Архитектурни пројекат

MindLink

Чланови тима:

Душан Марковић, 19212
Јован Станишић, 19369

Назив тима:

OG10D1

Датум: 21.1.2026

1. Контекст и циљ пројекта

MindLink представља online multiplayer игру која се заснива на игри Codenames. Игра се заснива на томе, да у једној партији која се игра, имамо 2 тима од по 2 играча, који се такмиче међусобно у томе да открију што више поља на табли, на основу вербалних смерница.

У принципу, један од играча има улогу да једном речју или кратком синтагмом, да смернице свом саиграчу како би он изабрао валидна поља на табли, што подразумева поља за тај тим и да избегне поља која су резервисана за други тим, а да притом не открије поље које означава моментални губитак партије.

2. Архитектурни захтеви

2.1 – Функционални захтеви

Регистровање корисника – подразумева креирање налога за корисника, како би могао да се придружи већ постојећој игри или креирање нове

Пријава и валидација корисника – све функционалности везане за омогућавање кориснику да приступи свом профилу, укључујући и проверу података који се притом уносе

Преглед соба – приказ свих постојећих соба којима корисник може да се прикључи

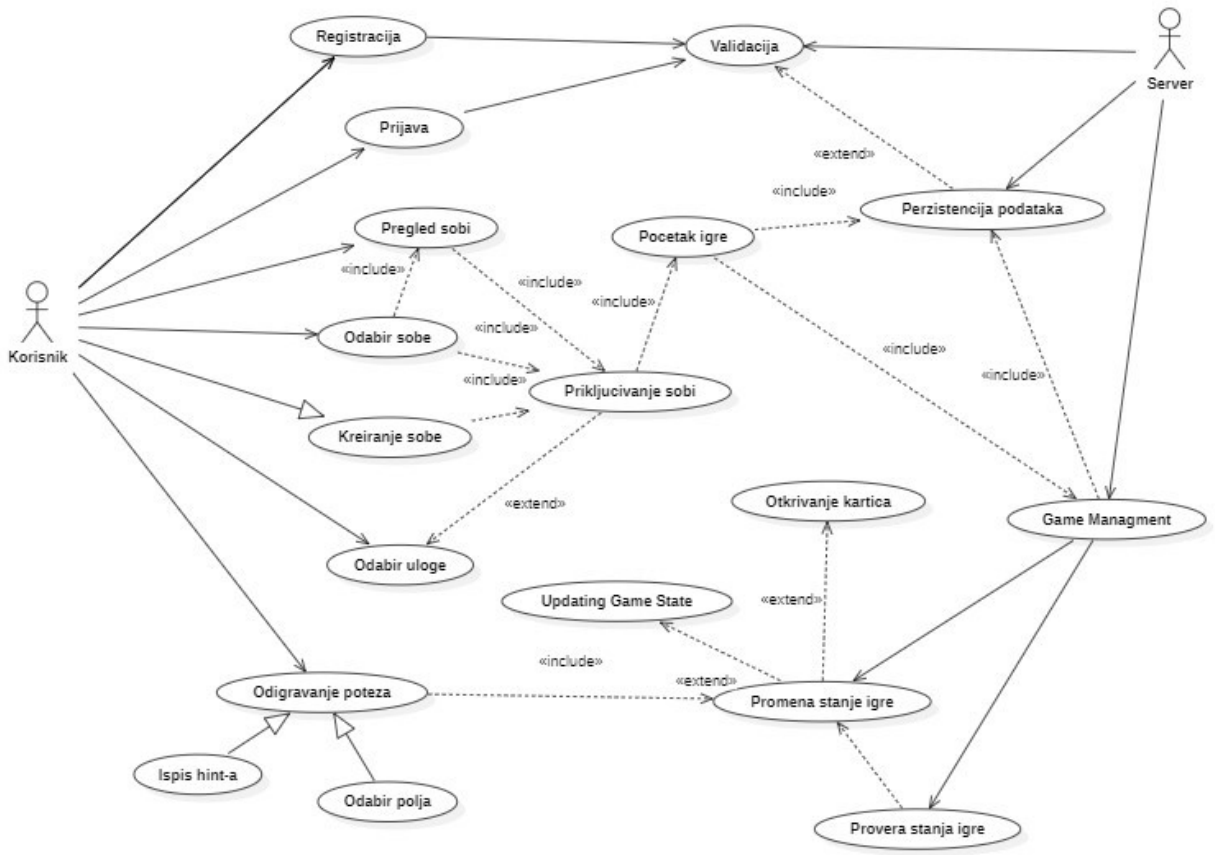
Креирање јавне собе – креирање собе која је слободна за приступ свим осталим корисницима без посебног начина приступа

Креирање приватне собе – потребно је да се обезбеди креирање посебне собе, којој могу да приступе и други корисници путем посебног кода који је везан искључиво за ту собу

