VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Kinder2 „CityPuzzle“ Sistemos architektūra

Autoriai:
Justas Baniulis 2 kursas, 1 grupė
Konstantinas Bernotas 2 kursas, 1 grupė
Olegas Borisovas 2 kursas, 1 grupė
Rokas Petrauskas 2 kursas, 1 grupė

Vilnius, 2022

Turinys

Sąvokos

Vartotojas – asmuo, įsidiegęs programėlę i telefoną, galintis žaisti tiek atviruose, tiek privačiuose žaidimuose. Pelną generuoja naudodamas uždaro kambario (aprašyta toliau) administratoriaus funkcijas.
Rėmėjas – asmuo, naudojantis internetinę sistemą su tikslu pridėti savo įstaigą į pagrindinę (išmaniuosiuose įrenginiuose prieinamą „City Puzzle“) sistemą, už šią paslaugą mokantis pinigus.
Partneris – turizmo centro, regioninio parko, valstybinio parko ar kitos įstaigos darbuotojas, kuris naudojasi internetine sistema su tikslu populiarinti jo atstovaujamą vietovę ir padedantis sistemos administratoriui rinkti informaciją programėlės veikimui gerinti. Šis asmuo pelno negeneruoja.
Šablonas – pradinis, dar nepublikuojamas ir neaktyvus uždaro kambario (aprašyta toliau) egzempliorius.
Kontekstas

Pirminė programos būsena:

„CityPuzzle“ – žaidimas, skatinantis keliauti, atrasti bei tyrinėti įvairius geografinius, istorinius ir kitus kultūrinę reikšmę turinčius objektus. Artėjant ar tolstant nuo tokio objekto, matomas atitinkamas indikatorius, o atstumui mažėjant – po truputį atsiveria ir minėtojo objekto nuotrauka. Jei rasti finišą sunku – galima naudoti pagalbos mygtuką, laikinai parodantį visą objekto vaizdą.

Planuojamas pokytis

Planuojame sukurti kambarius, kuriuose būtų galima įtraukti kitus vartotojus. Kambariai turėtų savo vidines užduotis, taškų sistemas, o vidinė informacija nebūtų prieinama viešai. Sukurti privatų kambarį kainuoja priklausomai nuo dalyvių skaičiaus. Kambario administratorius atsakingas už užduotis, apdovanojimus, užuominas, ir t.t.

Motyvacija pokyčiams

Privatūs žaidimų kambariai – antrasis pelno šaltinis sistemoje. Priešingai nei pirmasis, pelnas generuojamas ne rėmėjų, o vartotojų. Be to, privatūs kambariai gali būti kaip viena svarbiausių priežasčių, kodėl vartotojai naudoja sistemą, nes tai leidžia varžytis su draugais ar kolegomis unikaliu būdu, kadangi jo nesiūlo nei vienas potencialus konkurentas. Šis pakeitimas gali pritraukti daug naujų žaidėjų, kurių anksčiau nedomino vieši žaidimai, bet gali juos įtraukti, dėl ko gali netiesiogiai padėti pirmajam pelno šaltiniui.

Reikalavimai

Funkciniai:

1. “Private Rooms” lango sukūrimas
   1.1. Galimybė sukurti privatų kambarį
   1.2. Galimybė pamatyti sukurtų kambarių sąrašą su atvirų vietų skaičiumi ir vartotojo leidimu
   1.3. Galimybė prisijungti prie kambario
   1.4. Galimybė prisijungti prie kambario nuskenavus QR kodą
2. „My Rooms“ lango sukūrimas
   2.1. Galimybė pamatyti savo sukurtų šablonų sąrašą
3. „Room Designer“ lango sukūrimas
   3.1. Galimybė pakeisti kambario pavadinimą
   3.2. Galimybė pakeisti kambario slaptažodį
   3.3. Galimybė pakeisti maksimalų kambario žaidėjų skaičių
   3.4. Galimybė pratęsti kambario galiojimo periodą
   3.5. Galimybė aktyvuoti kambarį
4. „Payment Window“ lango sukūrimas
   4.1. Galimybė sumokėti už paslaugas
   4.2. Kortelės duomenų verifikacija
5. „Add/Remove Tasks Window“ lango sukūrimas
   5.1. Galimybė peržiūrėti konkretaus kambario šabloną
   5.2. Galimybė pridėti šablono užduotį
   5.3. Galimybė redaguoti šablono užduotį
   5.4. Galimybė pašalinti šablono užduotį
   5.5. Galimybė pakeisti šablono užduočių eiliškumą
6. „View Room Window“ lango sukūrimas
7. „Players Window“ lango sukūrimas
   7.1. Galimybė peržiūrėti kambario lyderių lentelę
   7.2. Galimybė pašalinti vartotoją iš kambario
   7.3. Galimybė pakviesti vartotojus su kodu
   7.4. Galimybė pakviesti vartotojus su QR kodu
8. „User Profile Window“ lango sukūrimas
   8.1. Galimybė peržiūrėti konkretaus vartotojo progresą
9. „Ban List Window“ lango sukūrimas
   9.1. Galimybė sugrąžinti pašalintą vartotoją
10. „Task Creation Window“ lango sukūrimas
    10.1. Galimybė parinkti užduoties vietą naudojant interaktyvų žemėlapį
    10.2. Galimybė pridėti užduoties nuotrauką iš įrenginio duomenų talpyklos

Sistemos architektūra

Sistemos architektūrą parėmėme “Service-oriented” stiliumi. Priežastys kodėl tai pasirinkome:

1. Sistema naudoja kitų įmonių servisus, kaip “Google maps” API.
2. Pirmosios iteracijos metu taip pat naudojome ir tobulinome savo sukurtą API (CityPuzzleAPI) ir sistema apskritai buvo paremta SOA, todėl per brangu keisti architektūrą.
3. Keičiantis ar plečiantis verslo dalykinei sričiai, galima lengvai praplėsti, kurti naujus, ar naudoti kitų įmonių servisus, atitinkančius naujus poreikius. Pavyzdžiui: pakvietimui ar prisijungimui į naują kambarį implementuoti “Google play” paskyrų servisą.

