УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Никола Љ. Мићић

РАЗВОЈ СИСТЕМА ЗА ПРЕПОРУКУ У AWS ОБЛАКУ КОРИШЋЕЊЕМ СЕРВИСА AMAZON PERSONALIZE

мастер рад

Београд, 2024.

Ментор:

др Александар Картељ, ванредни професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Владимир Филиповић, редовни професор
Универзитет у Београду, Математички факултет

др Сташа Вујичић Станковић, доцент
Универзитет у Београду, Математички факултет

Датум одбране:

Желео бих да изразим захвалност свом ментору, Александру Картељу, пре свега на квалитетним предавањима током студија на којима сам стекао велико знање, а затим и на прецизним и корисним саветима током писања мастер рада без којих реализација овог рада не би била могућа. Такође, желим да изразим и велику захвалност компанији Cloudwalker и колегама, а посебно Бојану Совиљу, као неком ко ми је био велика подршка током израде пројекта и без чијих савета пројекат не би био овог квалитета. Поред тога се захваљујем породици и пријатељима који су веровали у мене за све ове године студирања и који су ми били велика подршка при свакој препреци на коју сам наишао.

Наслов мастер рада: Развој система за препоруку у *AWS* облаку коришћењем сервиса *Amazon Personalize*

Резиме: Овај рад се бави истраживањем и развојем система за препоруку у области интернет платформи. Решење је модуларно и може се прилагодити различитим индустријским секторима и областима, чиме се повећава значај овог рада с обзиром на појаву све већег броја интернет платформи које имају потребу за персонализованим искуством корисника ради бољег сналажења и бржег проналажења одговарајућег садржаја.

Главни циљ овог рада је имплементација система за препоруку који се извршава у *AWS* облак инфраструктури, а који се састоји од различитих *AWS* сервиса. Главни део решења јесте *Amazon Personalize* сервис који представља сервис машинског учења за генерисање препорука, односно персонализацију садржаја унутар различитих веб страница/апликација.

Осим *Amazon Personalize* сервиса, ово решење користи и друге *AWS* сервисе као што су *AWS Glue* за ЕТЛ (*ETL – Extract, Transform and Load*) процесе, *Amazon S3* за складиштење података, *Step Functions* за дефинисање тока процеса, *AWS Lambda* за извршавање процеса, а поред тога и *Amazon DynamoDB* као *NoSQL* базу за чување генерисаних препорука. Имплементација ће бити фокусирана на генерисање скупова података, затим прикупљање података генерисаних скупова, на основу којих се врши обучавање модела и на крају генерисање препорука на основу обученог модела. Идеја је да целокупно решење буде имплементирано кроз *Инфраструктуру као кôд* (*Infrastructure as Code – IaC*) уз помоћ сервиса *AWS CloudFormation*.

Кључне речи: информатика, рачунарство, облак, систем, препорука

Садржај

Глава 1

Увод

У савременом дигиталном добу, системи за препоруку постали су један од најважнијих модула интернет платформи и играју кључну улогу у персонализацији корисничког искуства. Коришћењем напредних алгоритама и техника машинског учења, системи за препоруку анализирају корисничке податке и њихово понашање, на основу чега предлажу релевантне садржаје, производе или услуге. Системи за препоруку имају широку примену и значајан утицај на различите индустрије данашњице, укључујући е-трговину, друштвене мреже, платформе за онлајн оглашавање и емитовање мултимедијалног садржаја, а поред тога налазе примену и у туризму и образовању.

Један од највећих изазова данашњице је управљање огромном количином информација које су доступне на интернету. Корисници су често суочени са проблемом преоптерећености информацијама, што им отежава проналажење одговарајућег садржаја. Системи за препоруку решавају овај проблем тако што филтрирају и анализирају велике количине података, пружајући корисницима предлоге који су у складу са њиховим интересовањима и потребама. На овај начин, корисници не само да штеде време, већ добијају боље и персонализованије искуство.

Поред директне користи за кориснике, системи за препоруку доносе велике предности и за саме платформе. У е-трговини, на пример, персонализоване препоруке могу значајно повећати продају, подстичући кориснике на куповину производа који одговарају њиховим интересовањима. На платформама за емитовање садржаја, препоручивање релевантног садржаја повећава време које корисници проводе на платформи, што доводи до веће лојалности и задржавања корисника. Такође, системи за препоруку могу помоћи у опти-

мизацији оглашавања, циљајући огласе на основу корисничких преференција и понашања.

Овај рад се бави пројектом који представља решење система за препоруку у *AWS* облаку коришћењем *Amazon Personalize* сервиса, који представља сервис машинског учења за генерисање препорука. Посебна пажња ће бити посвећена опису архитектуре система, укључујући кораке од прикупљања и обраде података до генерисања и испоруке препорука корисницима. У раду ће бити приказано како *AWS* сервиси међусобно комуницирају и учествују у формирању једног сложеног система као што је систем за препоруку. Поред тога ће бити приказано како *AWS* архитектура не само да обезбеђује високе перформансе и поузданост, већ омогућава и лаку надоградњу и прилагођавање специфичним потребама корисника, чиме се постижу бољи пословни резултати и побољшава корисничко искуство.

1.1 Увод у *AWS* облак

Amazon Web Services (AWS) представља једну од водећих платформи за рачунарство у облаку (*cloud computing*), пружајући разноврсне услуге и ресурсе који омогућавају компанијама да граде сложене апликације и управљају комплексним инфраструктурама. *AWS* платформа је дизајнирана тако да буде скалабилна, флексибилна и економична, што је чини идеалним избором за формирање и имплементацију система за препоруку.

Системи за препоруку захтевају обраду великих количина података, примену напредних алгоритама машинског учења и могућност брзе прилагодљивости променама у корисничким понашању и њиховим потребама. *AWS* нуди велики скуп сервиса који подржавају ове захтеве, као што су: *Amazon Personalize* сервис машинског учења за генерисање препорука, *AWS Glue* за ЕТЛ процесе, *Amazon S3* за складиштење података, *Step Functions* за дефинисање тока процеса, *AWS Lambda* за извршавање процеса, а поред тога и *Amazon DynamoDB* као *NoSQL* базу за чување генерисаних препорука.

1.2 Анализа решења на апстрактном нивоу

Имплементација система за препоруку у *AWS* облак инфраструктури подразумева коришћење већег броја *AWS* сервиса који заједно доводе до формирања ефикасног и скалабилног решење за генерисање персонализованих препорука. Ово решење је дизајнирано тако да обрађује велике количине података, затим примењује алгоритме машинског учења помоћу *Amazon Personalize* сервиса за генерисање персонализованих препорука и на крају испоручује формиране препоруке на платформе као што су веб странице и мобилне апликације, чиме ће се побољшати корисничко искуство и повећати интеракција корисника са самом платформом. Кроз наредна поглавља биће детаљно анализирани аспекти конкретне имплементације, затим објашњен архитектурални дијаграм решења и наглашене предности развоја модерног система за препоруке у *AWS* облак инфраструктури.

1.3 Преглед коришћених *AWS* сервиса

У наставку се налази преглед и краће објашњење коришћених *AWS* сервиса у оквиру имплементираног решења система за препоруку.

На слици 1.1 која следи приказан је *AWS* архитектурални дијаграм који представља предложено решење система за препоруку и који се састоји од следећих сервиса:

1. *Amazon Personalize* – представља сервис уз помоћ ког се врши обучавање модела машинског учења и генерисање персонализованих препорука на основу корисничких интеракција и метаподатака који представљају скупове података за обучавање модела. [3]

2. *Amazon Redshift* – представља брзи и скалабилни складишни и аналитички сервис за велике количине података, одакле се дохватају подаци о корисницима и њиховим интеракцијама на основу којих се праве скупови података за обучавање модела уз помоћ претходно наведеног сервиса. [5]

3. *Amazon S3* – представља сервис за сигурно и скалабилно складиштење података у облаку који се у овом решењу користи за чување датотеке са подешавањима за *Amazon Personalize* моделе машинског учења, складиштење скупова података помоћу којих ће се обучавати *Amazon Personalize* модели и на крају за смештање резултата *Amazon Personalize* сервиса у виду генерисаних персонализованих препорука. [6]

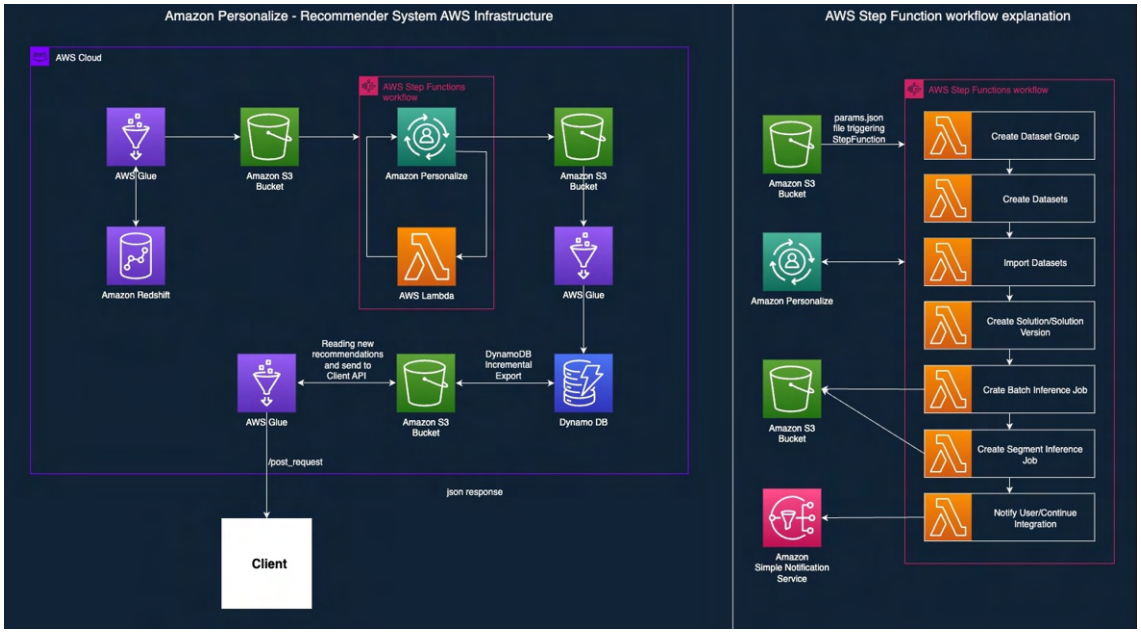
4. *AWS Glue* – представља сервис за ЕТЛ процесе и користи се у овом решењу за аутоматизацију процеса прикупљања, чишћења и трансформисања података који ће представљати скупове података потребне за обучавање *Amazon Personalize* модела. [9]

5. *AWS Step Functions* – представља сервис за усклађивање и дефинисање сложених токова процеса кроз дефинисање низа стања којима се може осигурати сигурно и скалабилно извршавање и прелази из једног у друго стање. [11]

6. *AWS Lambda* – представља сервис који омогућава покретање кôда без претходног обезбеђивања или управљања инфраструктуром. Подржава велики број програмских језика. Једна од важнијих карактеристика

овог сервиса је то што догађаји других *AWS* сервиса могу представљати окидач за покретање *AWS Lambda* кôда. [10]

7. *Amazon DynamoDB* – представља брзу и флексибилну *NoSQL* базу, која се у овом решењу користи за чување персонализованих генерисаних препорука по кориснику. [2]



Слика 1.1: *AWS* архитектурални дијаграм

Глава 2

Имплементација решења

Након увода о системима за препоруку, *AWS* облаку и кратког објашења коришћених сервиса у конкретном решењу, сада следи детаљнији опис сервиса који представља централну фигуру имплементираног решења, а то је *Amazon Personalize* сервис.

