обични корисници могу да се пријављују. Такође, организатор има на почетној страници приказ свих догађаја, као и опцију да прикаже догађаје које је креирао. На слици 4.11. је приказана почетна страница организатора.



Слика 4.11. Почетна страница организатора

Приликом креирања догађаја организатор има могућност да унесе наслов, текст и слике везане за догађај. Приликом писања текста могуће је користити стилизовања као што су подебљана и накривљена слова, могуће је додавање линкова, цитата и табела. Форма за креирање догађаја је приказана на слици 4.12.



Слика 4.12. Форма за креирање догађаја

Lambda функција која се бави креирањем новог догађаја је писана у NodeJS-у и приказана је у листингу 4.6. На почетку функције се добавља инстанца DynamoDB провајдера, креира се тренутно време које ће се искористити као информација за тренутак креирања догађаја и добавља се приступни токен из захтева. Након добијања приступног токена врши се ауторизација односно провера да ли корисник припада групи организатора и уколико припада добавља се тело захтева где се налазе информације о догађају које ће се искористити да се креира нови догађај у бази преко DynamoDB провајдера. Уколико корисник нема улогу организатора неће имати право да изврши дату функционалност.

Листинг 4.6. Код за креирање новог догађаја

// Include the AWS SDK module

const AWS = require('aws-sdk');

// Instantiate a DynamoDB document client with the SDK

let dynamodb = new AWS.DynamoDB.DocumentClient();

let date = new Date();

// Store date and time in human-readable format in a variable

let now = date.toISOString();

const { randomUUID } = require('crypto');

exports.handler = async (event) => {

var parsedJwt = parseJwt(event.headers.Authorization);

var roles = parsedJwt['cognito:groups'];

var userId = parsedJwt['sub'];

if (!roles.includes("ORGANIZER")) {

return unauthorizedResponse();

}

let body = JSON.parse(event.body);

let title = body["title"];

let text = body["text"];

let images = body["images"];

if (title == "" || text == "" || title == undefined || text == undefined) {

return badRequestResponse();

}

let params = {

TableName:'events',

Item: {

'eventId': randomUUID(),

'organizerId': userId,

'creationDateTime': now,

'title': title,

"text": text,

"images": images

}

};

// Using await, make sure object writes to DynamoDB table before continuing execution

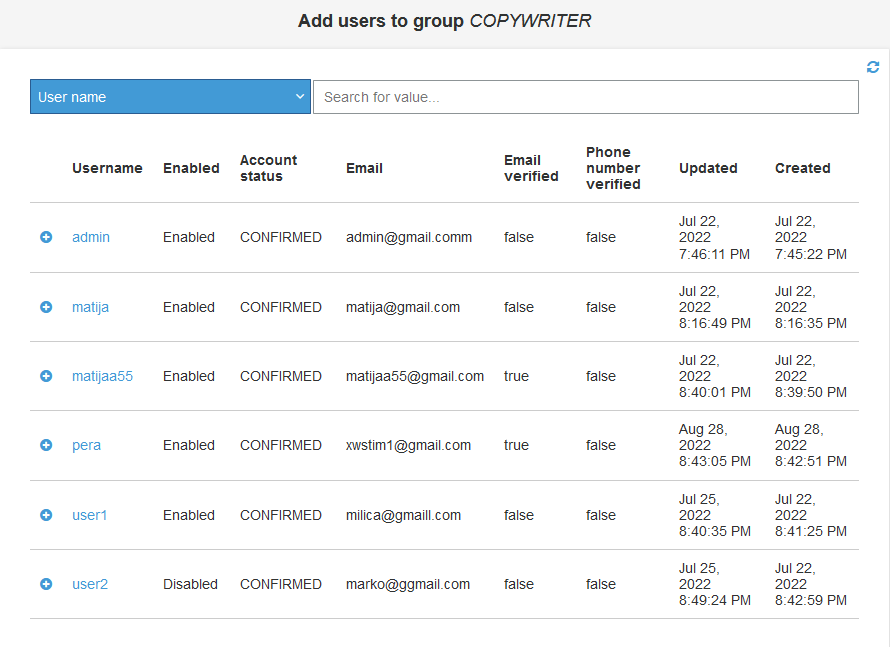
await dynamodb.put(params).promise();

return okResponse();