Architektūros šablonai (“patterns”)

Pagrindinis sistemos architektūrinis šablonas yra “svogūno” architektūra. Šis architektūrinis šablonas atskiria loginį (API ir kita verslo logika), duomenų (duomenų bazė) ir prezentacijos (“front-end”) sluoksnius. Šis šablonas koreliuoja su MVC šablonu. Priežastys kodėl tai pasirinkome:

1. Jautrūs vartotojų duomenys prieinami tik per prezentacijos ir loginį sluoksnį. Šie sluoksniai kontroliuoja, kokius duomenis gauna vartotojas (Pavyzdžiui: gauna tik kitų vartotojų slapyvardžius ir turimus taškus lyderių lentelėje, bet ne kur jie lankosi).
2. Šis architektūrinis šablonas leidžia lengvai sluoksnius plėsti ir suteikti jiems naujas užduotis, kurios gali atsirasti verslo dalykinėje srityje. Pokyčiai verslo dalykinėje srityje neturėtų išeiti iš sluoksnių ribų, nes verslas paremtas lokacijos objektais.

Komunikacijai tarp sluoksnių naudojame CRUD šabloną. Sistema atlieka operacijas, kaip “create”, “delete”, “update”, “read”. Pasirinkome tokį šabloną dėl paprastumo, populiarumo ir naudojimo pirmoje iteracijoje (pokyčiai per brangūs).

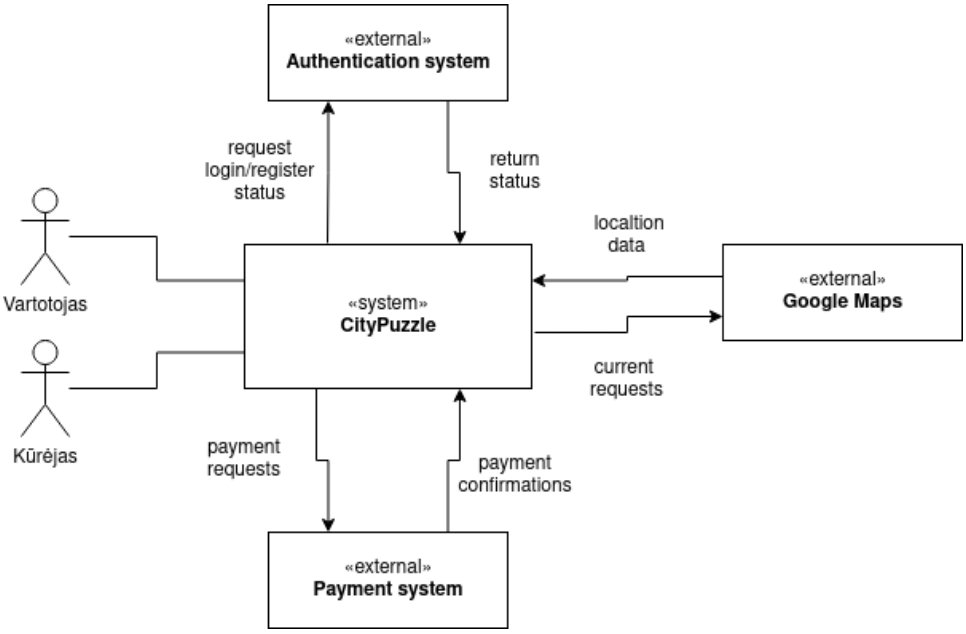
Pasirinkome “Data-centered” arch. šabloną, nes jis lengvai palaikomas ir yra paprastas (Pavyzdžiui: tik viena esanti duombazė neapkrauna darbuotojų darbu). Duombazėje susiformuojantys butelio kakleliai neturėtų būti reikšmingi vartotojui.

Pjūviai

Sistemos paleidimui apibūdinti pasirinkome "Context”, “Deployment”, “Operational”, “Development”, “Functional” ir “Information” pjūvius. “**Context**” pjūvis naudojamas suprasti sistemos apibendrintam veikimui, santykiui su verslo modeliu ir priklausomybes nuo kitų sistemų. “**Functional**” pjūvis padeda suprasti architektūrinius elementus, kokias jie atlieka sistemos funkcijas. “**Information**” pjūvis apibūdina, kaip sistema manipuliuoja informacija ir kokie jos srautai sistemoje egzistuoja. “**Development**” pjūvį naudojame, nes jis apibūdina sistemos veikimą programuotojam, testuotojam ir kitiems su kodu susijusiais “stakeholders”. “**Deployment**” pjūvis svarbus, nes jis apibūdina kokioje aplinkoje sistema veiks ir kokias