Формирање игре – могућност да се постави почетно стање игре, што подразумева креирање тимова, доделу улога и одабир речника појмова који представљају предмет игре

Провера стања игре – након сваког одиграног потеза (одабира речи на основу hint-a), потребно је приказати како се поље променило у зависности од одабира карата од стране корисника; укључује и проверу за крај игре

Приказ резултата – на основу тога како се партија завршила и у којем случају, сваком тиму је потребно приказати њихов резултат и информацију о крају игре



2.2 – Нефункционални захтеви

Употребљивост – сама апликација није графички захтевна али је и поред тога потребно олакшати коришћење кориснику на начин да се лако упозна са правилима, док само коришћење треба да буде интуитивно

Перформансе – веб апликација треба да обезбеди што мање време одазива пошто је реч о систему који је real-time и користи тајмер који служи за одигравање потеза, и да пружа добар одговор на оптерећење

Доступност – апликација мора бити доступна у било ком тренутку сваком кориснику

Сигурност – профил корисника као и сви подаци које се за њега везују, морају бити заштићени приликом процеса аутентификације

Скалабилност – апликација треба да има добар оквир за проширење капацитета, што подразумева додавање нових сервера, а да притом добро подноси раст броја корисника или раст у броју тренутних партија које се играју

2.3 – Техничка ограничења

Техничка ограничења се највише односе на саму природу апликације која подразумева комуникацију више корисника који треба да буду повезани преко интернета па је због тога просто неопходно користити технологије које то и омогућавају. Ту спадају и принципи међусобне комуникације између корисника али и њихова интеракција са системом, који на основу тога мора да буде спреман и способан да подржи и синхрону и асинхрону комуникацију унутар апликације.

Како би се обезбедило фер одигравање партије, кориснику је потребно приказати само оно што је везано за његову улогу и профил.

2.4. – Пословна ограничења

Пословна ограничења за ову апликацију подразумевају само ограничења и оквире који се везују за правила саме игре, и на тај начин, корисник се придржава само већ унапред дефинисаног скупа акција које може да преузме.

3. Архитектурни дизајн

3.1 – Архитектурални обрасци

Приликом пројектовања Mindlink система фокус је био на јасној подели одговорности између компоненти, подршци за више истовремених партија игре и омогућавању комуникације у реалном времену између учесника. Због тога је архитектура обликована коришћењем више компатибилних образаца, који заједно омогућавају стабилан, проширив и разумљив систем.

3.1.1. Слојевита архитектура

Mindlink је реализован као вишеслојни систем у коме је свака целина задужена за тачно дефинисану улогу. Систем је подељен на три основна слоја:

Клијентски слој - Клијентски слој представља део система са којим корисник директно комуницира. Његова улога је да омогући интеракцију са игром, приказ тренутног стања партије и слање корисничких акција ка серверу. Овај слој не доноси одлуке о правилима игре, већ служи искључиво као интерфејс према кориснику.

Апликациони (серверски) слој - Серверски слој представља језгро система. У њему се налази комплетна логика игре, управљање партијама, обрада потеза и контрола тока игре. Такође, овај слој је одговоран за комуникацију са клијентима и координацију размене података, као и за приступ трајном складишту података.

Слој за чување података - Овај слој обухвата базу података у којој се налазе информације о корисницима, партијама и стањима игре. Иако је технички део истог система, он је логички изолован и доступан искључиво преко серверског слоја.

3.1.4. Управљање перзистенцијом путем Repository обрасца

Приступ бази података у Mindlink пројекту реализован је коришћењем Репоситору обрасца. Овај приступ омогућава да пословна логика буде независна од конкретне имплементације базе података.

Серверске компоненте комуницирају са базом преко јасно дефинисаних интерфејса, чиме је омогућен безбедан и контролисан приступ подацима. Посебна пажња посвећена је раду са више истовремених партија и очувању конзистентности података. За реализацију перзистенције коришћен је .NET Entity Framework који има могућност за олакшаним са креирањем и радом са базом података.

3.1.2. Организација по MVC принципу

Унутрашња структура система организована је по Model-View-Controller принципу како би се раздвојили подаци, приказ и логика обраде захтева.

Модели представљају основне ентитете игре и система, као што су корисници, партије и стање игре. Они описују структуру података и правила која важе над тим подацима.

View део система односи се на кориснички интерфејс и приказ информација играчима. Приказују се само подаци који су релевантни за конкретног корисника и његову улогу у партији.

Контролери повезују ове две целине тако што примају захтеве са клијентске стране, примењују одговарајућу пословну логику и врше измене над моделима, након чега враћају ажуриране податке клијенту.