};

## Уређивач текстова

Улога уређивача текстова (*COPYWRITER*) је улога коју не може да бира корисник након регистрације. Ову улогу је могуће доделити само из конзоле за управљање Cognito сервисом. То је улога која, поред увида у све догађаје, има и право да мења текстове догађаја које креирају организатори, али не и да мења наслов и слике. Претпоставка је да је особа са овом улогом стручна у писању текстова тако да обичан корисник не може сам преко апликације да одабере ову улогу. Додавање корисника у одговарајућу Cognito групу, из конзоле за управљање Cognito сервисом, је могуће извршити након одабира Cognito групе где се добија приказ свих корисника који припадају тој групи као и могућност да се дода нови корисник из листе корисника који не припадају одабраној групи. На слици 4.13. је приказан изглед списка корисника који не припадају одабраној групи одакле је могуће те кориснике додати у групу.



Слика 4.13. Приказ корисника које је могуће убацити у одабрану групу

Lambda функција која се извршава када уређивач текста измени текст неког догађаја је приказана у листингу 4.7. Функција је писана у Python програмском језику. На почетку функције се добавља инстанца DynamoDB провајдера и приступни токен уз помоћ којег се врши ауторизација. Након успешне ауторизације позива се функција преко DynamoDB провајдера за измену догађаја.

import json

import boto3

import base64

def lambda\_handler(event, context):

client = boto3.client('dynamodb')

accessToken = event['headers']['Authorization']

base64url = accessToken.split(".")[1]

decodedJwt = base64.b64decode(base64url + '==').decode('UTF-8')

decodedJwt = json.loads(decodedJwt)

cognitoGroups = decodedJwt['cognito:groups']

if 'COPYWRITER' not in cognitoGroups:

return {

'statusCode': 401,

'body': json.dumps('Unauthorized')

}

body = json.loads(event['body'])

eventId = body['eventId']

text = body['text']

try:

response = client.update\_item(

TableName='events',

Key={

'eventId': { "S": eventId }

},

UpdateExpression="set #text = :text",

ExpressionAttributeNames= {

'#text': "text"

},

ExpressionAttributeValues={

':text': { "S" : text },

},

ReturnValues="UPDATED\_NEW"

)

except Exception as e:

return {

'statusCode': 400,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps(str(e))

}

else:

return {

'statusCode': 200,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

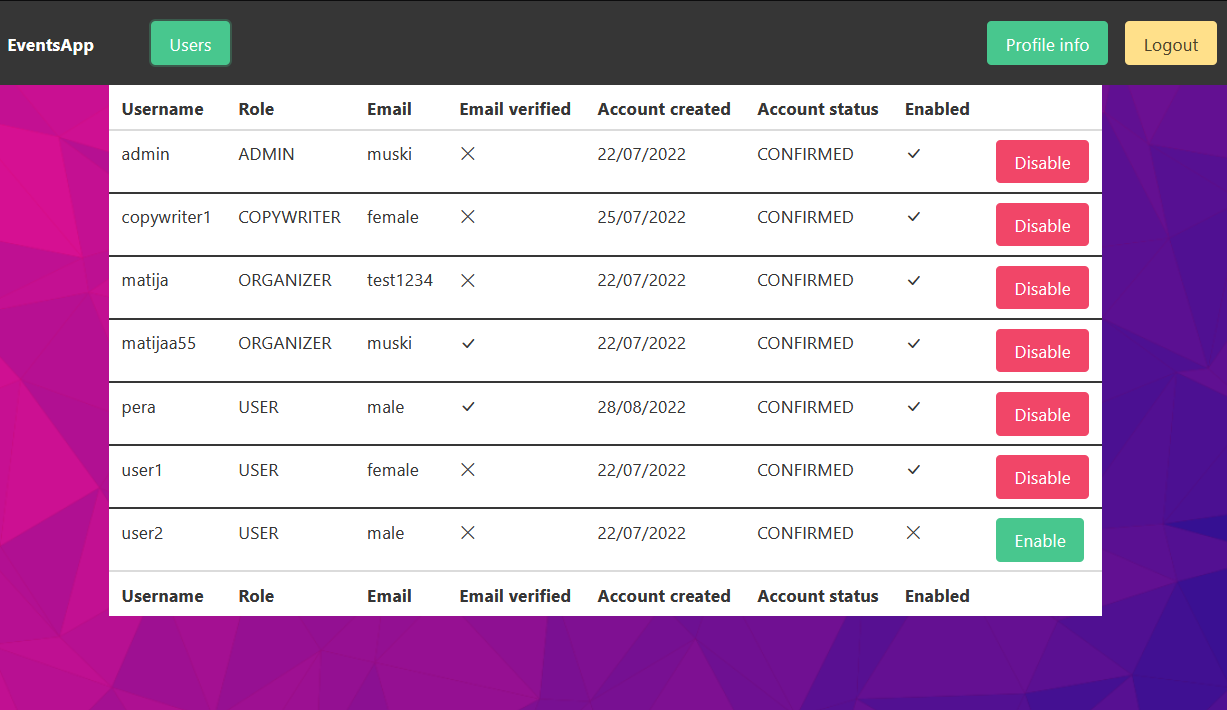
'body': json.dumps(response)