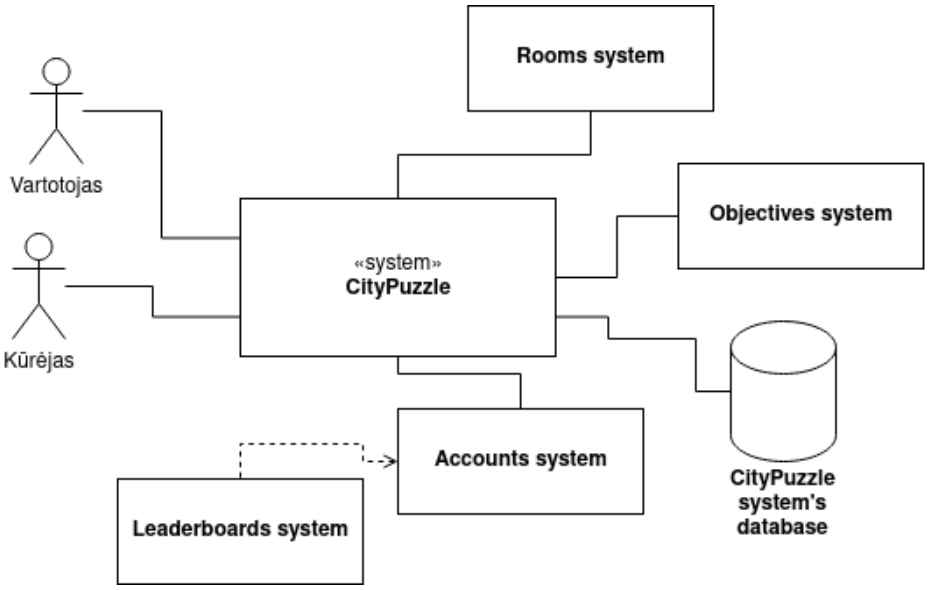
priklausomybes ji turi. “**Operational**” pjūvis reikalingas, nes jis apibūdina sistemos administraciją, operacijas ir palaikymą veikiant produkcinėje aplinkoje.

"Context” pjūvis



1 diagrama. Sistemos išorinės priklausomybės

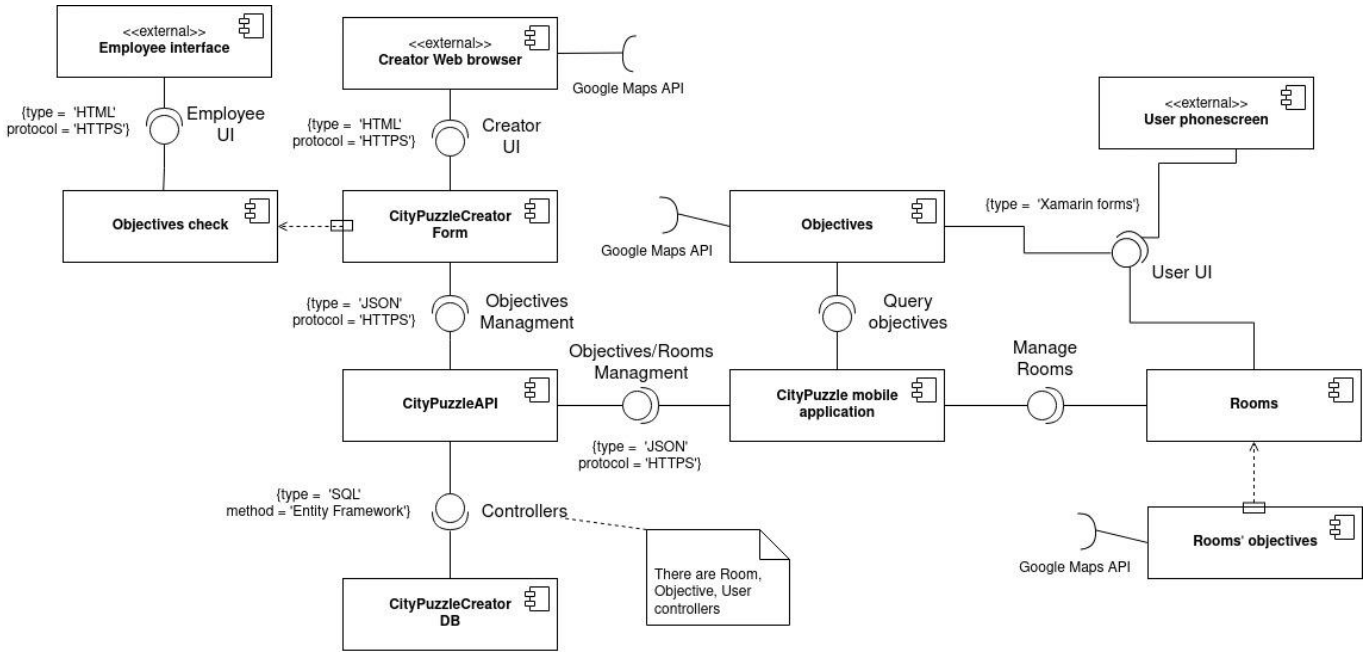
Sistemos pagrindinės išorinės priklausomybės susideda iš autentifikacijos, “Google Maps” ir apmokėjimų sistemų. Autentifikacija svarbi paskyrų kūrimui ir vėlesniam jų vartojimui, apmokėjimai svarbūs verslo modelio vykdymui/atsiperkamumui ir “Google Maps” reikalingas, nes įgyvendina pagrindinę verslo idėją - lokacija paremtą žaidimą.