3.1.3. Publish-Subscriber

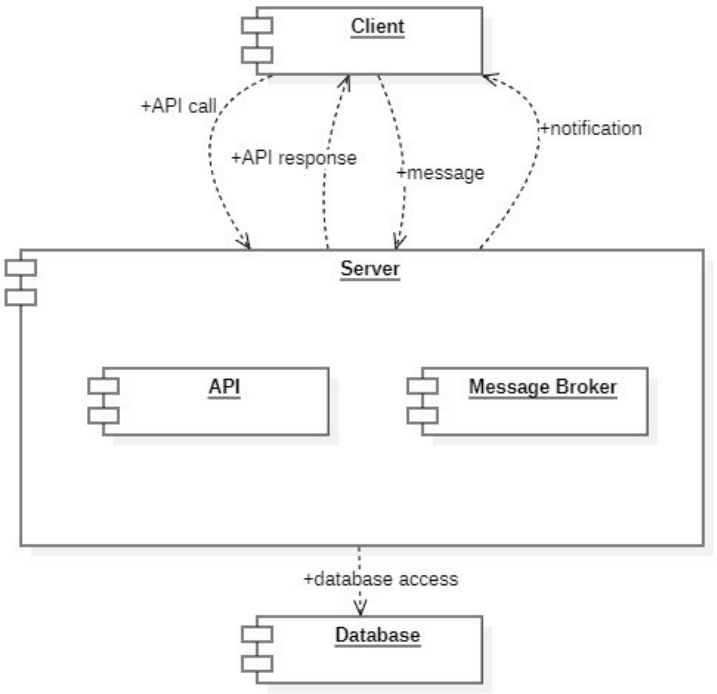
Због потребе да сви учесници у истој партији имају ажурирано стање игре у реалном времену, уведен је Publish-Subscriber образац комуникације. Клијенти се пријављују на догађаје везане за конкретну партију, док серверска страна објављује промене стања игре.

На овај начин се омогућава асинхрона размена информација, где сервер емитује промене свим релевантним клијентима, без потребе да сваки клијент константно проверава стање система.

3.2. Општи преглед архитектуре система

Архитектура Mindlink система заснива се на централизованом серверском моделу. Клијенти остварују комуникацију искључиво са серверском апликацијом, док директан приступ бази података није дозвољен.

Размена података између клијента и сервера врши се синхроно путем API позива, као и асинхроно путем механизама за размену порука. Серверска апликација обрађује све захтеве, извршава логику игре и по потреби врши читање или упис података у базу. Оваква организација омогућава поуздан рад система и једноставно проширење функционалности у будућности.

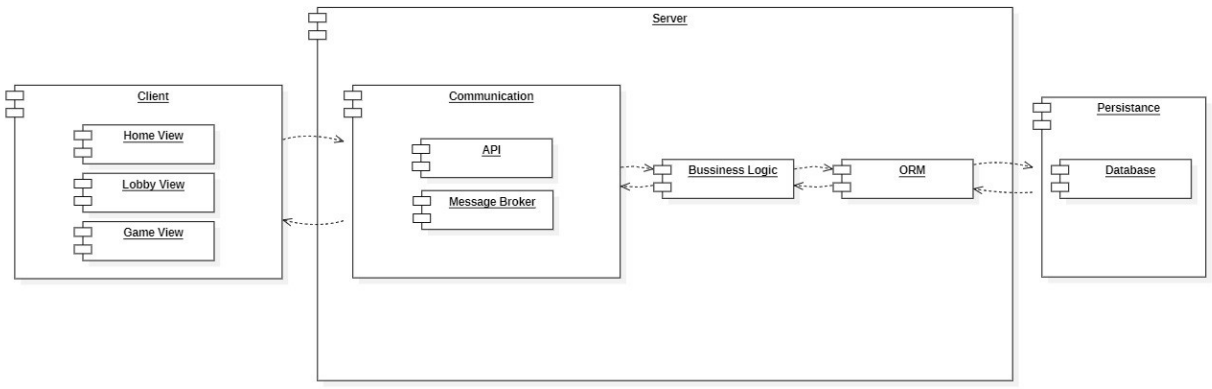


3.3 – Структурни поглед

На основу поделе из општег прегледа, видимо да је клијентски слој подељен на три главне секције од којих сваки представља поделу онога што корисник види а то је почетна страница, страница са собом, и наравно игра када она започне. Клијент комуницира са сервером преко комуникационог слоја.

Сервер садржи три компоненте где је једна задужена за комуникацију путем REST endpoint-а за HTTP захтеве и њихову обраду, пословну логику која представља сва правила и улоге унутар игре и њихово комуницирање и ORM (Entity Framework) који служи за аутоматску обраду миграција и мапирање објекта који се користе унутар апликације.