2.1 *Amazon Personalize* сервис

Као што је наведено у претходном поглављу, *Amazon Personalize* представља сервис машинског учења који омогућава развој и имплементацију персонализованих препорука за различите апликације и платформе. [3]

За добре резултате било ког модела машинског учења је потребан добро структуирани и велики извор података на основу којих ће се модели обучавати, како би излази тих модела давали добра предвиђања. Иста ситуација је и код *Amazon Personalize* сервиса, где прикупљање и припрема довољне количине података који ће бити структуирани на прави начин представљају веома важан корак у прављења било ког система за препоруку.

Како је организован овај сервис?

У оквиру сервиса *Amazon Personalize*, групе скупова података и скупови података представљају основне концепте и логичке јединице који омогућавају организацију и коришћење података за формирање персонализованих препорука.

Група скупова података представља логичку групу која садржи више скупова података и решења, где све компоненте унутар ове групе деле исти контекст и намењене су заједничком циљу, као што је генерисање препорука за одговарајућу апликацију или сервис.

Прављење ове групе представља први корак у процесу формирања система за препоруку и ова група омогућава да се организују сви скупови података и да се управља њима, моделима који су обучени на основу тих података и кампањама на основу којих се добијају препоруке из датих модела.

Скупови података су структуре унутар групе скупова података које чувају одговарајуће типове података потребне за обучавање модела машинског учења. Постоје три основна скупа тих података:

1. Корисници – Овај скуп садржи информације о корисницима, као што су јединствени кориснички идентификациони бројеви (ИД-еви) и атрибути као што су старост, пол, чланство програма лојалности или чак омиљена боја или хоби. Максимални број колона овог скупа података је 25.

2. Ставке – Овај скуп садржи бројевне и категоричке податке о ставкама које се препоручују, као што су јединствени идентификациони бројеви ставки, категорије тих ставки, описи, цене, доступности и било који други атрибути који могу повећати квалитет система за препоручивање. Максимални број колона овог скупа података је 100.

3. Интеракције – Овај скуп бележи интеракције између корисника и ставки, као што су куповине, прегледи, оцењивања или додавања у листу жеља. Свака интеракција садржи кориснички ИД, ИД ставке, информацију о времену када је дошло до интеракције и опционе атрибуте као што је примера ради тип интеракције (преглед, клик). Сваки скуп интеракција мора имати минимум 1000 интеракција између корисника и ставки и поред тога мора имати минимум 25 јединствених корисника са по минимум 2 интеркације. За што боље резултате система за препоруку препоручује се коришћење скупова података који имају минимум 50000 интеракција и минимум 1000 корисника са по бар 2 интеракције. [4]

Након формирања групе скупова података, потребно је дефинисати шему за сваки од претходно наведена три скупа података. За сваки скуп података (Корисници, Ставке, Интеракције) потребно је дефинисати по шему која

описује структуру и типове атрибута који ће бити укључени. Шема је дефинисана у *JSON* формату и укључује информације као што су називи колона и типови података (нпр. *integer*, *string*, *float*).

Након дефинисања шема, врши се учитавање података у одговарајуће скупове података. Аутоматизације процеса прикупљања, чишћења и трансформисања података у овом решењу су имплементирани помоћу сервиса *AWS Glue*, који представља сервис за ЕТЛ процесе. Помоћу наведеног *AWS Glue* сервиса подаци се довлаче из *Amazon Redshift*-а, који представља решење складишта података за конкретног клијента којем се испоручује систем за препоруку. Дато складиште података се константно освежава новим подацима који пристижу са клијентске продукционе базе уз помоћ *Kafka* платформе за емитовање промена на клијентској продукционој бази, тако да се сваким извршавањем *AWS Glue* сервиса прикупљају освежени и нови подаци који укључују нове кориснике, нове ставке и нове интеракције и сви се смештају на *Amazon S3*. Ово је сервис за сигурно и скалабилно складиштење података у облаку и у овом случају представља извор одакле се увозе подаци у наша три скупа података која ће се користити за формирање модела система за препоруку.

Пример шеме и података за скуп података интеракција

На кôду 2.1 приказана је шема за скуп података интеракција, где се може видети да се тај скуп састоји од три колоне које представљају идентификациони број корисника (тип *string*), идентификациони број ставке (тип *string*) и информацију о времену када је дошло до интеракције.

```
"Interactions":{
   "name":"users-casino-games-interactions",
   "schema":{
     "type": "record",
     "name": "Interactions",
     "namespace": "com.amazonaws.personalize.schema",
     "fields": [
        {
           "name": "user_id",
           "type": "string"
        },
        {
           "name": "item_id",
           "type": "string"
        },
        {
           "name": "timestamp",
           "type": "long"
        }
     ],
     "version": "1.0"
  }
}
```

Кôд 2.1: Шема скупа података интеракција

На кôду 2.2 је приказан део *CSV* датотеке која је смештена на *Amazon S3* и која представља податке са интеракцијама, где се може видети пример једног корисника са идентификационим бројем *000000a0-2vfd-3df2-b16f* који је интераговао са ставком *1134* у 4 различита временска оквира.

```
1 user_id, item_id, timestamp
2 000000 0 -2vfd-3df2-b16f,1134,1718015220
3 000000 0 -2vfd-3df2-b16f,1134,1718018348
4 000000 0 -2vfd-3df2-b16f,1134,1718018950
5 000000 0 -2vfd-3df2-b16f,1134,1718019402
```

Кôд 2.2: Пример датотеке у којем се налазе подаци са интеракцијама

2.2 Учитавање података на *Amazon S3*

Редовним дневним извршавањем послова *AWS Glue* сервиса врши се прикупљање нових података који укључују нове кориснике, нове ставке и нове интеракције које се смештају на *Amazon S3* складиште података. Логички ресурс сервиса *AWS Glue* који врши претходно описан процес представља *AWS Glue* ЕТЛ посао, најновије верзије 4.0, који је изграђен над *Spark* контекстом верзије 3.3.0 и написан у програмском језику *Python*, верзије 3.10.0. [8]

AWS Glue посао повлачи нове податке из *Redshift* складишта за податке користећи *boto3 redshift-data* клијента у оквиру методе чији кôд се може видети у оквиру кôда 2.3.

```
1  import boto3
2  redshift = boto3.client("redshift-data")
3
4  def execute_statement_no_params(redshift,query,statement_name):
5
6      print("statement started: ", statement_name)
7      response = redshift.execute_statement(
8          ClusterIdentifier="redshift-cluster",
9          Database="dwh",
10         SecretArn="arn:aws:secretsmanager:eu-central
               -1:110619981234:secret:sm-cwclient-redshift-
               credentials",
11         Sql=query,
12         StatementName=statement_name,
13     )
14     response_id = response["Id"]
15     check_statement_status(redshift,statement_name, response_id)
```

Кôд 2.3: Метода за позивање упита користећи *boto3 redshift-data* клијента

Користећи иницијализованог клијента, позива се *execute_statement* метода којој се прослеђују идентификатор *Redshift* кластера, назив базе и креденцијали за приступ бази. Креденцијали за приступ бази се прослеђују кроз јединствени идентификатор *SecretArn* сервиса *AWS Secrets Manager*, који се користи за централизовано и сигурно управљање креденцијалима. Поред тога се додељује упит који се прослеђује као параметар у оквиру методе

execute_ statement_ no_ params, а конкретан упит за корисничке интеракције се може видети у оквиру кôда 2.4.

Конкретан упит је параметризован вредношћу времена када је *AWS Glue* посао започет, а резултат упита се испоручује на *Amazon S3* локацију. Путања *Amazon S3* локације се састоји од *Amazon S3* репозиторијума, назива скупа података који се тренутно учитава (у овом случају су то интеракције) и партиционисања које указује на то које године, месеца и дана су учитани подаци. Поред тога се команди прослеђује ресурс *AWS IAM* сервиса за управљање дозволама које омогућавају *AWS Glue* сервису да приступи сервису *Redshift* и испоручи податке на *Amazon S3*.

```
query_interactions_unload = """
UNLOAD (
    '
    select distinct
            b.user_uuid as user_id,
            a.gid as item_id,
            extract(epoch from a.transaction_date::timestamp) as
                timestamp
    FROM ext_kafka.ic_user_account_transaction a
            LEFT join ext_kafka.ic_user_account b on a.
               user_account_id = b.id
    where a.transaction_date::timestamp <= ''{job_start_time}''
        and a.gid is not null
    ;
    '
)
TO 's3://project-cw-casino-rec-sys-personalize-inputbucket/
   datasets/Interactions/{partition_year}/{partition_month}/{
   partition_day}/Interactions_{date_oid_from_2}_{date_oid_to_2}
   _{hour_min_sec}_'
IAM_ROLE 'arn:aws:iam::11069981234:role/redshift-to-s3'
PARALLEL OFF
ALLOWOVERWRITE
HEADER
CSV;
"""
```

```
22 statement_name = "select_interactions_unload"
23 query_interactions_unload_parametrized =
      query_interactions_unload.format(job_start_time = params[7],
      partition_year = params[2] , partition_month = params[3] ,
      partition_day = params[4], date_oid_from_2=params[0] ,
      date_oid_to_2=params[1], hour_min_sec=params[5])
24 execute_statement_no_params(redshift,
      query_interactions_unload_parametrized,statement_name)
```

Кôд 2.4: Упит за извоз података са *Redshift* складишта података на *Amazon S3*

2.3 Учитавање података у формиране скупове података

Након што је *AWS Glue* сервис извршио редовни дневни посао прикупљања нових и освежених података и све их сместио у *Amazon S3* складиште података предвиђено за систем за препоруку, следи учитавање тих података у формирану групу скупова података која је објашњена у претходној глави.

AWS Lambda функција користи иницијализоване клијенте уз помоћ *boto3* библиотеке за комуникацију са различитим сервисима, позивањем *Loader* пакета из дељеног слоја *AWS Lambda* функције који служи за дефинисање зависности библиотека и писање кôда који ће се користити у више различитих *AWS Lambda* функција, чиме се избегава вишеструко писање истог кôда.

У оквиру кôда 2.5 се налази функција којом се учитавају подаци у формиране скупове података помоћу клијента за *Amazon Personalize* сервис. У оквиру те методе се спецификује која локација *Amazon S3* складишта података одговара ком скупу података одговарајуће групе која је претходно формирана. Догађај који је покренуо дату функцију садржи информације о томе који подаци *Amazon S3* складишта података одговарају којем скупу података и то је спецификовано параметром `dataSource`. `jobName` представља назив посла за учитавање података, `datasetArn` представља идентификатор шеме скупа података, `roleArn` представља идентификатор улоге *AWS IAM* сервиса која омогућава функцији да комуницира са сервисима *Amazon S3* и *Amazon Personalize*, док `importMode` представља начин учитавања података, односно да ли се врши иницијално учитавање – *FULL*, односно *INCREMENTAL* уколико је реч о неиницијалном учитавању.

```
1 LOADER.personalize_cli.create_dataset_import_job(
2             jobName="{datasetType}_{date}".format(**event),
3             datasetArn=event["datasetArn"],
4             dataSource={
5                 "dataLocation": "s3://{bucket}/datasets/{
                      datasetType}/{today_year}/{today_month}/{
                      today_day}/".format(**event)
6             },
7             roleArn=environ["PERSONALIZE_ROLE"],
```

```
8               importMode=import_parameter_value
9           )
```

Кôд 2.5: Учитавање података у *Amazon Personalize* скупове података

На овај начин се свакодневно пуне скупови података новим корисницима, ставкама и интеракцијама и уз помоћ тих скупова података се врши обучавање *Amazon Personalize* модела који ће бити објашњени у наредном поглављу.