2 diagrama. Sistemos vidaus sandara

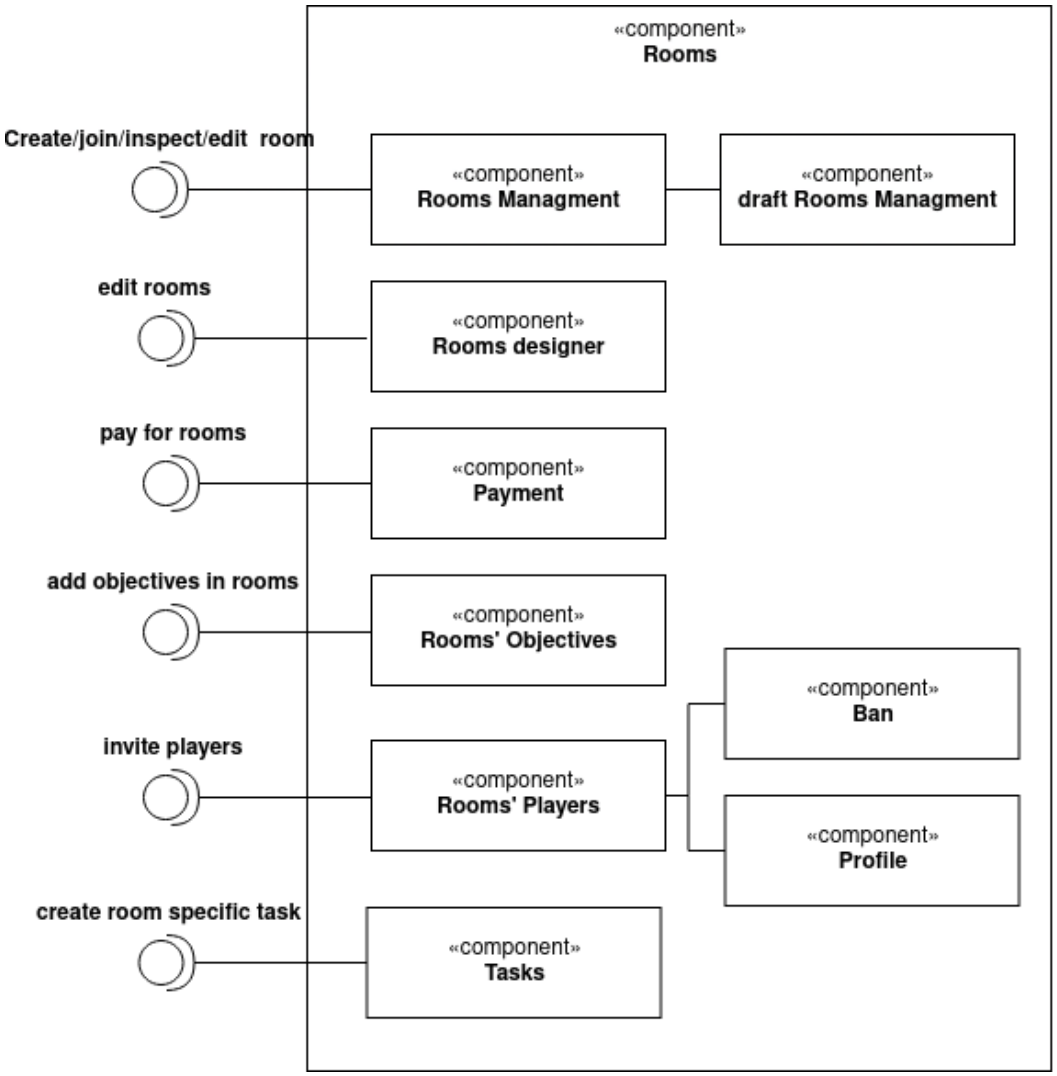
Sistemos vidus susideda iš kambarių, užduočių, paskyrų sistemų. Kambariai talpina Vartotojus ir leidžia jiems žaisti tik tame kambaryje. Užduočių sistema tvarko užduotis, kurios gali būti viešos, juodraštinės ir priklausyti kambariams. Paskyros atlieka vartotojų ir kūrėjų identifikaciją sistemos viduje ir atitinkamus leidimus sistemoje. Nuo paskyrų priklauso lyderių lentelė, joje kaupiami taškai (“žaidybinizacija”). Duombazė sistemoje laikoma tik viena, pagal “data-first” principą.

“Functional” pjūvis



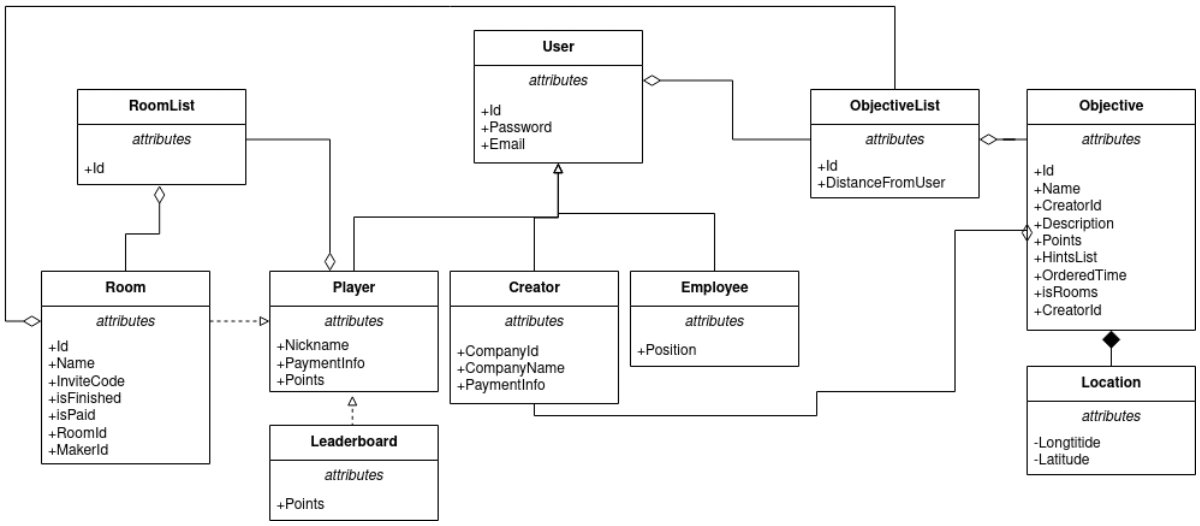
3 diagrama. Sistemos funkcinė komponentų diagrama

Prezentacijos funkcijas atliekantys elementai yra išoriniai telefono ekranas, internetinis puslapis ir darbuotojo interfeisas. Juos visus su kitais komponentais jungia jų atitinkamas UI, tik telefono aplinka naudoja ne HTML, o xamarin formas. Šių komponentų darbas yra reprezentuoti esamą būseną. Pačios aplikacijos esmė yra perduoti užduotis ir suteikti kambarių tvarkymo funkcijas, o pastarieji komponentai jas įgyvendina (su paliepimu ateinančio iš ”front-end”). Pažymėtina, kad egzistuoja atskiras užduočių komponentas kambariams, kurio funkcionalumas priklauso nuo kambario, o ne nuo viešų užduočių komponento. Visi užduočių komponentai priklauso nuo Google Maps API. CityPuzzleCreator funkcija yra kurti užduotis (su patvirtinimu, kad ji saugi), mobilios aplikacijos komponento funkcija yra pateikti galimas užduotis ir suteikti galimybę sukurti privatų kambarį. CityPuzzleAPI ir CityPuzzleCreator su mobilia aplikacija jungia REST (https protokolas). Visi duomenys saugomi duombazėje, prie jos prieina API su kontroleriais, kuriuos turi kiekviena esybė (vartotojas, kūrėjas, kambarys, užduotis ir kitos).