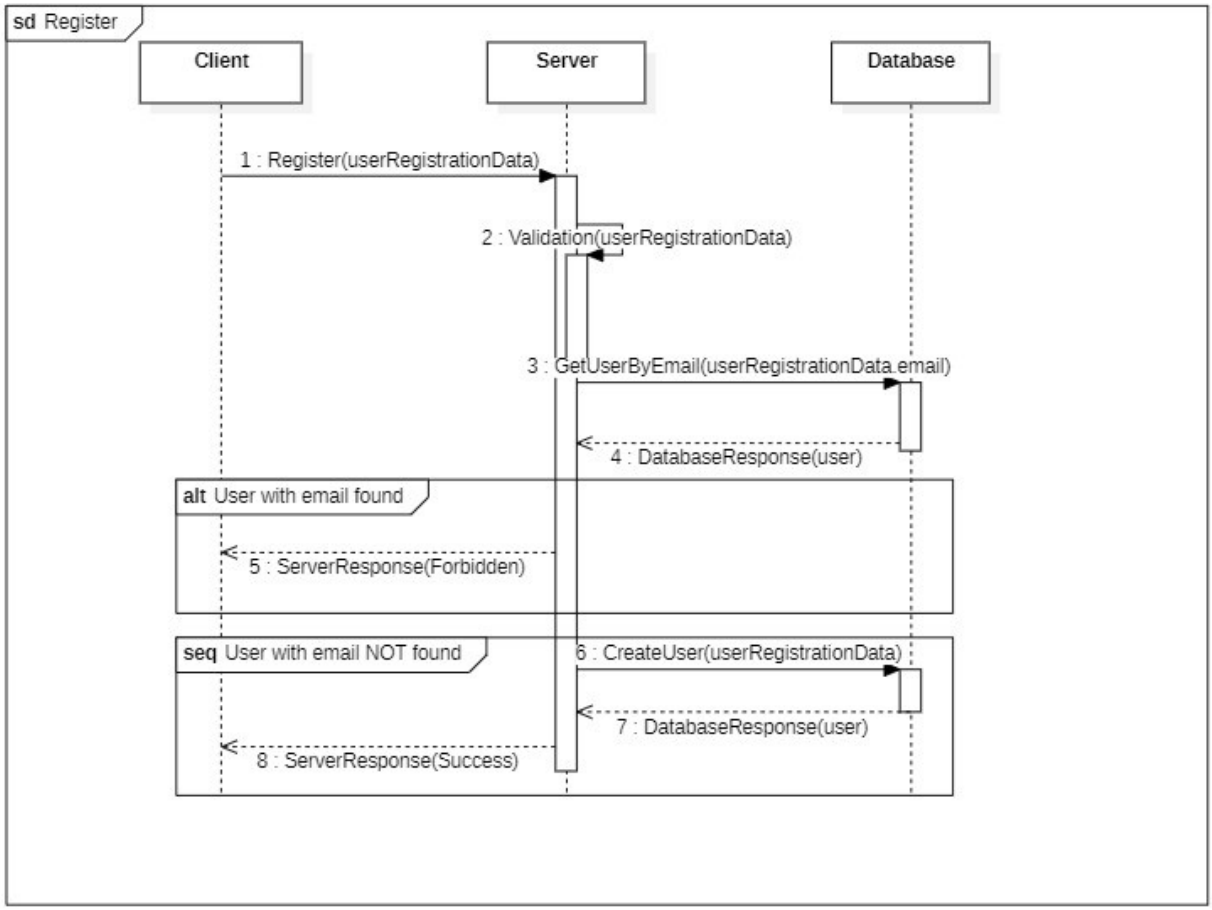
Последњи слој је перзистенција који чува све потребне податке а комуницира са пословним слојем путем ORM-а.