}

Листинг 4.7. Код за измену догађаја

## Администратор

Улога администратора апликације за организацију догађаја се такође додаје из Cognito конзоле. Администратор има увид у све догађаје, али и увид у све кориснике система као и у њихове основне информације. На слици 4.14. се налази пример приказа свих корисника система апликације за организацију догађаја.



Слика 4.14. Приказ корисника апликације за организацију догађаја

У оквиру приказа свих корисника администратор има могућност да неком кориснику онемогући приступ апликацији, али и да омогући приступ онима којима је то већ онемогућено.

Lambda функција која служи за добављање свих корисника је писана у Python програмском језику и приказана је у листингу 4.8. На почетку функције се добавља Cognito провајдер и приступни токен из којег се извуче улога којој корисник припада. Укорико корисник има улогу администратора функција успешно враћа листу свих корисника.

Lambda функција која служи за контролу приступа корисника апликацији је писана у Python програмском језику и приказана је у листингу 4.9. На почетку функције се добавља Cognito провајдер и приступни токен из којег се извуче улога и након тога се изврши ауторизација корисника. Након успешне ауторизације, из тела захтева се узима информација о кориснику и о томе да ли се омогућава или онемогућава приступ апликацији том кориснику и на основу те информације се позива одговарајућа функција Cognito провајдера.

import json

import boto3

import base64

def lambda\_handler(event, context):

client = boto3.client('cognito-idp')

accessToken = event['headers']['Authorization']

base64url = accessToken.split(".")[1]

decodedJwt = base64.b64decode(base64url + '==').decode('UTF-8')

decodedJwt = json.loads(decodedJwt)

cognitoGroups = decodedJwt['cognito:groups']

if 'ADMIN' not in cognitoGroups:

return {

'statusCode': 401,

'body': json.dumps('Unauthorized')

}

body = json.loads(event['body'])

username = body['username']

newStatus = body['newStatus']

Листинг 4.8. Код за добављање свих корисника апликације

import json

import boto3

import base64

def lambda\_handler(event, context):

client = boto3.client('cognito-idp')

accessToken = event['headers']['Authorization']

base64url = accessToken.split(".")[1]

decodedJwt = base64.b64decode(base64url + '==').decode('UTF-8')

decodedJwt = json.loads(decodedJwt)

cognitoGroups = decodedJwt['cognito:groups']

if 'ADMIN' not in cognitoGroups:

return {

'statusCode': 401,

'body': json.dumps('Unauthorized')

}

response = client.list\_users(

UserPoolId='us-east-1\_mOGgp8jYV'

)

return {

'statusCode': 200,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps(response['Users'], default=str)

}

Листинг 4.9. Код за контролу приступа корисника апликацији за организацију догађаја

if newStatus == 'disabled':

try:

response = client.admin\_disable\_user(

UserPoolId='us-east-1\_mOGgp8jYV',

Username= username

)

except Exception as e:

return {

'statusCode': 400,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps(str(e))

}

else:

return {

'statusCode': 200,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps('Successfuly disabled user!')