2.4 Увод у *Amazon Personalize* моделе и решења

Када се сви подаци унесу у формиране скупове података, долази на ред прављење решења, односно подешавање модела сервиса *Amazon Personalize*, који ће се обучавати уз помоћ унетих података и чији ће се резултати искористити за персонализоване препоруке.

У оквиру сервиса *Amazon Personalize*, решења (*Solution*) и рецепти (*Recipe*) представљају основне концепте за прављење и обучавање модела који генеришу персонализоване препоруке. *Amazon Personalize* решење је подешавање модела машинског учења које обухвата све потребне параметре за обучавање и генерисање препорука, а ово решење је дефинисано на основу специфицираног рецепта и скупова података увезених у групу скупова података. Постоје 2 основна типа решења која се бирају приликом његовог формирања, а то су *Item Recommendations*, који представља случај када је потребно решење за генерисање персонализованих препорука ставки по кориснику, односно формирање система за препоруку, што и јесте тема овог рада. Други тип је *User Segmentation*, у ком случају се прави решење коме се на основу уноса одговарајуће ставке добија излаз који представља скуп сегментираних корисника којима би требало препоручити дату ставку. Овај тип се користи у сврхе маркетиншких кампања.

Рецепт представља претходно дефинисан алгоритам који је оптимизован за одговарајуће типове препорука. *Amazon Personalize* сервис нуди више унапред дефинисаних рецепата (за *Item Recommendations* тип решења), од којих је сваки рецепт дизајниран за специфичан случај употребе, као што су:

1. *User personalization* – користи се у случају предвиђања ставки са којима

ће корисник интераговати на основу метаподатака о самим корисницима, ставкама и њиховим интеракцијама.

2. *Similar items* – користи се у случају препоручивања сличних ставки, ставкама које су спецификоване као улаз датог решења и помажу корисницима у истраживању и откривању нових ставки, на основу метаподатака ставки и корисничких интеракција са платформом.

3. *Item popularity* – користи се у случају препоручивања најпопуларнијих ставки међу корисницима и ставкама које постају све популарније на платформи.

4. *Personalized ranking* – користи се у случају када је потребно за одговарајући скуп ставки, направити персонализовану ранг листу, односно поредак којим ће се специфичном кориснику предложити дате ставке.

Након што се изабере рецепт који највише одговара специфичним потребама апликације, прави се решење које укључује назив решења, одабрани рецепт, формирана група скупова података за коју се прави решење и додатна подешавања као што су метрике за оптимизацију и број препорука. Након тога се прави конкретна верзија решења која користи тренутне податке из три основна скупа података за кориснике, ставке и интеракције. Конкретни подаци из ова три скупа података су кључни за обучавање модела који је подешен у оквиру решења које је описано у претходном пасусу и на основу тих података се генерише верзија решења, односно обучени модел помоћу којег добијамо прецизне и персонализоване препоруке.

2.5 Опис изабраног *Amazon Personalize* модела и решења система за препоруку

Предложено решење система за препоруку које представља тема овог рада користи први тип решења који је описан у претходном поглављу, односно *Item recommendations*, који пружа персонализоване препоруке за ставке корисницима на основу њихових претходних интеракција са другим ставкама. Ово решење је посебно корисно за компаније које желе да унапреде своје платформе помоћу персонализованих препорука и тиме побољшају ангажованост

корисника чиме се повећава број прегледа и задржавање корисника на платформи.

Рецепт који користимо за описано решење је *User personalization* рецепт, који је специјално дизајниран за генерисање персонализованих препорука корисницима на основу њихових претходних интеракција са различитим ставкама. Овај рецепт је осмишљен тако да препоручи ставке које ће највероватније заинтересовати појединачног корисника.

User personalization рецепт користи методе колаборативног филтрирања и дубоког учења како би предвидео које ставке ће бити најрелевантније за појединачног корисника. Колаборативно филтрирање користи информације о интеракцијама свих корисника да би направило препоруке за појединог корисника, док дубоко учење користи сложене неуронске мреже да би научило сложене обрасце у подацима.

Описани рецепт функционише на следећи начин:

1. Анализа интеракција – Алгоритам анализира историјске податке о интеракцијама корисника са ставкама и на основу ових података, алгоритам учи које ставке су релевантне за различите кориснике.

2. Предвиђање релевантности – Када систем затражи препоруке за конкретног корисника, алгоритам користи обучени модел да би предвидео које ставке ће бити најрелевантније за тог корисника. Ово се ради на основу сличности између корисника, као и сличности између ставки.

3. Генерисање препорука – Алгоритам генерише листу препорука за корисника, рангирајући ставке по њиховој релевантности, а најрелевантније ставке су на врху листе.

Предности описаног рецепта:

1. Висока прецизност –Користећи алгоритме машинског учења, рецепт може генерисати високо прецизне и релевантне препоруке.

2. Персонализовано искуство – Препоруке су прилагођене индивидуалним интересима и понашању корисника, што побољшава корисничко искуство.

3. Флексибилност – Алгоритам је довољно флексибилан да се прилагоди различитим типовима апликација и индустрија, укључујући е-трговину,

друштвене мреже, платформе за онлајн оглашавање и емитовање мултимедијалног садржаја.

2.6 Формирање описаног *Amazon Personalize* модела и решења

Кôд 2.6 представља *AWS Lambda* функцију за креирање решења *Amazon Personalize* модела која се позива при сваком покретању система, а параметри који садрже све потребне податке за прављење решења су прослеђени кроз догађај `event` који окида извршавање ове функције. Приликом иницијалног позивања система ови подаци ће се искористити за прављење решења, а сваки наредни пут када се буде извршавала ова функција, само ће доћи до провере да ли је одговарајуће решење већ направљено позивањем методе `describe_solution` и уколико јесте, извршавање система се наставља.

```
1  def create_solution(solutionArn, params):
2      try:
3          status = LOADER.personalize_cli.describe_solution(
4              solutionArn=solutionArn
5          )['solution']['status']
6
7      except LOADER.personalize_cli.exceptions.
           ResourceNotFoundException:
8          LOADER.logger.info(
9              'Solution not found! Will follow to create a new
                   solution.'
10         )
11         LOADER.personalize_cli.create_solution(**params)
12         status = LOADER.personalize_cli.describe_solution(
13             solutionArn=solutionArn
14         )['solution']['status']
15
16     while status in {'CREATE PENDING', 'CREATE IN_PROGRESS'}:
17         status = LOADER.personalize_cli.describe_solution(
18             solutionVersion=solutionArn
19         )['solution']['status']
```

```
20
21      if status != 'ACTIVE':
22          raise actions.ResourceFailed
23
24  def lambda_handler(event, context):
25      solutionArn = ARN.format(
26          region=environ['AWS_REGION'],
27          account=LOADER.account_id,
28          name=event['solution']['{}'.format(event['solutionType']
                )]['name']
29      )
30
31      create_solution(solutionArn, event['solution']['{}'.format(
            event['solutionType'])])
```

Кôд 2.6: Прављење решења

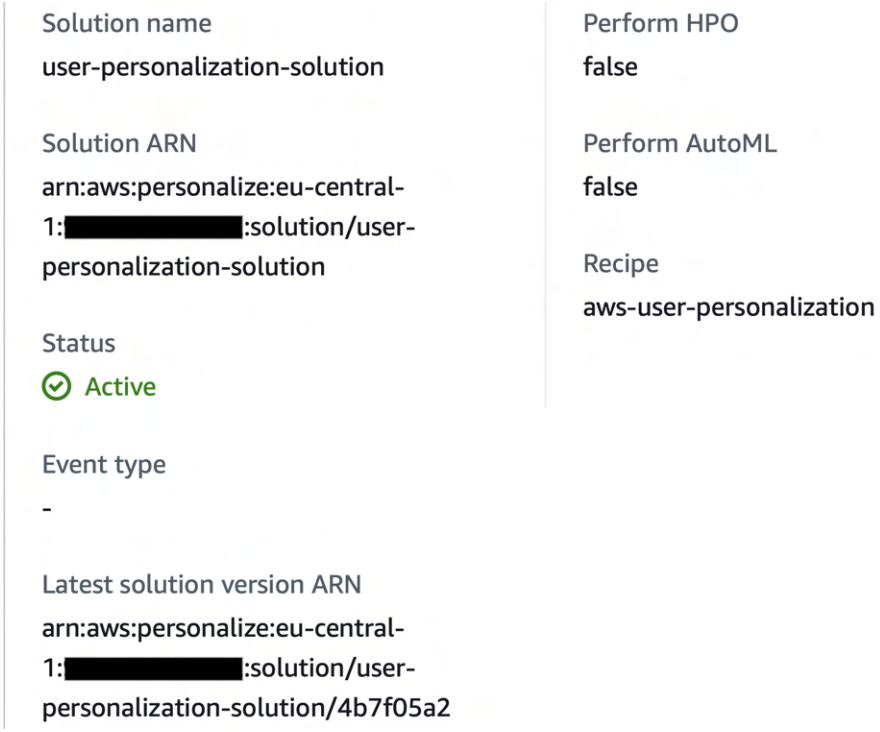
Као што се може видети у оквиру кôду 2.7, након провере да већ постоји направљено решење, извршавање система се наставља позивањем методе за прављење нове верзије решења. Параметром ове методе се спецификује тачно решење и додељени рецепт тог решења који ће се користити при прављењу нове верзије.

```
solutionArn = ARN.format(
    region=environ["AWS_REGION"],
    account=LOADER.account_id,
    name=event["solution"]["{}".format(event["solutionType"])]["
        name"]
)

event["solution"]["{}".format(event["solutionType"])]["
    datasetGroupArn"] = event["datasetGroupArn"]
# check if solution is already created
create_solution(solutionArn, event["solution"]["{}".format(event
    ["solutionType"])])

solutionVersionArn = LOADER.personalize_cli.
    create_solution_version(
     solutionArn=solutionArn,
     trainingMode="FULL"
)["solutionVersionArn"]

return solutionVersionArn
```

Кôд 2.7: Прављење нове верзије решења

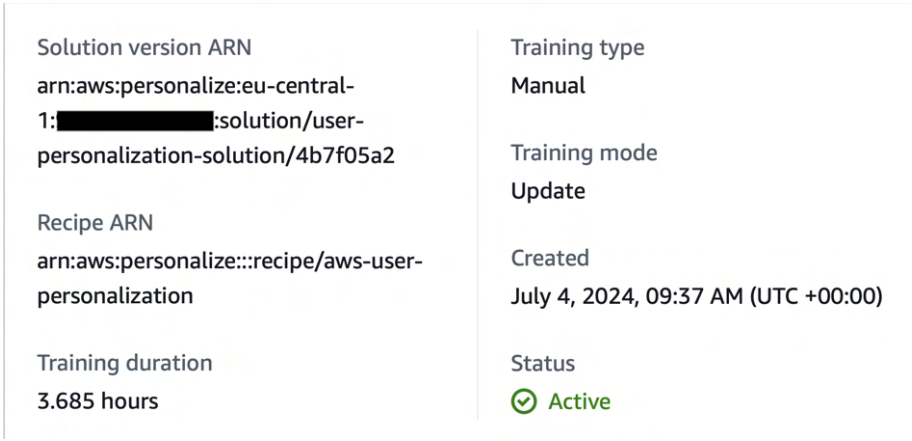
2.7 Валидација *Amazon Personalize* модела/решења

Кроз конзолу се може видети како изгледа формирано *Amazon Personalize* решење, а на слици 2.1 је приказано конкретно решење коришћено у пројекту које прати овај рад.