3.4 – Бихевијорални погледи

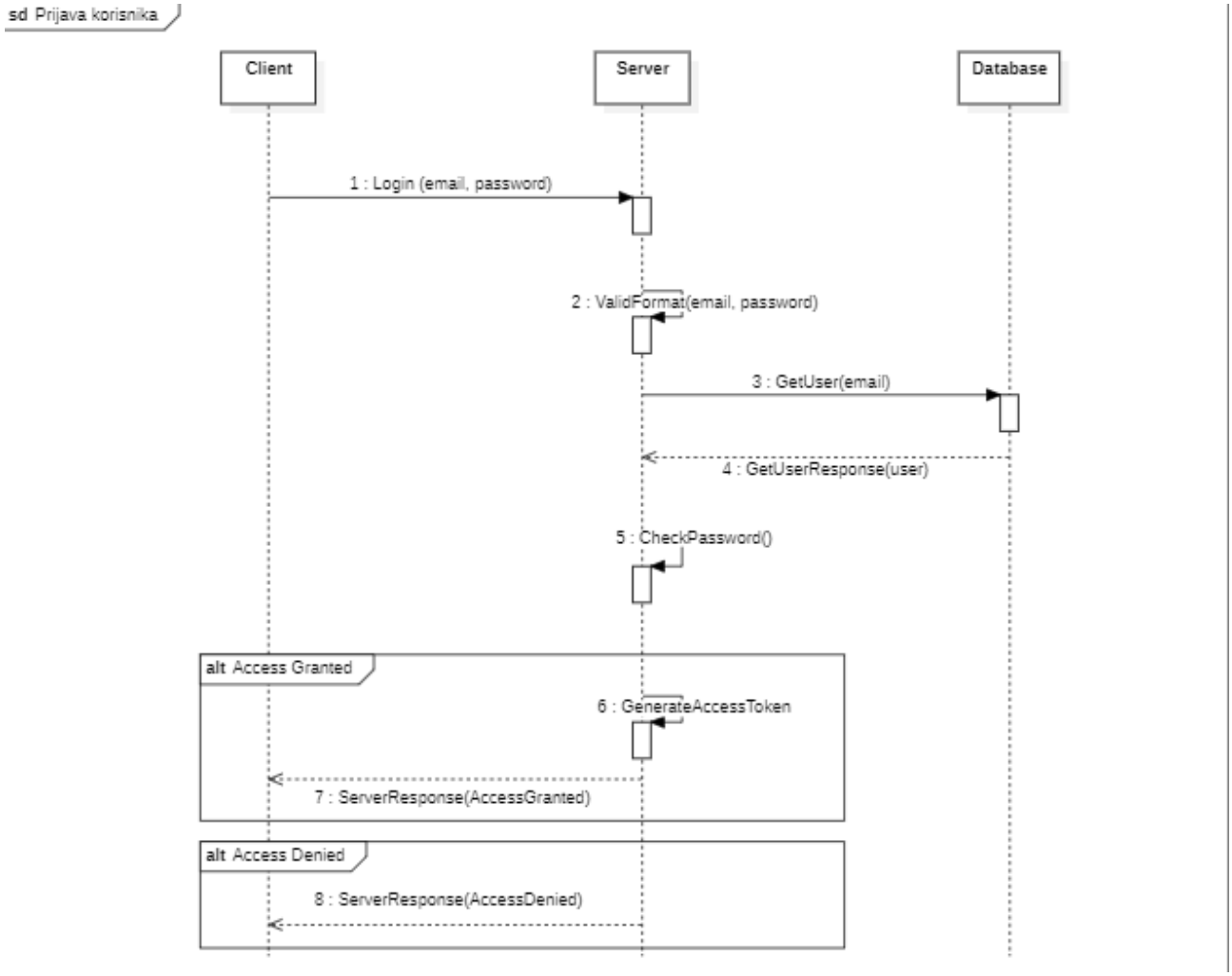
3.4.1 – Регистрација корисника

Приликом регистрације, серверу се шаљу подаци везани за корисника што укључује његову емаил и лозинку, након чега он врши проверу тих података (да ли је корисник већ регистрован на послату емаил адресу и ако није, да ли шифра одговара постојећим крединцијалима или не. Корисник се памти у делу за перзистенцију и након тога може да се пријави путем дела апликације за пријаву корисника.