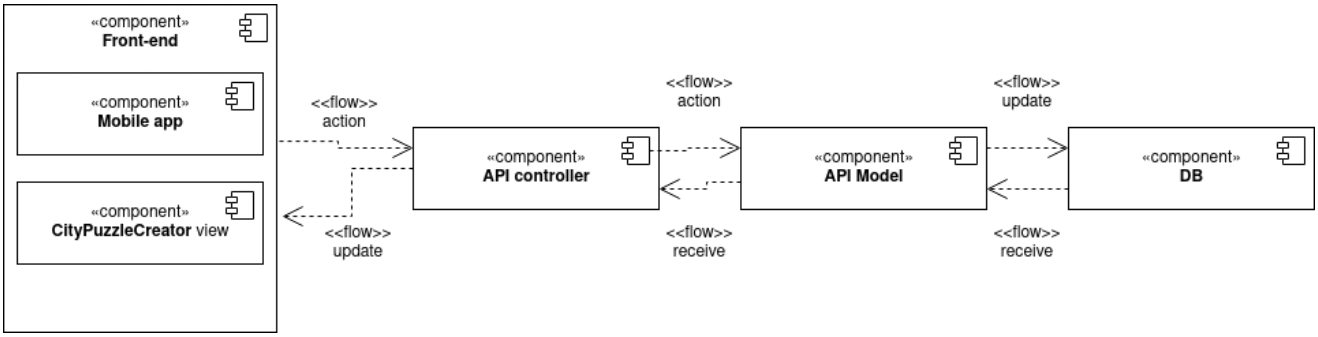
4 diagrama. Kambarių funkcinė diagrama

“Information” pjūvis



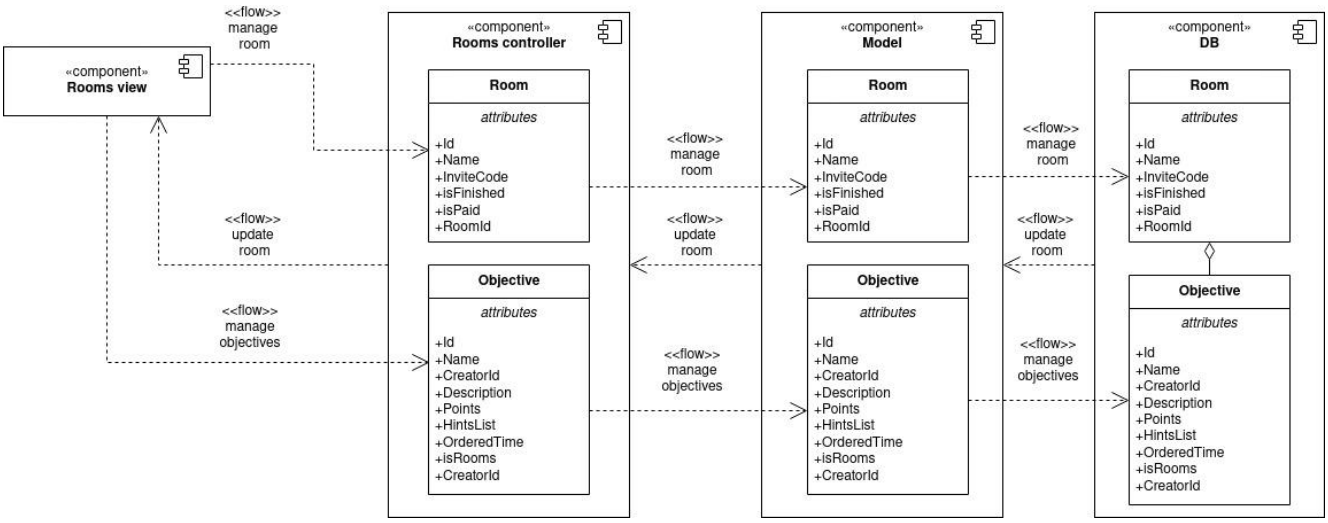
5 diagrama. Statinės informacijos diagrama

Planuojamas pakeitimas reikalauja šiek tiek atnaujinti Duomenų bazės esybes. Kambariai yra apmokestinami ir pridedama galimybė juos padaryti “juodraštiniais” su “IsFinished” būsena. Taip pat planuojame parengti sąrašus užduočių ir kambarių, su kuriais vartotojas gali dirbti. Darbuotojas, būdamas atsakingas už sistemos tvarką (pavyzdžiui sekti užduočių etikos standartų) gali prieiti prie visų sąrašų.



6 diagrama. Informacijos tėkmės sistemoje diagrama

Veiksmas, kurį vartotojas atlieka yra nusiunčiamas API kontroleriui. Ten įvykdoma atitinkama logiką, kuri atnaujinama modelio pagalba. Modelis atnaujina duombazę ir siunčia atgal atnaujintą informaciją kontroleriui. Ji atnaujinama ”front-end” komponentuose iš kontrolerių.