}

elif newStatus == 'enabled':

try:

response = client.admin\_enable\_user(

UserPoolId='us-east-1\_ycpyfk08i',

Username= username

)

except Exception as e:

return {

'statusCode': 400,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps(str(e))

}

else:

return {

'statusCode': 200,

'headers': {

'Access-Control-Allow-Origin': '\*'

},

'body': json.dumps('Successfuly enabled user!')

}

else:

return {

'statusCode': 400,

'body': json.dumps('Invalid status parameter')

}

# ЗАКЉУЧАК

У овом раду описана је реализована апликација за организацију догађаја са нагласком на имплементацију аутентификације и ауторизације корисника. Из разлога што апликација треба да подржи различите кориснике, њихову регистрацију и пријаву на систем, неопходно је обезбедити сигурно чување корисничких података као и контролисан приступ различитим информацијама и функцијама. То представља тежак и одговоран задатак за онога ко развија ту апликацију. Амазонови сервиси користе савремене и проверене технологије како би оствариле тај тежак и одговоран задатак чијом употребом се постиже лакше решавање споменутог проблема.

Детаљно су описани алати и Амазонови сервиси коришћени за реализацију апликације. Описан је начин функционисања сервиса као и њихово међусобно повезивање и шта је неопходно како би сервиси успоставили успешну комуникацију.

Објашњене су све улоге и њихове могућности у оквиру апликације, као и делови програмског кода који омогућавају функционисање апликације. Нагласак је стављен на делове имплементације везане за аутентификацију и ауторизацију.

У даљем развоју решења могуће унапређење би било омогућавање вишефакторске аутентификације, што би повећало сигурност корисника приликом пријаве.

# ЛИТЕРАТУРА

[1] Amazon documentation „What is IAM“ (Identity and Access Management), <https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>

[2] Amazon documentation, „What is Amazon Cognito“ (Amazon Cognito), <https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html>

[3] Amazon documentation „Unified Authentication and Authorization Mechanisms“ (аутентификација и ауторизација), <https://docs.aws.amazon.com/whitepapers/latest/logical-separation/unified-authentication-and-authorization-mechanisms.html>

[4] Amazon documentation „What is the AWS Management Console“ (AWS Management Console), <https://docs.aws.amazon.com/awsconsolehelpdocs/latest/gsg/learn-whats-new.html>

[5] Amazon „What is AWS“ (AWS), <https://aws.amazon.com/what-is-aws/>

[6] Amazon IAM Features „Manage IAM Permissions“ (Дозволе), [https://aws.amazon.com/iam/features/manage-permissions/](https://aws.amazon.com/iam/features/manage-permissions/%20%20)

[7] Amazon documentation „IAM users“ (IAM корисници), [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_users.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html%20)

[8] Amazon documentation „IAM user groups“ (IAM групе), [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_groups.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_groups.html%20)

[9] Amazon documentation „IAM roles“ (IAM улоге), [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html%20)

[10] Amazon documentation „Polices and permissions in IAM“ (политике), <https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html>

[11] Amazon documentation „Amazon Resource Names“ (Amazon Resource Name), [https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html](https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html%20)

[12] Amazon documentation „Amazon Cognito user pools“ (кориснички скуп), <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html>

[13] Amazon what-is „What is Multi-Factor Authentication“ (мултифакторска аутентификација), <https://aws.amazon.com/what-is/mfa/>

[14] Jwt introduction „Introduction to JSON Web Tokens“ (JSON Web Token), <https://jwt.io/introduction>

[15] IBM „Digital signature overview“ (Дигитално потписивање), <https://www.ibm.com/docs/en/b2badv-communication/1.0.0?topic=overview-digital-signature>