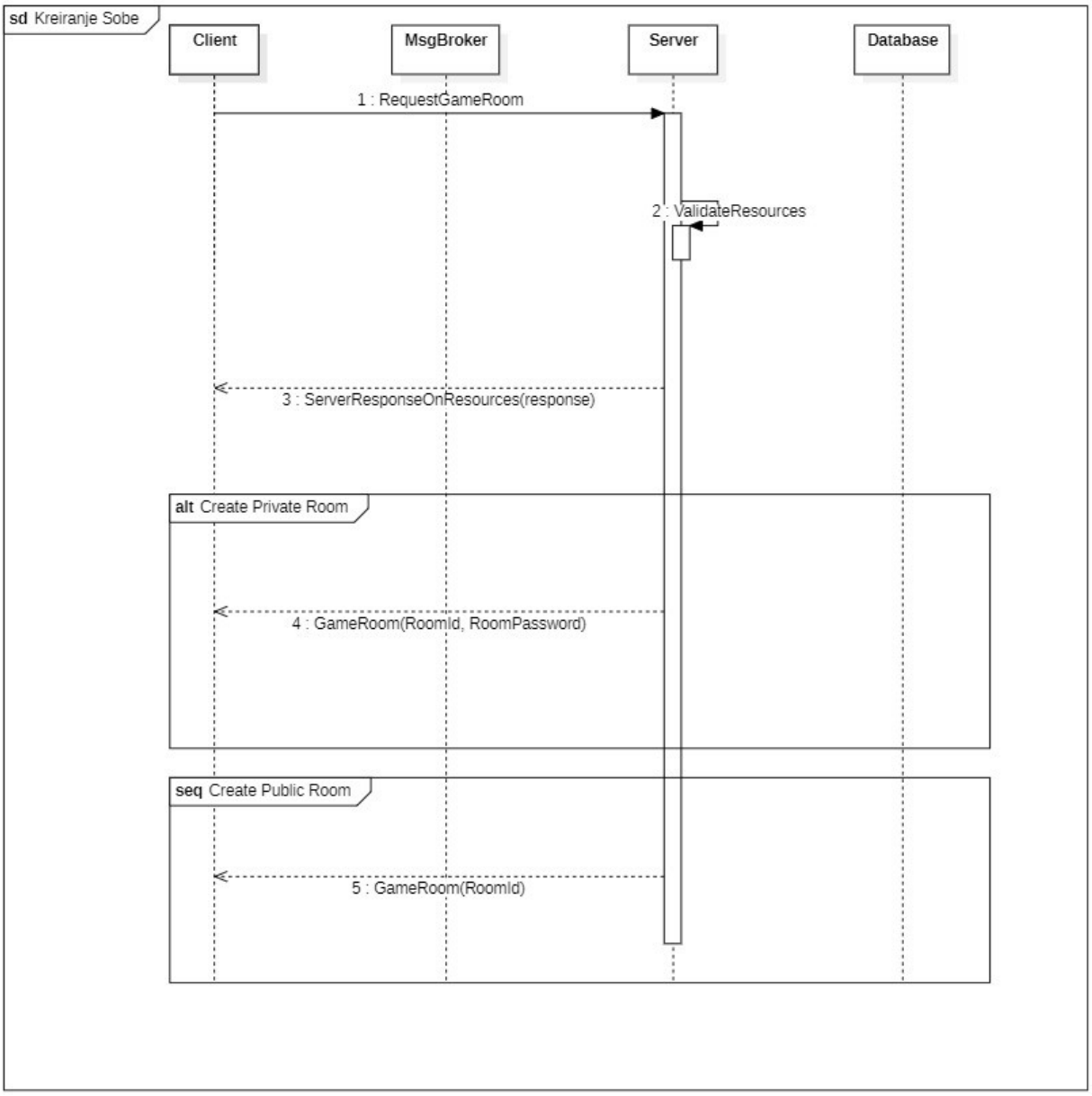
3.4.2 – Пријава већ постојећег корисника

Исто као и код регистрације, сервер проверава податке које му је послао сам клијент и на основу тога креира токен за приступ, на основу кога корисник може бити и аутоматски улогован у апликацију. Сервер такође обавештава корисника о томе да ли је пријава успела или не.