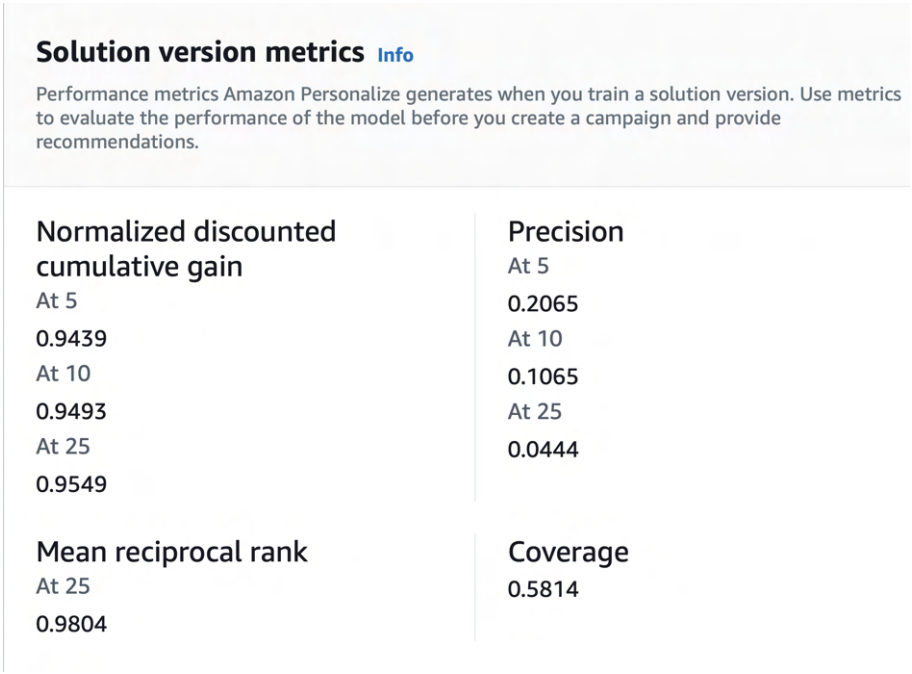
Слика 2.1: Преглед конкретног решења *Amazon Personalize* сервиса

Када се верзија решења генерише након обучавања модела, он се валидира и процењује коришћењем различитих метрика као што су прецизност и покривеност, што помаже у одређивању успешности модела у генерисању релевантних препорука. На слици 2.2 је приказана конкретна верзија решења на којем се могу видети информације о трајању обуке, времену формирања, коришћеном рецепту, типовима обука и коначном статусу.



Слика 2.2: Преглед конкретне верзије решења *Amazon Personalize* сервиса

На слици 2.3 се може видети преглед метрика конкретне верзије решења које указују колико је дата верзија решења успешна и самим тим колико ће њене препоруке бити релевантне и у складу са корисничким потребама.



Слика 2.3: Преглед метрика конкретне верзије решења *Amazon Personalize* сервиса

Ранг који се помиње у метрикама односи на позицију препоручене ставке у листи препорука, а претпоставља се да је прва позиција најрелевантнија за

корисника. За сваку метрику максимална вредност је 1, а што је вредност ближа броју 1, то је метрика боља.

Метрике које се анализирају приликом валидирања верзије решења су:

1. *Coverage* – односно покривеност представља пропорцију јединствених ставки које обучени модел може да препоручи у односу на укупни број јединствених ставки у скуповима података. Већа покривеност нам говори да модел препоручује више различитих ставки, уместо да свим корисницима препоручује само мањи проценат целог скупа ставки.

2. *Mean reciprocal rank at 25* – ова метрика означава способност модела да генерише релевантну препоруку на врху рангиране листе (од 25 понуђених препорука), при чему релевантна препорука представља препоруку за ставку са којом је корисник заиста интераговао.

3. *Normalized discounted cumulative gain at 5/10/25* – ова метрика показује колико добро модел рангира препоруке, где је K величина узорка од 5, 10 или 25 препорука.

4. *Precision at 5/10/25* – ова метрика показује колико препорука модела је релевантно на основу узорка од K (5, 10 или 25) препорука. *Amazon Personalize* израчунава ову метрику тако што број релевантних препорука од првих K препорука за сваког корисника у тестном сету подели са бројем K, а коначна метрика је просек за све кориснике у тестном сету. На пример, ако модел препоручује 10 ставки кориснику, и корисник интерагује са 3 од њих, прецизност на K је 3 тачно предвиђене ставке подељено са укупно 10 препоручених ставки, што даје рачуницу: 3 / 10 = 0.30.

Анализом метрика конкретне верзије решења које се могу видети на слици 2.3, може се утврдити да су метрике *Normalized discounted cumulative gain at 5/10/25* и *Mean reciprocal rank at 25* веома близу броју 1, што указује на квалитет обученог модела. Метрика *Coverage* има вредност 0.58, што указује да се више од пола свих ставки препоручује корисницима и та вредност би се могла још више унапредити повећањем параметра за истраживање нових ставки које утичу на то да се одговарајући проценат (10-15%) препорука састоји од нових ставки које нису још увек постале популарне.

Precision метрика има вредности које су мање у овој конкретној верзији решења, а главни разлог је тај што је скуп података над којим је обучен овај модел повучен са платформе на којој у том моменту није постојао систем за препоруку и дистрибуираност ставки по корисницима је била веома мала, односно корисници су у великом броју случајева своје активности на платформи сводили само на једну, до максимум 2 ставке.

Овим системом за препоруку ће бити унапређена активност корисника на платформи и на лакши начин ће долазити до ставки које су у складу са њиховим преференцама, тако да ће у будућности и сама материка прецизности бити унапређена.

Након завршене валидације верзије решења, решење је спремно за употребу и може се користити за генерисање препорука у реалном времену путем *API* позива или кроз *batch* процесирање. У случају препорука у реалном времену, потребно је направити *Amazon Personalize* кампању, помоћу које се преко *API* позива препоруке генеришу у реалном времену на основу тренутних корисничких интеракција. Решење које је имплементирано у пројекту који прати овај рад је генерисање препорука кроз *batch* процесирање које представља свакодневно периодично генерисање препорука на основу освежених података и биће детаљније објашњено у наредним поглављима.

2.8 Генерисање препорука кроз *batch* процесирање

Како се подаци о корисницима и интеракцијама мењају, а нове ставке појављују на платформама, важно је редовно ажурирати и обучавати модел како би препоруке остале релевантне и тачне. *Amazon Personalize* подржава аутоматско ажурирање модела на основу нових података, што обезбеђује да систем за препоруке увек пружа најбоље могуће резултате. Једно од ограничења је то што се аутоматско ажурирање може извршити максимално једном дневно, а услед великог броја интеракција се јавља потреба за чешћим ажурирањем препорука. Из тог разлога је имплементиран аутоматизовани систем који може већи број пута у току дана ажурирати дате скупове података, извршити нова обучавања модела и враћање резултата кроз *batch* процесирање, а детаљније ће имплементација бити објашњена у оквиру поглавља за сервис *Step Functions*.

Након успешног формирања верзије решења система за препоруку позива се *Lambda* функција помоћу које је имплементирано *batch* процесирање користећи клијента за *Amazon Personalize* сервис. Помоћу клијента се позива метода *create_batch_inference_job* којом се спецификује локација на *Amazon S3* где се налази датотека са идентификационим бројевима свих корисника који су имали интеракције и за које је потребно направити излаз са препорученим ставкама. У оквиру кôда 2.8 се може видети дефиниција дате методе, где се поред локације датотека са идентификационим бројевима свих корисника налази и путања на којој ће бити датотека са корисничким персонализованим препорукама, као и *AWS IAM* улога којом су регулисане дозволе за приступ сервисима *Amazon Personalize* и *Amazon S3*.

```
1 def create_batch_inference_job_user_personalization(
    solutionVersionArn, jobName):
2     response = LOADER.personalize_cli.create_batch_inference_job
        (
3         solutionVersionArn=solutionVersionArn,
4         jobName=jobName,
5         jobInput={
6             "s3DataSource": {
```

```
7                 "path": "s3://project-cw-rec-sys-personalize-
                     inputbucket/
                     batch_segment_inference_jobs_input_output/
                     users_ids_input/"
8             }
9         },
10        jobOutput={
11            "s3DataDestination": {
12                "path": "s3://project-cw-rec-sys-personalize-
                     inputbucket/
                     batch_segment_inference_jobs_input_output/
13                    users_personalization_recommendations_output
                         /"
14            }
15        },
16        roleArn="arn:aws:iam::110619981234:role/project-cw-
             casino-rec-sys-CreateBatchInferenceRole-1LJY9EEKX4PEK
             "
17    )
18
19    return response
```

Кôд 2.8: Метода за *batch* процесирање

Пример једног *JSON* реда препорука за корисника из датотеке која представља излаз поменуте методе *create_ batch_ inference_ job* се може видети у оквиру кôда на 2.9.

```
{
    "input": {
        "userId": "eddb8de4-afec-4ac6-8061-66047cb1d174"
    },    "output": {
        "recommendedItems": [
            "9188","4573","8602","1134","9187","3293","8703","
               424","407","399","9172","4815","1752"
            ,"4025","7923","9327","9186","6681","2189","7482","
               3257","5086","405","2885","8502"
        ],        "scores": [
            0.3272924,0.3198537,0.0981209,0.0390863,0.0145454
            ,0.0140903,0.0135336,0.0125328,0.0108843,0.0108725
            ,0.0097366,0.0070035,0.0066291,0.0064128,0.0054717
            ,0.0054625,0.0050756,0.0048714,0.004753,0.0044331
            ,0.0040509,0.0038266,0.0030338,0.0028515,0.0025298
        ]
    },    "error": null
}
```

Кôд 2.9: Пример препорука за једног корисника

Поред листе препоручених ставки се налазе и скорови сваке препоруке за конкретног корисника, где већи скор препоруке указује на већу шансу да је нешто слично или исто већ интераговано од стране тог корисника.

2.9 Чување препорука у *NoSQL* бази *Amazon DynamoDB*

Након што су препоруке генерисане и смештене на *Amazon S3*, први наредни корак представља архивирање датих препорука на посебној путањи у оквиру *Amazon S3* сервиса како би се чувала историја свих препорука. Архивирање препорука се ради из разлога да бисмо у будућности могли да радимо аналитику и поређење како су генерисане препоруке утицале на понашање и нове интеракције корисника. Аналитика такође може дати бољи поглед на

унапређење платформе, односно увид у повећање ангажованости корисника, корисничке лојалности и промовисање нових ставки.