[16] Amazon what-is „What is an API“ (Application programming interface), <https://aws.amazon.com/what-is/api/>

[17] Amazon documentation „Types of access control - RBAC“ (Role Base Access control), [https://docs.aws.amazon.com/prescriptive-guidance/latest/saas-multitenant-api-access-authorization/access-control-types.html](https://docs.aws.amazon.com/prescriptive-guidance/latest/saas-multitenant-api-access-authorization/access-control-types.html%20)

[18] Amazon documentation „User pool attributes“ (Атрибути корисника), <https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-settings-attributes.html>

[19] Amazon documentation „Adding user pool password requirements“ (Подешавања лозинке), <https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-settings-policies.html>

[20] Amazon documentation „Customizing user pool workflows with Lambda triggers“ (Окидачи), <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools-working-with-aws-lambda-triggers.html>

[21] Amazon documentation „Configuring a user pool app client“ (Клијентске апликације), <https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-settings-client-apps.html>

[22] Mozilla documentation „CSS: Cascading Style Sheets“ (CSS), <https://developer.mozilla.org/en-US/docs/Web/CSS>

[23] Amazon documentation „What is AWS Lambda“ (AWS Lambda), [https://docs.aws.amazon.com/lambda/latest/dg/welcome.html](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html%20)

[24] Amazon documentation „Lambda function scaling“ (Контрола паралелности и скалирања), <https://docs.aws.amazon.com/lambda/latest/dg/invocation-scaling.html>

[25] Amazon documentation „Creating Lambda container images“ (Слике контејнера), <https://docs.aws.amazon.com/lambda/latest/dg/images-create.html>

[26] Amazon documentation „Configuring code signing for AWS Lambda“ (Потписивање кода), [https://docs.aws.amazon.com/lambda/latest/dg/configuration-codesigning.html](https://docs.aws.amazon.com/lambda/latest/dg/configuration-codesigning.html%20)

[27] Amazon documentation „Lambda Extensions API“ (Lambda екстензије), [https://docs.aws.amazon.com/lambda/latest/dg/runtimes-extensions-api.html](https://docs.aws.amazon.com/lambda/latest/dg/runtimes-extensions-api.html%20)

[28] Amazon What is NoSQL (NoSQL база података), [https://aws.amazon.com/nosql/](https://aws.amazon.com/nosql/%20)

[29] Amazon documentation „Working with REST APIs“ (REST API), [https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-rest-api.html](https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-rest-api.html%20)

[30] Mozilla Guides „What is URL“ (URL), [https://developer.mozilla.org/en-US/docs/Learn/Common\_questions/What\_is\_a\_URL](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL%20)

[31] Amazon documentation „What is Amazon API Gateway“ (API Gateway), <https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html>

[32] Amazon documentation „What is Amazon CloudWatch“ (CloudWatch), <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html>

[33] Amazon documentation „Using the ID token“ (идентификациони токен), [https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-the-id-token.html](https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-the-id-token.html%20)

[34] Amazon documentation „Using the access token“ (приступни токен), [https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-the-access-token.html](https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-the-access-token.html%20)

[35] Amazon documentation „Using the refresh token“ (освежавајући токен), <https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-the-refresh-token.html>

[36] Amazon documentation „Use API Gateway Lambda authorizers“ (овлашћења), <https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html>

[37] Mozilla Guides „HTTP“ (HTTP), <https://developer.mozilla.org/en-US/docs/Web/HTTP>

[38] Angular guide „What is Angular“ (Angular радно окружење), [https://angular.io/guide/what-is-angular](https://angular.io/guide/what-is-angular%20)

# БИОГРАФИЈА

Матија Ускоковић рођен је 30. августа 1999. године у Врбасу. Основну школу „Светозар Милетић“ завршио је у Врбасу 2014. године. Након тога уписује Гимназију „Жарко Зрењанин“ у Врбасу, коју завршава 2018. године. Исте године уписао се на Факултет техничких наука, смер Рачунарство и информатика. Положио је све испите предвиђене планом и програмом.

# КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

|  |  |
| --- | --- |
| Редни број, **РБР**: |  |
| Идентификациони број, **ИБР**: |  |
| Тип документације, **ТД**: | монографска публикација |
| Тип записа, **ТЗ**: | тексуални штампани документ |
| Врста рада, **ВР**: | дипломски рад |
| Аутор, **АУ**: | Матија Ускоковић |
| Ментор, **МН**: | проф. др Горан Сладић, ФТН Нови Сад |
| Наслов рада, **НР**: | Примена AWS IAM и Cognito сервиса за контролу приступа |
| Језик публикације,**ЈП**: | српски |
| Језик извода, **ЈИ**: | српски / енглески |
| Земља публиковања, **ЗП**: | Србија |
| Уже географско подручје, **УГП**: | Војводина |
| Година, **ГО**: | 2022 |
| Издавач, **ИЗ**: | ауторски репринт |
| Место и адреса, **МА**: | Нови Сад, Факултет техничких наука,  Трг Доситеја Обрадовића 6 |
| Физички опис рада, **ФО**: | бр. поглавља 5/ страница 47/ цитата / табела / слика 23/ графикона / прилога / листинга 9 |
| Научна област, **НО**: | Електротехничко и рачунарско инжењерство |
| Научна дисциплина, **НД**: | Безбедност у системима електронског пословања |
| Предметна одредница/кључне речи, **ПО**: | аутентификација, ауторизација, пријава, регистрација |
| **УДК** |  |
| Чува се у, **ЧУ**: | Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад |
| Важна напомена, **ВН**: |  |
| Извод, **ИЗ**: | У раду је описана апликација за организацију догађаја која за аутентификацију и ауторизацију корисника користи Амазонове сервисе |
| Датум прихватања теме, **ДП**: |  |
| Датум одбране, **ДО**: |  |
| Чланови комисије, **КО**: |  |
| председник | др Гордана Милосављевић, ред. проф, ФТН Нови Сад |
| члан | др Мирослав Зарић, ванр. проф, ФТН Нови Сад |
| ментор | др Горан Сладић, ред. проф, ФТН Нови Сад |
| Потпис ментора | |

# KEY WORDS DOCUMENTATION

|  |  |
| --- | --- |
| Accession number, **ANO**: |  |
| Identification number, **INO**: |  |
| Document type, **DT**: | monographic publication |
| Type of record, **TR**: | textual material |
| Contents code, **CC**: | BSc thesis |
| Author, **AU**: | Matija Uskoković |
| Mentor, **MN**: | Goran Sladić, PhD, assoc. prof., FTN Novi Sad |
| Title, **TI**: | Application of AWS IAM and Cognito services for access control |
| Language of text, **LT**: | serbian |
| Language of abstract, **LA**: | serbian / english |
| Country of publication, **CP**: | Serbia |
| Locality of publication, **LP**: | Vojvodina |
| Publication year, **PY**: | 2022 |
| Publisher, **PB**: | author’s reprint |
| Publication place, **PP**: | Novi Sad, Faculty of Technical Sciences, Square Dositej Obradović 6 |
| Physical description, **PD**: | no. of chapters 5/ pages 47/ quotes / tables / pictures 23/ graphs / appendix / listings 5 |
| Scientific field, **SF**: | Electrical and computer engineering |
| Scientific discipline, **ND**: | Security in e-business systems |
| Subject / Keywords, **S/KW**: | authentification, authorization, signIn, signUp |
| **UDC** |  |
| Holding data, **HD**: | Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad |
| Note, **N**: |  |
| Abstract, **AB**: | This paper describes application for event organization that uses Amazon services for authentication and authorization |
| Accepted by sci. board on, **ASB**: |  |
| Defended on, **DE**: |  |
| Defense board, **DB**: |  |
| president | Gordana Milosavljević, PhD, full. prof, FTN Novi Sad |
| member | Miroslav Zarić, PhD, assoc. prof, FTN Novi Sad |
| mentor | Goran Sladić, PhD, full. prof, FTN Novi Sad |
| Mentor’s signature | |