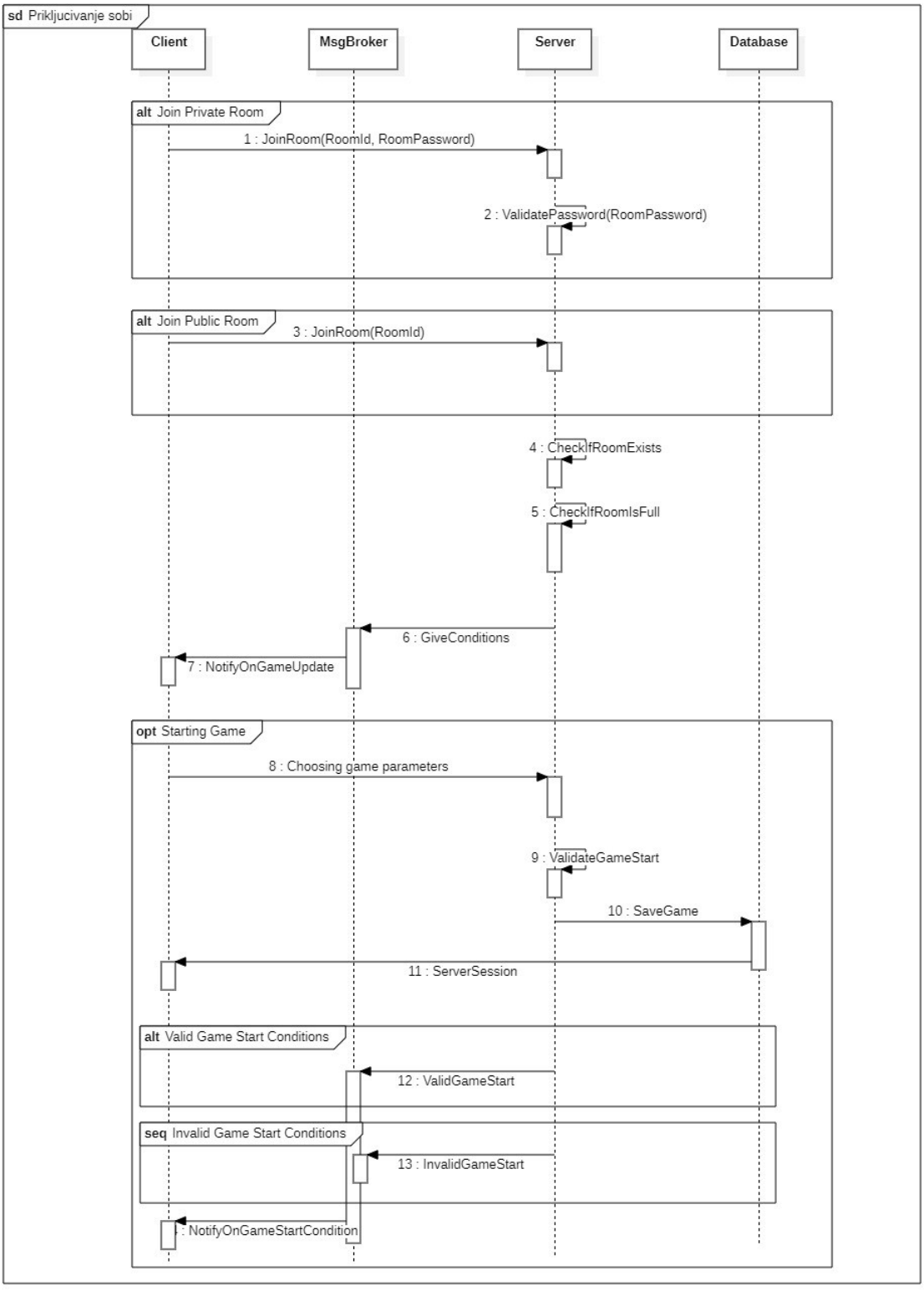
3.4.3 – Креирање собе

Креирање собе се врши тако што корисник шаље захтев за креирање собе, и након провере и обавештењу корисника да су адекватни ресурси система њему расположиви, сервер враћа нову собу, заједно са лозинком уколико је реч о приватној соби.