Архивирање генерисаних препорука са једне на другу *Amazon S3* локацију се ради помоћу *Amazon S3* клијента и методе `copy_object`, чији позив се може видети у оквиру кôда 2.10. Овој методи се прослеђује параметар `Bucket`, који представља циљни фолдер у који ће се прекопирати датотека са генерисаним препорукама, параметар `CopySource` представља апсолутну путању изворне генерисане датотеке, а параметар `Key` представља нову путању до датотеке где ће се прекопирати и архивирати изворна датотека. Нова путања датотеке ће садржати партиције које се односе на годину, месец и дан када је формирана датотека како би се у будућности лакше извршиле потребне провере како су дате препоруке утицале на кориснике у одговарајућим временским периодима.

```
1 def save_personalize_recommendations(s3_target_bucket,file_key,
    partition_year,partition_month,partition_day):
2
3     file_name = file_key.split("/")[-1]
4     personalize_model_type = file_key.split("/")[-2]
5     new_file_key=f"
        batch_segment_inference_jobs_recommendations_history/{
        personalize_model_type}/{partition_year}/{partition_month
        }/{partition_day}/{file_name}"
6
7
8     s3_client.copy_object(Bucket=s3_target_bucket, CopySource=
        s3_target_bucket+"/"+file_key, Key=new_file_key)
```

Кôд 2.10: Архивирање генерисаних препорука

Након архивирања препорука, а пре њиховог слања на клијентске платформе преко *API* позива, долази на ред чување тренутних препорука за сваког корисника у *NoSQL* базу *Amazon DynamoDB*. Разлози за избор овог сервиса су брзина, флексибилност, скалабилност при читању и писању, као и потреба за кључ-вредност типом складишта за кориснике и њихове препоруке.

Упис корисника и њихових препорука се ради у оквиру *AWS Glue* посла, који је имплементиран тако да учита генерисану датотеку са корисничким

препорукама, затим итерира кроз датотеку тако што дохвата корисника по корисника и након тога уз помоћ *Amazon DynamoDB* клијента врши упис у *Amazon DynamoDB* табелу која је направљена за чување корисника и њихових тренутних препорука које ће даље бити испоручене на клијентску платформу.

Имплементирана метода за упис корисничких препорука у *Amazon DynamoDB* табелу се може видети у оквиру кôда 2.11. *Amazon DynamoDB* клијент позива методу `update_item`, која функционише на принципу команде `UPSERT`, односно врши упис уколико дати корисник са његовим препорукама не постоји у табели, а освежава корисничке препоруке са новим препорукама уколико корисник већ постоји у табели. Методи се прослеђује параметар који спецификује табелу у коју се врши упис, затим атрибути, односно кључеви који се уписују у табелу, као и повратна вредност која приказује ново стање реда у табели. [1]

```
1  def dynamodb_put_item_user_personalized_recommendations(
       line_decoded_json,campaign_casino_games):
2      user_id = line_decoded_json["input"]["userId"]
3      rec_items = line_decoded_json["output"]["recommendedItems"]
4      for casino_game_id, casino_game_position in
           campaign_casino_games:
5          rec_items.insert(casino_game_position, casino_game_id)
6          rec_items.pop()
7      response = dynamodb.update_item(
8          TableName = "usersPersonalizedRecommendations",
9          ExpressionAttributeNames={
10             "#RI": "recommendedItems",
11         },
12         ExpressionAttributeValues={
13             ":ri": {
14                 "S": str(rec_items),
15             },
16         },
17         Key={
18             "userId": {
19                 "S": user_id
20             }
```

```
21          },
22          ReturnValues="ALL_NEW",
23          UpdateExpression="SET #RI = :ri"
24      )
```

Кôд 2.11: Упис препорука у *Amazon DynamoDB* табелу

2.10 Слање препорука на клијентску платформу

Један од великих бенефита овог сервиса је инкрементални извоз података из *Amazon DynamoDB* табела, односно праћење промена за сваки ред табеле. Уз помоћ овога, клијентској платформи се шаљу само препоруке за нове кориснике и за кориснике којима су препоруке измењене у односу на последње слање. На тај начин се убрзава процес слања података за платформе које имају огроман број корисника, и олакшава се клијентској платформи која врши приказивање послатих препорука, тако што преко *API* позива добијају само нове податке.

Имплементирана метода за инкрементални извоз из *Amazon DynamoDB* табеле се може видети у оквиру кôда 2.12. Помоћу клијента *Amazon DynamoDB* сервиса се позива метода за инкрементални извоз података, чији је назив `export_table_to_point_in_time`. Овој методи се прослеђују параметри који садрже информације на коју *Amazon S3* путању вршимо извоз одговарајуће табеле, затим формат извезене датотеке (*JSON/CSV*) и параметар који одређује временски период у оквиру којег се дохватају промене направљене над *Amazon DynamoDB* табелом. `ExportViewType` параметар спецификује изглед измењеног реда, односно да ли се добија приказ старе и нове вредности датог реда који је измењен у табели или само приказ нове вредности, што је и спецификовано параметером *NEW_IMAGE*.

```
1  def create_dynamodb_export(dynamodb_table_arn, s3_bucket_name,
       s3_bucket_prefix, export_start_datetime_format):
2
3      to_time = datetime.today()
4
5      export_response = LOADER.dynamodb_cli.
           export_table_to_point_in_time(
6                  TableArn=dynamodb_table_arn,
7                  S3Bucket=s3_bucket_name,
8                  S3Prefix = s3_bucket_prefix,
9                  ExportFormat="DYNAMODB_JSON",
10                 ExportType="INCREMENTAL_EXPORT",
11                 IncrementalExportSpecification={
```

```
12                      "ExportFromTime":
                            export_start_datetime_format,
13                      "ExportToTime": to_time,
14                      "ExportViewType": "NEW_IMAGE"
15                  }
16              )
17
18      return export_response["ExportDescription"]["ExportArn"]
```

Кôд 2.12: Инкрементални извоз података из *Amazon DynamoDB* табеле

Након што је извршен инкрементални извоз нових података из *Amazon DynamoDB* табеле на *Amazon S3* локацију, долази на ред слање корисничких персонализованих препорука на клијентску платформу користећи њихов *API endpoint*.

Процес слања корисничких препорука на клијентску платформу се врши помоћу *AWS Glue* посла и покреће се након што се заврши инкрементални извоз препорука.

Прво се позива метода у оквиру кôда 2.13 која итерира кроз инкрементално извезену датотеку и пролази линију по линију те датотеке и трансформише у *dictionary* објекат који одговара формату који се очекује са стране клијентског *API*-а. Тај формат се састоји од идентификатора корисника и листе препорука за тог конкретног корисника, при чему се број препорука у листи спецификује током креирања решења *Amazon Personalize* модела. Након што се свака линија трансформише тако да одговара очекиваном формату клијентског *API*-а, оне се смештају у листу, која представља комад (*chunk*) препорука. Сваки тај комад у овом конкретном решењу се састоји од по 5000 корисника и њихових x препорука, које се шаље у једном *API* позиву и на тај начин се већи број препорука шаље у мањем броју *API* позива. Величина овог комада (*chunk*-а) представља цифру која се договори са клијентском страном у складу са подешавањима њиховог *API endpoint*-а.

```
1  def iterating_s3_object_file(file_object):
2      cwclient_api_key, cwclient_api_url = get_cwclient_api_secret
           ()
3      record_num = 0
4      batch_size = 5000
```

```
5       gzipped = GzipFile(None, "rb", fileobj=file_object["Body"])
6       data = TextIOWrapper(gzipped)
7       chunk_data = []
8
9       for line in data:
10          try:
11              record_num += 1
12              transformed_line =
                    transform_incremental_line_for_api(line)
13              chunk_data.append(transformed_line)
14
15          except ValueError:  # includes JSONDecodeError
16              print("ValueError: json.loads has failed, reason of
                    error: ", str(ValueError))
17              print("ValueError: json.loads has failed, current
                    row value: ", line)
18
19          if record_num % batch_size == 0:
20              print("5000 rows!")
21
22              try:
23                  cwclient_api_post_request(chunk_data,
                        cwclient_api_key, cwclient_api_url)
24                  print("chunk_data sent: ", record_num)
25              except Exception as error:
26                  message_error  = "Exception/Error:
                        cwclient_api_post_request failed. reason:" +
                        str(error) + "\nTraceback: " + traceback.
                        format_exc()
27                  message_subject= "[personalize-send-
                        recommendations-to-cwclient-api - Error in
                        cwclient_api_post_request]"
28                  print(message_error)
29                  send_email_via_sns(message_error,message_subject
                        )
30              chunk_data = []
31
```

```
32      if len(chunk_data) > 0:
33          print("last <5000 rows: ")
34          print(len(chunk_data))
35          try:
36              cwclient_api_post_request(chunk_data,
                    cwclient_api_key, cwclient_api_url)
37              print("chunk_data sent: ", record_num)
38          except Exception as error:
39              message_error  = "Exception/Error: LAST
                    cwclient_api_post_request failed. reason:" + str(
                    error) + "\nTraceback: " + traceback.format_exc()
40              message_subject= "[personalize-send-recommendations-
                    to-cwclient-api - Error in
                    cwclient_api_post_request]"
41              print(message_error)
42              send_email_via_sns(message_error,message_subject)
```

Кôд 2.13: Припрема корисничких препорука за слање на клијентски *API endpoint*

Процес трансформисања редова датотеке који су у *JSON* формату се врши у методи у оквиру кôда 2.14 и излаз дате методе представља *dictionary* објекат који одговара формату са стране клијентског *API*-а.

```
1  def transform_incremental_line_for_api(line):
2      json_line = json.loads(line)
3      user_uuid = json_line["NewImage"]["userId"]["S"]
4      recommended_items = json_line["NewImage"]["recommendedItems"]
           ["S"]
5
6      # Remove '[' and ']' and split by ', ' to get individual IDs
           as strings
7      ids_list_str = recommended_items.strip('[]').split(', ')
8
9      # Convert each ID from string to integer after removing the
           single quotes
10     ids_list_int = [int(id_str.strip("'")) for id_str in
           ids_list_str]
11
12
13     json_line = {"userUuid": user_uuid, "games": ids_list_int}
14     return json_line
```

Кôд 2.14: Трансформисање редова датотеке за слање на *API endpoint*

Креденцијали *API endpoint* се чувају помоћу сервиса *AWS Secrets Manager*, који се користи за централизовано и сигурно управљање креденцијалима. У оквиру *AWS Glue* посла се на сигуран начин довлаче креденцијали користећи *Secret Manager* клијента и методу *get_ secret_ value*, као што се може видети у оквиру кôда 2.15.

```
def get_cwclient_api_secret():
    secret_arn = args['cwclient_rec_api_secret_arn']
    region_name = "eu-central-1"

    # Create a Secrets Manager client
    session = boto3.session.Session()
    client = session.client(
        service_name='secretsmanager',
        region_name=region_name
    )

    secret_val = client.get_secret_value(
        SecretId=secret_arn
    )

    print(secret_val)
    sec_val_str = secret_val['SecretString']
    json_object = json.loads(sec_val_str)

    return json_object["api_key"], json_object["api_url"]
```