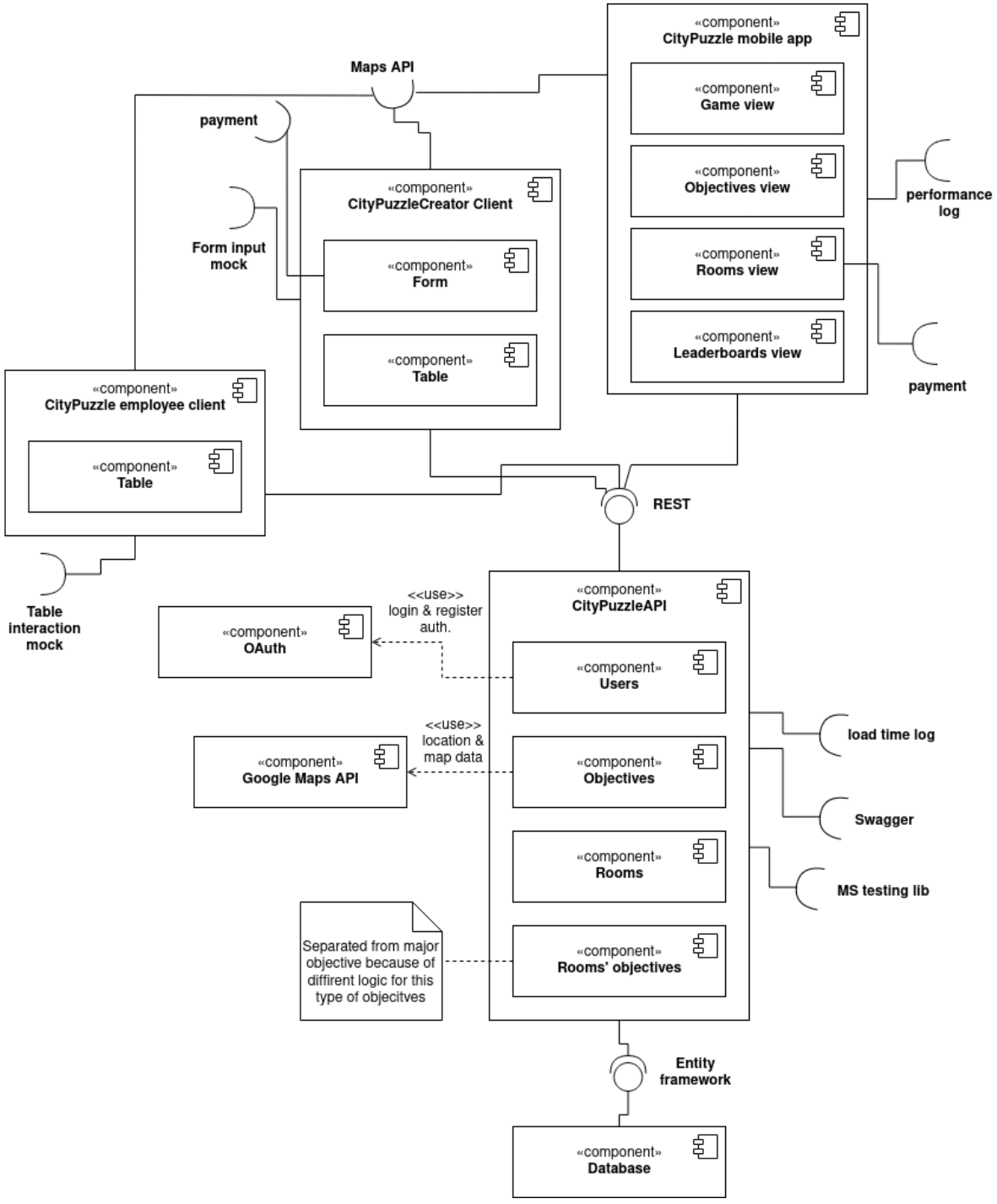


7 diagrama. Kambarių sistemos informacijos tėkmės diagrama

Vartotojui paspaudus mygtuką (pridėti, pašalinti, pakeisti) užduotims, sistema perduoda tą informaciją kontroleriui. Kontroleris, žinodamas savybes susieja lango įeigą su modeliu ir jam siunčia. Modelis perduoda pakeitimą duombazei ir joje yra surišti kambariai su užduotimis. Įvykus pakeitimui viskas siunčiama atgal.

Tuo pačiu principu veikia ir užduočių manipuliavimas (pridėti, pašalinti, pakeisti) - kontroleris perima informaciją iš CityPuzzleCreator kliento paspaudus mygtuką. Kontroleris siunčia modeliui, modelis duombazei. Atnaujinta informacija grąžinama atgal (dėl panašumo su praeita tėkme diagrama nebraižyta).