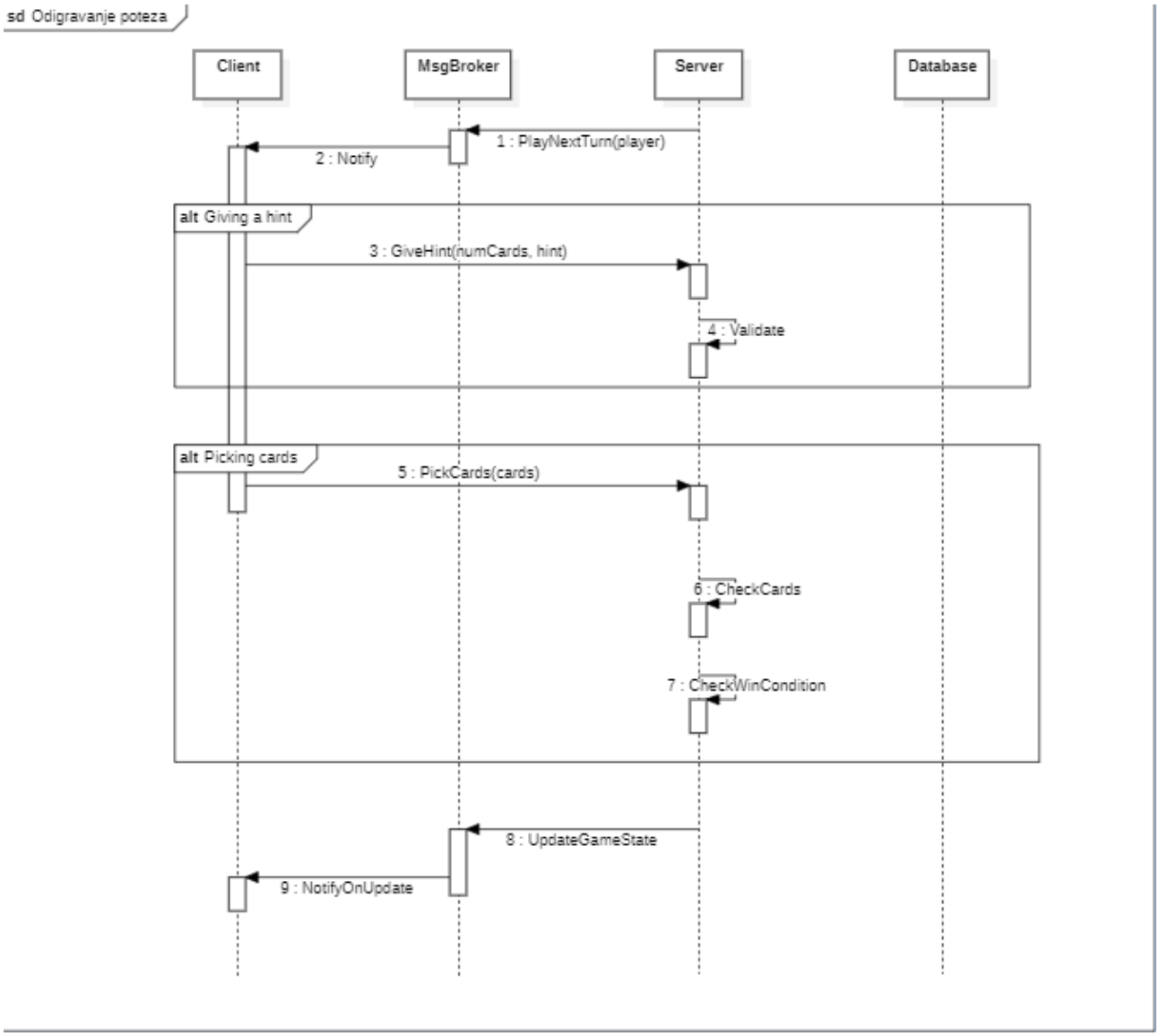
3.4.3 – Прикључивање соби

Корисник бира собу којој жели да се прикључи, на основу доступне листе која се налази на почетној страници и уноси лозинку за исту уколико је то потребно (ако је соба приватна). На основу провера које сервер извршава приликом поступка прикључивања, корисник који жели да уђе, као и сви корисници који се већ налазе у соби, бивају обавештени о изменама које су се десиле. Такође, корисник у том случају може да изабере и да започне игру, и о томе бивају обавештени након додатних провера које сервер извршава и при том чува игру у бази података.



3.4.5 – Одигравање потеза

На основу улоге, корисник унутар већ започете партије, може свом саиграчи проследити траг, или изабрати које карте жели да открије. Након тога сервер проверава стање игре након тога и обавештава све играче о томе које карте су откривене и шта то значи за тренутну партију.



3.5 – Имплементациона питања

Клијентска страна

Клијентски део Mindlink апликације реализован је као једностранична веб апликација, са фокусом на брзу интеракцију и јасан приказ стања игре. За имплементацију корисничког интерфејса коришћен је React у комбинацији са програмским језиком TypeScript, што омогућава развој компонентно оријентисаног интерфејса и већу поузданост кода.

Кориснички интерфејс је подељен на више логичких целина које одговарају различитим фазама игре, као што су почетни екран, простор за окупљање играча и сам приказ активне партије. Свака од ових целина приказује само информације релевантне за тренутни контекст корисника.

Серверска страна

Серверска апликација представља централни део система и задужена је за обраду захтева клијената, спровођење правила игре и координацију комуникације између учесника. Имплементација је извршена коришћењем ASP.NET окружења и програмског језика C#, што омогућава развој стабилних и проширивих веб сервиса.

Комуникација са клијентима реализована је путем REST API интерфејса, кроз који се обрађују захтеви као што су креирање партије, прикључивање игри и извршавање потеза.

За рад са базом података коришћен је Entity Framework, који омогућава мапирање доменских модела на релациону структуру базе и поједностављује управљање перзистентним подацима, без директног писања SQL упита.

У циљу подршке комуникацији у реалном времену, систем користи RabbitMQ као посредника за размену порука. Сервер објављује догађаје који представљају промене у току игре, као што су прикључивање играча, откривање картица, задавање асоцијација и завршетак партије. Оваква архитектура омогућава да се комуникација изолује од основне логике апликације и лако прошири по потреби.

Догађаји који се обрађују преко RabbitMQ система прослеђују се клијентима коришћењем WebSocket веза. Посебан процес на серверској страни слуша поруке из редова и шаље одговарајућа обавештења повезаним клијентима, чиме се обезбеђује тренутно ажурирање стања игре.

Перзистенција

За трајно складиштење података коришћена је PostgreSQL релациона база података. У бази се чувају подаци о корисничким налозима, собама за игру, активним и завршеним партијама, учесницима, историји потеза, резултатима, као и системским подешавањима. Оваква организација омогућава поуздано чување података и њихово накнадно коришћење у оквиру апликације.

4. Анализа архитектуре

4.1 – Потенцијални ризици у имплементацији и стратегије превазилажења

Сам рад веб апликације који се заснива на комуникацију са сервером и остваривањем могућности да се више партија одиграва истовремено, са собом повлачи просто питање о капацитетима сервера као и начину да скалабилност пројекта буде довољно добро обрађена. Притом, сервер комуницира са централизованом базом података што додатно ствара оптерећење на његов рад, па би основна ствар била то да се обрати велика пажња на могућност дељења складишта на више засебних делова као и то да се серверски капацитети прошире и да то не утиче на њихов целокупан рад.