Кôд 2.15: Довлачење креденцијала са *Secret Manager*

Након што су прочитани креденцијали за испоручени клијентски *API endpoint*, последњи корак представља слање формираних скупова препорука на клијентску платформу и за то се користи метода *post*, учитане библиотеке *requests*, као што се може видети у оквиру кôда 2.16.

```
def cwclient_api_post_request(chunk_data,cwclient_api_key,
    cwclient_api_url):
    HEADERS = {"x-api-key": cwclient_api_key,"Content-Type": "
        application/json"}
```

```
3
4       post_request_successful = False
5       num_of_tries = 0
6       max_num_of_tries = 3
7
8       while post_request_successful != True:
9           r = requests.post(url = cwclient_api_url, headers=
                HEADERS, data=json.dumps(chunk_data))
10          num_of_tries = num_of_tries + 1
11
12          if r.status_code == 200:
13              post_request_successful = True
14              response_text = r.text
15              print(f"Post request successful: {response_text}")
16              break
17          else:
18              print(f"Request failed with status code {r.
                    status_code}")
19
20              if num_of_tries > max_num_of_tries:
21                  print("Maximum number of post request tries
                        exceeded.")
22                  raise Exception('Maximum number of post request
                        tries exceeded')
```

Кôд 2.16: Слање препорука на клијентски *API*

Након сваког слања скупа препорука проверава се статус *post* захтева и уколико враћени одговор нема вредност 200, значи да је дошло до неке грешке приликом слања захтева. У том случају се понавља слање захтева и максималан број покушаја је 4. Уколико ни након 4 покушаја не дође до успешног слања захтева, онда се подиже изузетак *Exception* који је ухваћен у претходној методи, након чега се исписује порука о грешкама које су настале и шаље се мејл преко *Simple Notification Service*-а као вид алармирања у случају грешке. Систем за алармирање је веома битан како би администратори што пре постали свесни да постоји одговарајући проблем са испорученим *API endpoint*-ом. Имплементација методе за слање мејла у случају грешке се може видети у оквиру кôда 2.17.

```
def send_email_via_sns(message_info, message_subject):

    sns_client = boto3.client("sns", region_name="eu-central-1")
    personalizer_sns_topic = args["sns_topic_arn"]

    try:
        if message_info is not None:
            response = sns_client.publish(
                TopicArn= personalizer_sns_topic,
                Message=message_info,
                Subject=message_subject[0:99],
            )
        else:
            print("Error: message_info is None.")
    except Exception as e:
        print(e.response["Error"]["Message"])
        raise Exception("send_email_via_sns")
    else:
        if response and "MessageId" in response:
            print("Email sent! Message ID:", response["MessageId
                "])
        else:
            print("Email sent, but no Message ID returned.")
```

Кôд 2.17: Слање мејла у случају грешке са клијентским *API endpoint*-ом

Уколико је повратна вредност захтева једнака 200, наставља се испоручивање нових скупова препорука док се не пошаљу за све кориснике који су активни у том моменту. Након што су послати сви подаци, клијентска платформа је освежена новим персонализованим корисничким препорукама и оне се неће мењати до наредног циклуса покретања система за препоруку који ће одрадити све претходно описане кораке и поновити слање нових препорука. Број циклуса покретања система за препоруку у току дана је ствар договора са клијентом и зависи од тога колико често клијентска платформа захтева освежене податке.

Глава 3

Аутоматизација система за препоруку

У овој глави ће бити описан начин на који је аутоматизован целокупни систем за препоруку уз помоћ сервиса *AWS Step Functions* који представља сервис за усклађивање и дефинисање сложених токова процеса.

3.1 *AWS Step Functions* сервис

AWS Step Functions је сервис који омогућава усклађивање послова помоћу визуелних радних токова. Уз помоћ овог сервиса, корисници могу лако координисати различите *AWS* сервисе у низ корака како би се формирале сложене апликације. Овај сервис пружа графички интерфејс који олакшава дизајнирање и надгледање радних токова, омогућавајући корисницима да брзо виде како се њихови процеси извршавају и идентификују потенцијалне проблеме.

Основне карактеристике *AWS Step Functions* укључују подршку за аутоматско понављање, управљање стањем и руковање грешкама, што поједностављује формирање комплексних апликација. Корисници могу дефинисати радне токове као стање машине користећи *Amazon States Language (ASL)*, што омогућава прецизну контролу над низом и условима извршења појединачних корака. [11]

Amazon States Language (ASL) је *JSON* базирани формат који се користи за дефинисање машине стања у *AWS Step Functions*. *ASL* омогућава корисницима да спецификују низ стања и транзиција које чине радни ток, укључујући

правила за руковање грешкама, контролу гранања и паралелно извршавање. Основни концепти *ASL* укључују:

1. Стања (*States*) – Јединице рада дефинисане машине које могу бити процеси као што су *AWS Lambda* функције, акције над подржаним *AWS* сервисима, паралелне гране, избори/провера вредности, чекања на извршавање.

2. Транзиције (*Transitions*) – Путање које повезују стања и одређују редослед њиховог извршавања.

3. Почетно стање (*StartAt*) – Одређује почетно стање у машини стања.

4. Крајње стање (*End*) – Одређује завршно стање у машини стања.

5. Коментар (*Comment*) – Опционо поље које пружа опис или објашњење машине стања.

AWS Step Functions има могућност интеграције са широким спектром *AWS* сервиса, као што су *AWS Lambda*, *Amazon S3*, и *Amazon DynamoDB*, као и са екстерним *API*-има, омогућујући флексибилност и скалабилност потребну за подршку различитим случајевима употребе датог сервиса. Уз помоћ овог сервиса, развојни тимови могу брзо имплементирати и променити своје апликације, смањујући време потребно за развој и одржавање сложених система.

На слици 3.1 се може видети дефиниција једне једноставне машине стања у оквиру конзоле сервиса *AWS Step Functions*.

```
{
  "Comment": "Simple State Machine example",
  "StartAt": "HelloWorld",
  "States": {
    "HelloWorld": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:110619981234:function:HelloWorldFunction",
      "Next": "ChoiceState"
    },
    "ChoiceState": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.statusCode",
          "NumericEquals": 200,
          "Next": "GoodbyeWorld"
        }
      ],
      "Default": "ErrorState"
    },
    "GoodbyeWorld": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:110619981234:function:GoodbyeWorldFunction",
      "End": true
    },
    "ErrorState": {
      "Type": "Fail",
      "Error": "ErrorOccurred",
      "Cause": "Invalid status code"
    }
  }
}
```

Слика 3.1: Једноставни пример машине стања из конзоле сервиса *AWS Step Functions*

Објашњење по корацима:

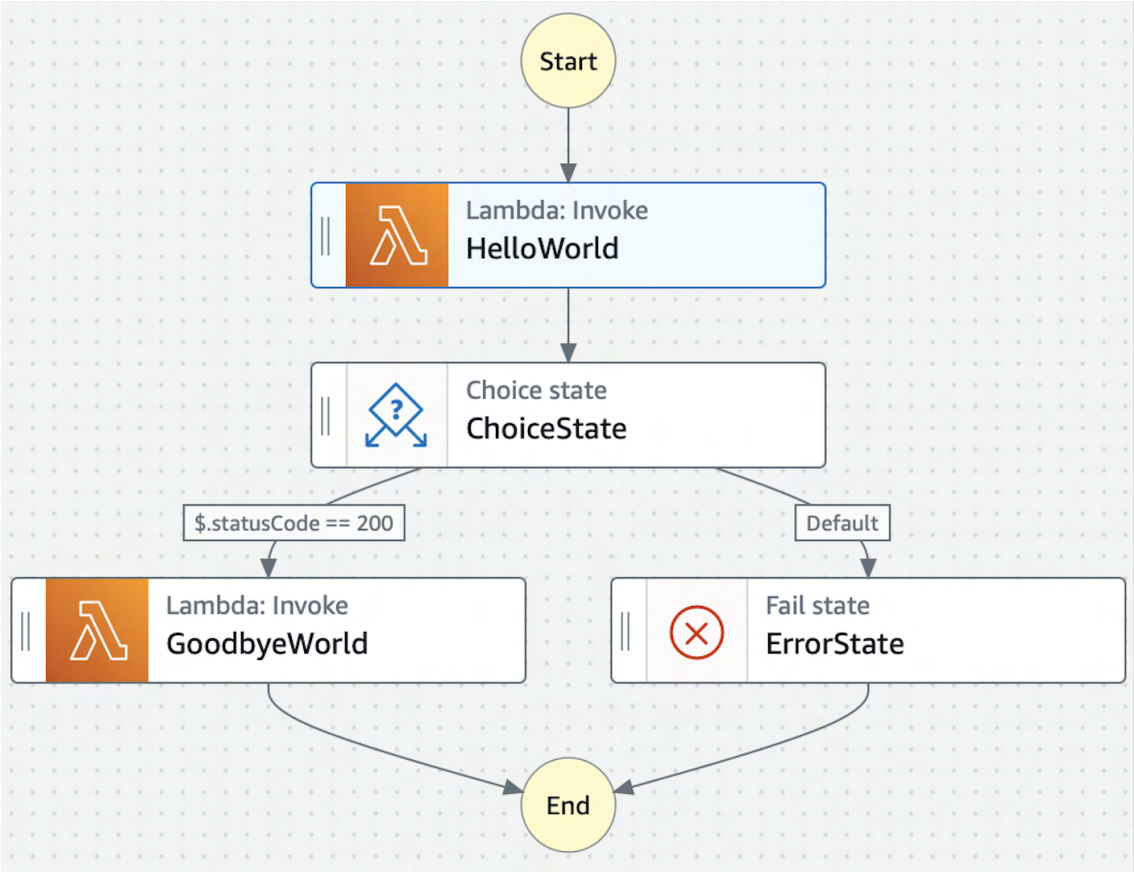
1. Стање *HelloWorld* – Тип овог стања је задатак (*Task*), који позива једноставну *AWS Lambda* функцију *HelloWorldFunction* која исписује поруку поздрава и има повратну вредност 200 уколико је све прошло без грешке.

2. Стање *ChoiceState* – Тип овог стања је избор (*Choice*), који проверава уз помоћ параметра *„$.statusCode"* да ли је повратна вредност претходно извршене функције 200, односно да ли је функција успешно завршена. Уколико јесте, радни ток прелази на стање *GoodbyeWorld*, а уколико није, онда прелази на стање *ErrorState*.

3. Стање *GoodbyeWorld* – Тип овог стања је задатак (*Task*), који позива једноставну *AWS Lambda* функцију *GoodbyeWorldFunction*. У овом стању је параметар *End* постављен на Тачно (*True*), тако да се након исправног извршавања овог задатка прелази у стање Крај (*End*).

4. Стање *ErrorState* – Тип овог стања је неуспешно извршавање (*Fail*) и оно указује на то да је претходна функција неуспешно извршена и да је дошло до одговарајуће грешке која је ухваћена као изузетак. Анализом овог стања се може проверити разлог због којег претходна функција није успешно извршена и предузети потребни кораци у исправљању грешке.