“Development” pjūvis

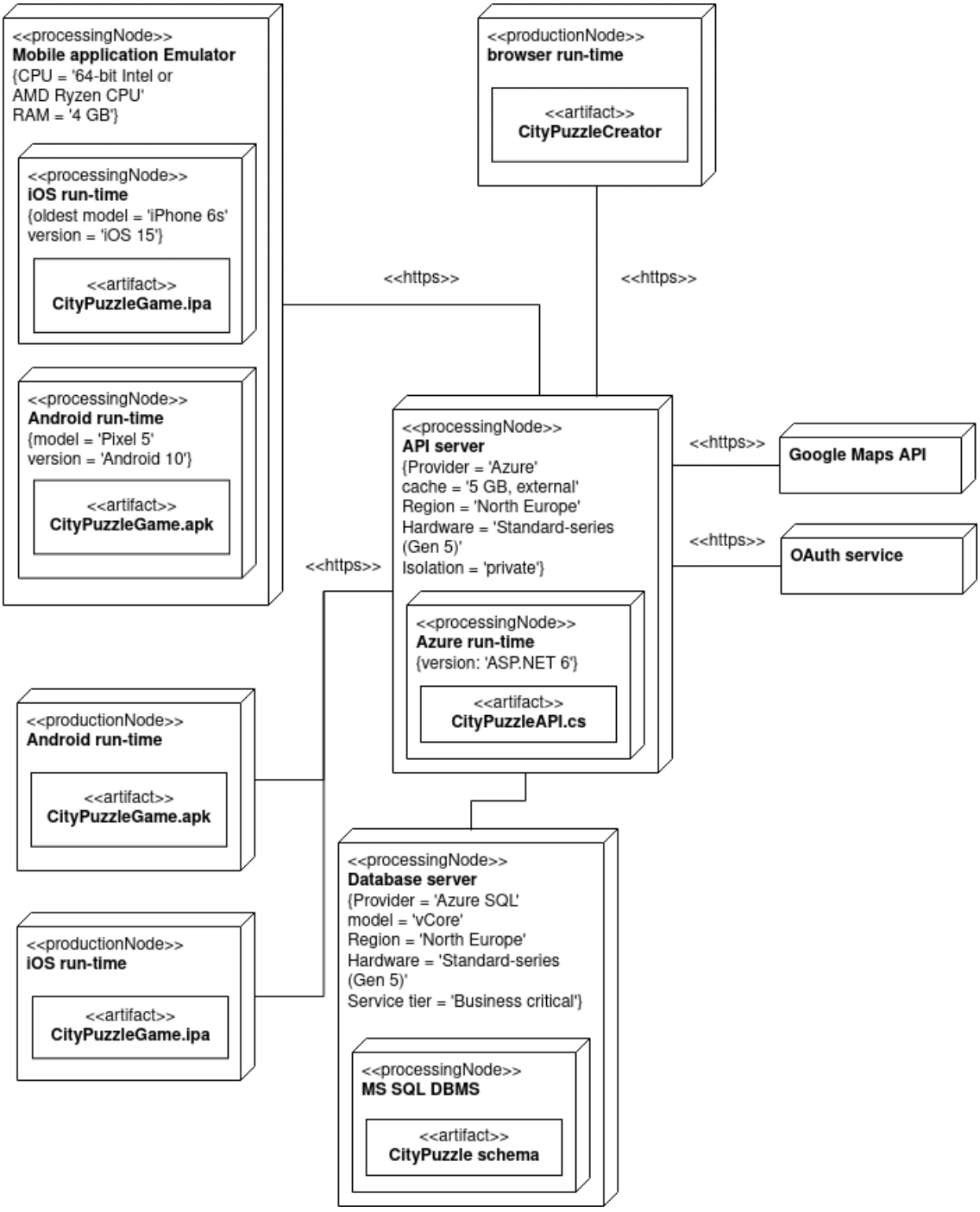


8 diagrama. Sistemos komponentai

CityPuzzleAPI dirba su esybėmis, gauta iš duombazės, ir su jomis susijusia logika. “Development” pjūvyje atskyrėme kambarių užduotis nuo bendrų užduočių, nes jų verslo procesai skiriasi (kambarių užduotys valdomos vartotojų, o bendrosios kūrėjų, tad galimybės

skiriasi, pavyzdžiui kūrėjų užduotys laikas galioja, kol sumokėtas laikotarpis eina, o kambario užduotys gali būti kuriamos dažniau, nemokamai (bet pats kambarys mokamas) ir turinys neprižiūrimas darbuotojų). Užduotims įgyvendinti reikia “Google Maps” API, prisijungimui ir registracijai OAuth. Sekti duomenų gavimo, atidavimo, manipuliacijos rodmenims implementuojamas statistikos sekimas (“logging”), kad darbuotojai matytų, kada kyla laiko trukmė apkraunant serverį. “Swagger” ir testavimo bibliotekos implementuojamos testavimo darbams. API perima duomenis iš duombazės “Entity framework” pagalba. Pasirinkta ši biblioteka, nes ji susieta su “.NET” aplinka, yra aktyviai palaikoma ir susieja .NET objektus su duombazės esybėmis programavimo paprastumui. Prezentacijos sluoksnyje, kiekvienas komponentas turi savo “mock”, nes jie visi dirba su formomis, išeitimis, įeitimis - testuoti reikia UI elementus. Darbuoto klientui reikia tik lentelės, nes darbuotas tik tikrina objektus. Kūrėjas gali kurti objektus su forma, manipuliuoti jais lentelėje ir už juos sumokėti (tai vyksta per mokėjimo sistemos langą (“view”)). Mobilos aplikacijos turi žaidimo langą, kuris leidžia vartotojui vykdyti objektus, objektų katalogo langą, leidžia vartotojui pasirinkti objektą ir kambarių langą, kuriame vartotojas kuria kambarius ir jų užduotis arba ima užduotis iš viešai prieinamų. Nusprendėme prie mobilios aplikacijos prijunkti ir “performance” matuoklį, kadangi mobilių įrenginių “hardware” stipriai skiriasi (pvz.: įvairūs Android įrenginiai, Apple įrenginiai). Krovimo laiko matuoklis leidžia priimti sprendimus optimizuojant (pvz.: Apple išleidus naują iPhone seriją, gamintojas sulėtina senų telefonų procesorių dažnį ir mūsų įmonės darbuotojai gauna signalą, jog reikia “mažinti apsukas” programėlės sistemoje, kai vartotojams ilgiau užtrunka krauti žemėlapį ar stringa UX)