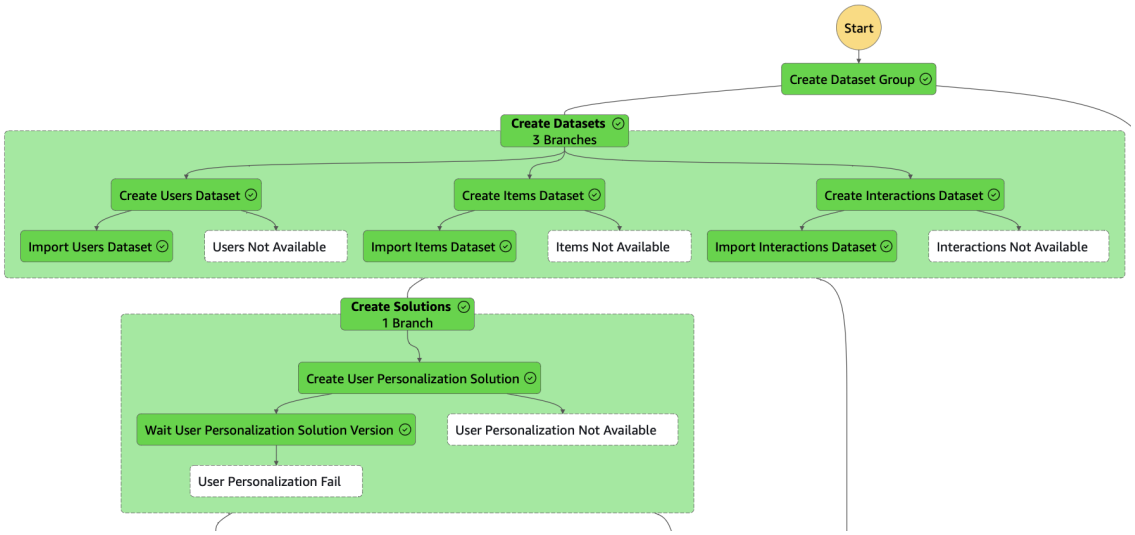
На наредној слици 3.2 се може видети и графички приказ описане машине стања.



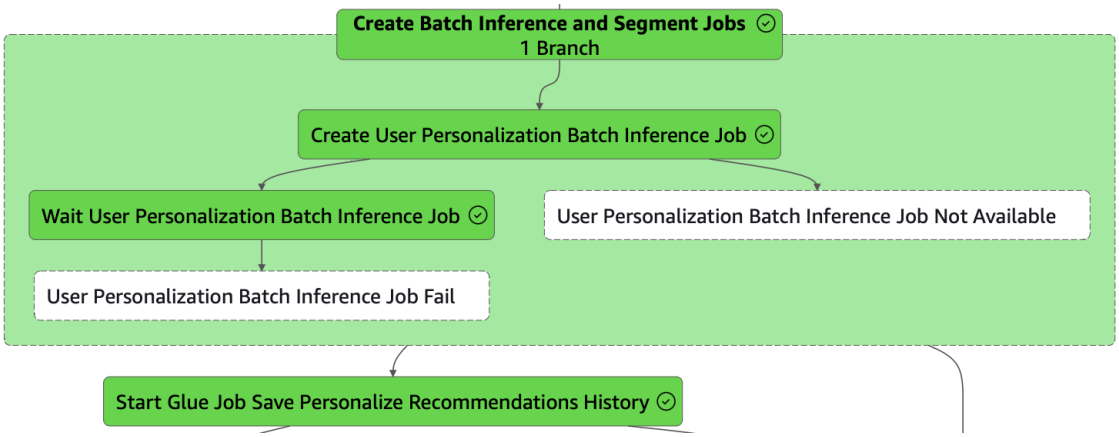
Слика 3.2: Графички приказ описане *AWS Step Functions* машине стања

3.2 *AWS Step Functions* машина стања имплементираног система за препоруку

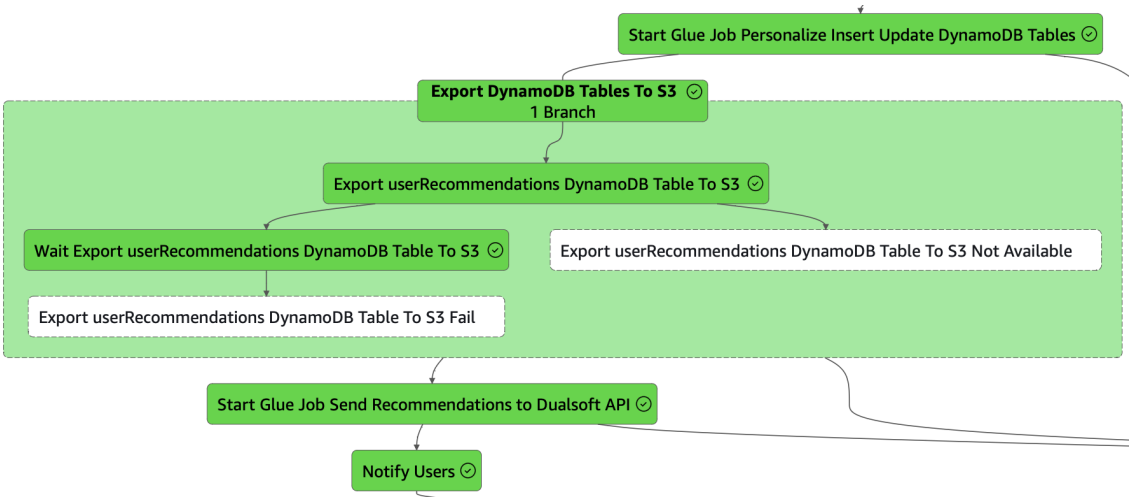
На сликама 3.3, 3.4 и 3.5 се може видети како је кроз дефинисану *AWS Step Functions* машину стања (*State Machine*) аутоматизован целокупни процес извршавања система за препоруку који је описан у другој глави. На слици се такође може видети како су стања међусобно повезана и да је покретање наредног стања условљено успешним извршавањем претходног.



Слика 3.3: *AWS Step Functions* машина стања – први део графичког приказа



Слика 3.4: *AWS Step Functions* машина стања – други део графичког приказа



Слика 3.5: *AWS Step Functions* машина стања – трећи део графичког приказа

У дефинисаној машини стања, вредности излаза једног стања се аутоматски прослеђују као улази у следећа стања у радном току. Овај процес се назива `преношење података између стања` и за размену података се користи *JSON* објекат што омогућава флексибилност и лакоћу манипулације подацима. Помоћу овог система се на једноставан начин прослеђују информације о формираним ресурсима из једних у друга стања, што помаже при аутоматизацији целог система.

У сваком од великих зелених правоуганика на претходним сликама могу се видети мањи правоугаоници који представљају *Lambda* функције које асинхроно извршавају описану логику целог система корак по корак. Након сваке функције која врши формирање одговарајућег ресурса или извршавање услуге неког *AWS* сервиса, почевши од прављења групе скупова података, па све до извоза промена у *DynamoDB* табелама, постоји функција која проверава да ли је претходно формирање ресурса или услуге успешно завршено и да ли се може наставити извршавање *AWS Step Functions* машине стања. Дате провере нису могле бити извршене у оквиру самих *Lambda* функција јер су оне временски ограничене на максимално извршавање од 15 минута, а формирање нове верзије решења у зависности од нове количине података може трајати и до неколико сати. Такође свака целина поред функција за асинхроно покретање формирања ресурса и чекање на успешно формирање има и функције које прате да ли је дошло до одговарајуће грешке приликом извршавања тих функција и у том случају се врши транзиција стања на Неу-

спешно (*Fail*) стање које се налази у доњем десном углу и у оквиру тог стања се добија мејл нотификација да је дошло до одговарајуће грешке, како би се што пре отклонио проблем. Поред асинхроних позивања *Lambda* функција, виде се и синхрона извршавања *AWS Glue* послова, где се у наредно стање машине стања прелази тек након што се одговарајући посао успешно изврши.

Глава 4

Организација пројекта кроз Инфраструктуру као кôд

Целокупно решење система за препоруку је имплементирано кроз Инфраструктура као кôд (*Infrastructure as Code – IaC*) помоћу сервиса *AWS CloudFormation* који омогућава корисницима да лако направе и управљају *AWS* ресурсима помоћу описане инфраструктуре као кôда. Помоћу овог сервиса, корисници могу дефинисати и имплементирати инфраструктуру за своје апликације користећи декларативне шаблоне у *YAML* или *JSON* формату. [7]

4.1 *AWS CloudFormation* сервис

AWS Cloudformation је моћан сервис за аутоматизацију управљања *AWS* инфраструктуром. Коришћењем шаблона, корисници могу лако и брзо формирати, ажурирати и брисати *AWS* ресурсе, управљајући својом инфраструктуром као кôдом. Ово не само да повећава ефикасност и смањује могућност грешака, већ поспешује модуларност кôда, што доводи до боље искоришћености кôда.

Поред шаблона који су поменути у уводу, основни концепти *AWS CloudFormation* сервиса су:

1. Стекови (*Stacks*) – Када корисник имплементира шаблон, *AWS CloudFormation* формира стек, који представља колекцију *AWS* ресурса дефинисаних у шаблону. Сваки стек може бити формиран, ажуриран или брисан као

јединствена јединица, што омогућава лако управљање инфраструктуром.

2. Управљање зависностима – *AWS CloudFormation* аутоматски управља зависностима између ресурса. На пример, ако један ресурс зависи од другог, сервис ће осигурати да су сви зависни ресурси правилно формирани у исправном редоследу.

3. *Rollback* – У случају да дође до грешке током формирања или ажурирања стека, сервис ће аутоматски вратити све промене (*Rollback*) како би осигурао да инфраструктура остане у стабилном стању.

4. *Cross-Stack References* – Омогућава референцирање ресурса из једног стека у другом, што олакшава поделу инфраструктуре на више делова који могу бити управљани и ажурирани независно.

4.2 Прављење *DynamoDB* табеле кроз *AWS CloudFormation* стек

Поменути концепти олакшавају организацију и аутоматизацију целокупног пројекта, а пример једног формираног ресурса кроз *AWS CloudFormation* стек је *DynamoDB* табела која ће садржати информације о корисничким идентификационим бројевима и листи препоручених ставки за тог корисника.

Као што се може видети у оквиру *CloudFormation* кôда 4.1, логички назив ресурса у *AWS CloudFormation* стеку је *usersPersRecommDynamoDBTable* и неопходно је да тај назив буде јединствен у самом стеку и да именовање буде такво да нам јасно упућује на који ресурс се односи. Основни параметри формираног ресурса су *Type* који се односи на тип *AWS* ресурса који се формира и параметар *Properties* који се односи на специфичне својства самог ресурса.

```
1 usersPersRecommDynamoDBTable:
2     Type: AWS::DynamoDB::Table
3     Properties:
4       DeletionProtectionEnabled: true
5       PointInTimeRecoverySpecification:
6         PointInTimeRecoveryEnabled: true
```

```
7       AttributeDefinitions:
8         -
9           AttributeName: "userId"
10          AttributeType: "S"
11      KeySchema:
12        -
13          AttributeName: "userId"
14          KeyType: "HASH"
15      ProvisionedThroughput:
16        ReadCapacityUnits: "5"
17        WriteCapacityUnits: "20"
18      TableName: "usersPersonalizedRecommendations"
```

Кôд 4.1: Прављење ресурса *DynamoDB* табеле кроз *AWS CloudFormation*

Друга својства ресурса која су попуњена су *TableName* који се односи на назив табеле. *PointInTimeRecoverySpecification* који се односи на омогућавање инкременталног извоза који је описан у другој глави и који се користи за праћење промена препорука од последњег извршавања система. *AttributeDefinitions* спецификује шему табеле, односно назив колона и типове података. У овом случају је то колона *userId*, која је типа *string*. Друга колона *recommendedItems* се аутоматски додаје приликом уноса редова кроз описани *AWS Glue* посао који спецификује као другу колону листу идентификационих ставки које представљају листу препорука и то се може видети на слици 4.1.