“Deployment” pjūvis



9 diagrama. Priklausomybės

API ir duombazės serverio aplinką pasirinkome “Azure”, nes ji, kaip ir .NET aplinka, palaikoma “Microsoft”. Duombazės serverio pasirinktas planas automatiškai kuria atsargines kopijas, kas leidžia sumažinti žalą įvykus nelaimei sistemoje, naudojami naujausi procesoriai ir regionas arčiausiai verslo vykdymo vietos (Lietuvos). API serveris taip pat turi naujausios serijos CPU ir

yra izoliuotas naudojimui tik įmonės servisam, regiono pasirinkimo priežastis ta pati. DBVS ir API versijos turi būti naujausios, dėl saugumo. API serveris HTTPS protokolu komunikuoja su “Google Maps”, “OAuth” servisais, bei su kitais klientais ir emuliatoriais. CityPuzzleCreator klientas veikia naršyklės aplinkoje su React “framework”. Tiksliai nespecifikavome aplinkos, nes naršyklių varikliai yra dokumentuoti ir kiekvienas turi savo specifikacijas į kuriuos . Viena iš “deployment” aplinkų yra emuliatoriai, kuriuose simuliuojamas programėlės veikimas skirtinguose įrenginiuose. Emuliatoriai yra svarbūs programuotojams, testuotojams, bei kitiems “stakeholders” nes pirkti įrenginius kiekvienam naujam modeliui išeinant yra per brangu. Pagal “Xamarin” emuliatoriaus tech. specifikaciją parinktas ir CPU, RAM. Seniausias įrenginys kurį emuliuojame yra “iPhone 6s” ir “Pixel 5”, nes jie yra seniausi įrenginiai dar palaikomi gamintojų. Įrenginiai gali keistis ateityje, kai gamintojai nutraukia palaikymą. Produkcijoje taip pat aplikacija paleidžiama iOS ir Android aplinkose.

“Operational” pjūvis

Klientų aptarnavimas – Norint išlaikyti gera vartotojų ir klientų patirtį, reikia atsakyti į jų turimus klausimus. Jei vartotojas ar klientas neranda atsakymo į savo klausima dažniausiai užduodamų klausimų sąraše, jie gali tiesiogiai susisiekti su klientų aptarnavimu naudodamiesi ticket sistema. Ši sistema prašys pateikti kokio tipo klausima nori užduoti, su tai susijusia informacija ir teksto laukelyje bus galima aprašyti problemą. Po pateikimo, vartotojas gaus elektroninį laišką patvirtinantį, kad jų klausimas yra užregistruotas ir klientų aptarnavimas su jais susisieks su atsakymu.

Duomenų bazės atsarginė kopija – Sistemos duomenų bazė yra laikoma Azure, tai atsarginės kopijos bus kuriamos automatiškai.

Sistemos atnaujinimai – Atnaujinimai turi užtikrinti, kad visos esamos funkcijos veikia teisingai ir naudojamų plug-in versijos yra pačios naujausios. Tai išpildyti padės sistemai paruošti unit ir integration testai. Atnaujinimai bus leidžiami naktį, kai vartotojų kiekis yra mažiausias.

Saugumas

Siekiant apsaugoti vartotojų, klientų bei partnerių duomenis bei privatumą, atsižvelgus į OWASP (Open Web Application Security Project) pateikiamus duomenis, buvo išreikštos pagrindinės sistemos spragos, kurias reikia apsaugoti, bei 2 pagrindinės CityPuzzle saugumo užduotys:

1) Tinkamai apsaugoti vartotojų, partnerių bei klientų jautrią informaciją.
2) Apsaugoti programą nuo galimų puolimų siekiant paveikti programos veikimą ar nutekinti/ištrinti duomenis.

Viewpoint	Galimos grėsmės/sprendimai
Context	• Grėsmė: ♦ Vartotojų, klientų bei partnerių paskyrų ar duomenų vagystė. • Sprendimai: ♦ Sistema naudoja slaptažodžių maišos funkcijas. ♦ Slaptažodžiai yra patikimai užsūdyti (salted). ♦ Duomenų bazė nėra tiesiogiai prieinama. Visa informacijos tėkmė vyksta per API.
Functional	• Pažeidžiami programos komponentai: ♦ Duomenų bazė • Sprendimai: ♦ Apsauga nuo SQL injekcijų: ◊ Užklausos turi būti parametrizuotos bei iš anksto paruoštos. ◊ Vartotojo įvesties tikrinimas.
Information	• Duomenų bazė yra labiausiai pažeidžiama sistemos dalis, todėl sistema turėtų kas 2 savaites, padaryti duomenų bazės kopiją ir ją išsaugoti, tam kad išvengti duomenų praradimo.
Development	• Naudotojas, klientas ar partneris turi turėti prieigą tik prie UI, tam kad išvengti prieigos prie duomenų bazės ir pačios sistemos vidaus.
Deployment	• Testai užtikrina sistemos stabilų darbą ilgame laiko tarpe. • CI/CD apsaugo sistemą nuo nepageidaujamų atnaujinimų.
Operational	• „Microsoft“, kadangi naudojame jų Azure duomenų bazę. • „Google“ bei „Apple“ dėl jų telefonų naudojimo bei seniausių Android ir iOS operacinių sistemų naudojimo ir emuliavimo.

1 lentelė. Saugumo perspektyva. Požiūrių analizė.

Testavimas

Norint užtikrinti sistemos efektyvu ir tikslingą veikimą, naudojame automatizuotus unit ir integration testus. Šitaip programuotojai gali pagauti klaidas, kurios kitaip galėjo būti sumergintos į pagrindinę repozitoriją. Jei kodas nepraeina visų testų, jo neleidžiama merginti.

Testavimo įrankiai – Unit testavimui naudojome xUnit NuGet rinkinį ir priklausomybes imitavome su MoQ framework. Testų metu duomenis laikėme In-Memory duomenų bazėje. Swagger Inspector buvo naudojamas generuoti API dokumentacijai.

Implementuoti testai – Testai buvo implementuojami mūsų ruoštiems pakeitimams, nes ši sistemos dalis yra nauja, todėl jai suteikėme pirmenybę ir pilnam sistemos testui paruošti neužteko laiko ir resursų.

Atsekamumas

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10
Rooms Management	X		X							
Draft Rooms Management		X								
Rooms Designer			X							
Payment				X						
Rooms Objectives					X					
Rooms Players							X			
Ban							X		X	
Profile						X		X		
Tasks										X