Слика 4.1: Пример реда из *DynamoDB* табеле

KeySchema спецификује примарни кључ табеле, а тип кључа се односи на то да ли је дата колона партициони (*partition key*) кључ или кључ за сортирање (*sort key*). *ProvisionedThroughput* је својство које се односи на спецификовање броја обезбеђених јединица читања и писања направљене табеле. Јединицама читања и писања се одређује укупан проток података, односно читања и писања који ће бити дозвољени у једној јединици времена (секунди).

Јединице за читање и писање могу бити следеће:

1. Једна јединица капацитета за читање = једно *strongly consistent* читање по секунди или два *eventually consistent* читања по секунди, за ставке величине до 4KB.

2. Једна јединица капацитета за писање = једно писање по секунди, за ставке величине до 1KB.

Када се врше *eventually consistent* читања из *DynamoDB* табеле, можда се неће прочитати промене у табели настале од стране недавно завршене операције писања, док то није случај код *strongly consistent* читања.

4.3 *AWS CloudFormation* у *CI/CD* процесима

Поред тога што је *AWS CloudFormation* моћан алат за аутоматизацију управљања *AWS* инфраструктуром, савршено се уклапа и у процесе континуиране интеграције (*Continuous Integration – CI*) и континуиране испоруке (*Continuous Delivery – CD*). *AWS CloudFormation* се интегрише са системима за верзионисање кôда као што су *AWS CodeCommit, Github.* Затим свака промена која се направи у репозиторијуму кôда која је повезана са *AWS CloudFormation* стековима окида *AWS CodePipeline* сервис помоћу којег је имплементиран *CI/CD.* Наредни корак је пролазак кроз фазе изградње (*Build*) и на крају испорука (*Deploy*) на одговарајући *AWS* налог.

Ова интеграција осигурава да се нови ресурси и исправке брзо и безбедно примењују на *AWS* окружење, чиме се побољшава брзина и поузданост процеса испоруке софтвера.

Глава 5

Демонстрација рада система за препоруку

Предложено решење система за препоруку које је описано у овом раду је примењено у индустрији игара на срећу, конкретно за слот и казино игре једне од кладионица на српском тржишту.

Наредним сликама екрана биће приказана веб страница клијентске компаније у оквиру које је имплементиран систем за препоруку.

5.1 Приказ препорука новим корисницима на веб страници

Након што се нови корисници региструју и валидирају на почетној страници кладионице, отвара им се страница где могу да бирају прозор са слот играма. Након отварања странице са слот играма, новим корисницима се у табу *PREPORUKA* приказују најпопуларније игрице на платформи као што се може видети на слици 5.1.

Слика 5.1: Препоруке најпопуларнијих слот игара за нове кориснике

Препоруке најпопуларнијих слот игара се добијају уз помоћ система за препоруку, тако што се након припрема скупова података за сервис *Amazon Personalize*, прави скуп најпопуларнијих игара на основу броја свих трансакција над слот играма. Тај скуп се приказује новим корисницима све док не почну са интеракцијама над препорученим играма. Након што нови корисници направе прве интеракције и трансакције над слот играма, ти подаци се у првој наредној итерацији система за препоруку укључују у скупове података за сервис *Amazon Personalize*, који ће на основу тих података, тренирањем модела машинског учења које је описано у претходним главама, дати релевантне препоруке које ће се убудуће приказивати тим корисницима.

Одређивање најпопуларнијих игара се извршава у *AWS Glue* послу чији је задатак припрема података система за препоруку, а део посла који се бави издвајањем 16 најпопуларнијих игара који ће се приказати тек регистрованим корисницима у оквиру прозора слот игара се ради помоћу наредног кôда 5.1.

```
1 query_cold_start_casino_games =
```

```
        """
        select  g.game_gid,
                g.game_provider,
                g.game_name,
                count(*) as broj_tran
        from    public.f_acc_tran a
            left join public.d_casino_game g on a.tran_gid = g.
               game_gid
        where g.game_gid is not null
                and tran_date_oid >= {date_oid_from}
                and tran_date_oid <= {date_oid_to}
                and tran_m_timestamp < {
                   job_start_epoch_timestamp}
        group   by 1,2,3
        order by broj_tran desc
        limit 16
        ;
        """
statement_name = "select_cold_start_casino_games"
query_cold_start_casino_games_parametrized =
   query_cold_start_casino_games.format(date_oid_from=
   date_oid_from_all_users_items_ids, date_oid_to=params[1],
   job_start_epoch_timestamp = params[6])
e
execute_statement_no_params_user_item_ids(redshift,
   query_cold_start_casino_games_parametrized,statement_name,"
   cold_start_casino_games",date_oid_from_all_users_items_ids)
```

Кôд 5.1: Одређивање 16 најпопуларнијих слот игара за новорегистроване кориснике

Резултат извршавања упита из кôда 5.1 се чува на одговарајућој *Amazon S3* путањи, одакле се уз помоћ *AWS Glue* посла врши упис најпопуларнијих игара у *DynamoDB* табелу за чување препорука под кључем *userId*, која има вредност *new_users_recommendations*, као што се може видети на наредној слици 5.2.

Кључ *recommendedItems* има вредност низа од 16 идентификационих бројева најпопуларнијих игара које се затим шаљу на клијентски *API endpoint*,

Слика 5.2: Најпопуларнијих 16 слот игара за нове кориснике

као што је описано у поглављу 2.10. Након што клијентска платформа прими послате препоруке за све кориснике, у случају да се на страници појави корисник који није имао до тог момента ниједну интеракцију са играма, за тог корисника се повлаче игре за кориснички *userId*, који има вредност *new_users_recommendations*. Након тога се том новом кориснику приказују најпопуларније игре као што се може видети на почетној страни клијентске веб странице са слике 5.1.

5.2 Приказ препорука редовним корисницима на веб страници

Након што корисник направи интеракцију, односно трансакцију над неком од игара, аналитички систем компаније која испоручује ово решење добија информације о тим трансакцијама уз помоћ *Kafka* платформе за стримовање садржаја у реалном времену, након чега се тај податак смешта у *Redshift* складиште података. Информације о новим корисницима, слот играма и њиховим трансакцијама над тим играма се у првој наредној итерацији система за препоруку повлаче и учитавају у скупове података које сервис *Amazon Personalize* користи за тренирање модела машинског учења (поглавље 2.2). На основу истренираног модела машинског учења добијају се персонализоване препоруке за кориснике (поглавље 2.8). Персонализоване препоруке представљају листе релеватних препорука које се на идентичан начин као што је описано у претходном случају чувају у *DynamoDB* табели, али у овом случају под одговарајућим корисничким идентификационим бројем. Затим следи слање препорука за све кориснике на клијентску платформу (поглавље 2.10). Приликом наредног уласка корисника на страницу са слот играма, који је раније имао интеракције са слот играма, веб страница повлачи послате препоруке на основу идентификационог броја играча. Препоруке се приказују под табом *PREPORUKA*, као што се може видети на слици 5.3.

Слика 5.3: Персонализоване препоруке слот игара конкретног постојећег корисника

На слици 5.3 се могу видети персонализоване препоруке за корисника чији је кориснички идентификациони број *d868c6e0-9703-4d84-b958-c36370cfc0ca*. Конкретне препоруке које се чувају у *DynamoDB* табели за тог корисника се могу видети на слици 5.4. Оне одговарају редоследу приказаних игара на веб страници.

Attribute name	Value	Type
userId - *Partition key*	d868c6e0-9703-4d84-b958-c36370cfc0ca	String
recommendedItems	['6736', '6702', '4021', '2261', '399', '358', '1974', '6521', '7810', '8825', '6695', '248', '1750', '3996', '385', '424', '396']	String

Слика 5.4: Персонализоване препоруке сачуване у *DynamoDB* табели

Глава 6

Закључак

У овом раду је истражена и анализирана имплементација система за препоруку на *AWS*-у, чиме су покривене кључне компоненте и технологије које омогућују формирање и функционалност оваквог система.

Као што је приказано у претходним главама, *AWS* нуди богат скуп услуга и сервиса које заједно чине снажну основу за развој, имплементацију и одржавање система за препоруку.

Такође је представљено на који начин се може аутоматизовати извршавање целог система користећи *AWS Step Functions* сервис, као и формирање и одржавање целе инфраструктуре кроз *AWS CloudFormation*.

Имплементирани систем за препоруку дизајниран је тако да буде изузетно прилагодљив и флексибилан, што омогућава његову примену у различитим бизнис случајевима и индустријским секторима. Оно што је најбитнији услов за прилагодљивост решења другим бизнис случајевима јесте приступ подацима, односно извору података из којег можемо извући податке који се тичу три основна скупа података који су описани у другој глави, а односе се на информације о корисницима, ставкама које се испоручују на платформи за коју се прави систем за препоруку и интеракције између корисника и датих ставки.

Системи за препоруке су данас незаобилазни у савременим апликацијама, јер знатно унапређују корисничко искуство кроз персонализоване препоруке. Поред тога што директно користе корисницима, ови системи пружају значајне предности и платформама које их користе. У е-трговини, персонализоване препоруке могу значајно повећати продају, мотивишући кориснике да купују производе који су у складу са њиховим интересовањима. На платформама

за емитовање садржаја, препорука релевантног садржаја продужава време проведено на платформи, што резултира већом лојалношћу и задржавањем корисника. Осим тога, системи за препоруке могу побољшати ефикасност оглашавања тако што циљају огласе на основу преференција и понашања корисника.

Библиографија

[1] Amazon Web Services. Amazon DynamoDB boto3 client update item, 2024. on-line at: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb/client/update_item.html.

[2] Amazon Web Services. Amazon DynamoDB service, 2024. on-line at: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

[3] Amazon Web Services. Amazon Personalize service, 2024. on-line at: https://docs.aws.amazon.com/personalize/latest/dg/what-is-personalize.html.

[4] Amazon Web Services. Amazon Personalize service datasets, 2024. on-line at: https://docs.aws.amazon.com/personalize/latest/dg/datasets.html.

[5] Amazon Web Services. Amazon Redshift service, 2024. on-line at: https://docs.aws.amazon.com/redshift/latest/mgmt/welcome.html.

[6] Amazon Web Services. Amazon S3 service, 2024. on-line at: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

[7] Amazon Web Services. AWS CloudFormation service, 2024. on-line at: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html.

[8] Amazon Web Services. AWS Glue Jobs versions, 2024. on-line at: https://docs.aws.amazon.com/glue/latest/dg/release-notes.html.

[9] Amazon Web Services. AWS Glue service, 2024. on-line at: https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html.

[10] Amazon Web Services. AWS Lambda service, 2024. on-line at: `https://docs.aws.amazon.com/lambda/latest/dg/welcome.html`.

[11] Amazon Web Services. AWS Step Functions service, 2024. on-line at: `https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html`.

Биографија аутора

Никола Мићић (*Лозница, 11. јун 1998.*) завршио Основну школу „Вук Караџић" у Лозници, 2013. године. Затим уписао Гимназију „Вук Караџић" у Лозници, коју је завршио 2017. године, а исте године уписао и Математички факултет у Београду, смер информатика. Основне студије факултета завршио 2022. године, а први семестар треће године завршио на факултету инжењерства на Универзитету Западне Атике у Атини, у оквиру Еразмус размене, која је трајала од октобра 2019. до фебруара 2020. године. Мастер студије смера информатика на Математичком факултету је уписао 2022. године.