PSP Tech

Верзија 1.0

FIND YOUR PETROL

Архитектурен дизајн

PSP Tech:

Никола Станојковски / 181076

Ташко Павлов / 181104

Емил Петровски / 181201

Содржина

1. Архитектурни погледи

1.1 Концептуална архитектура

1.1.1 Клучни концепти

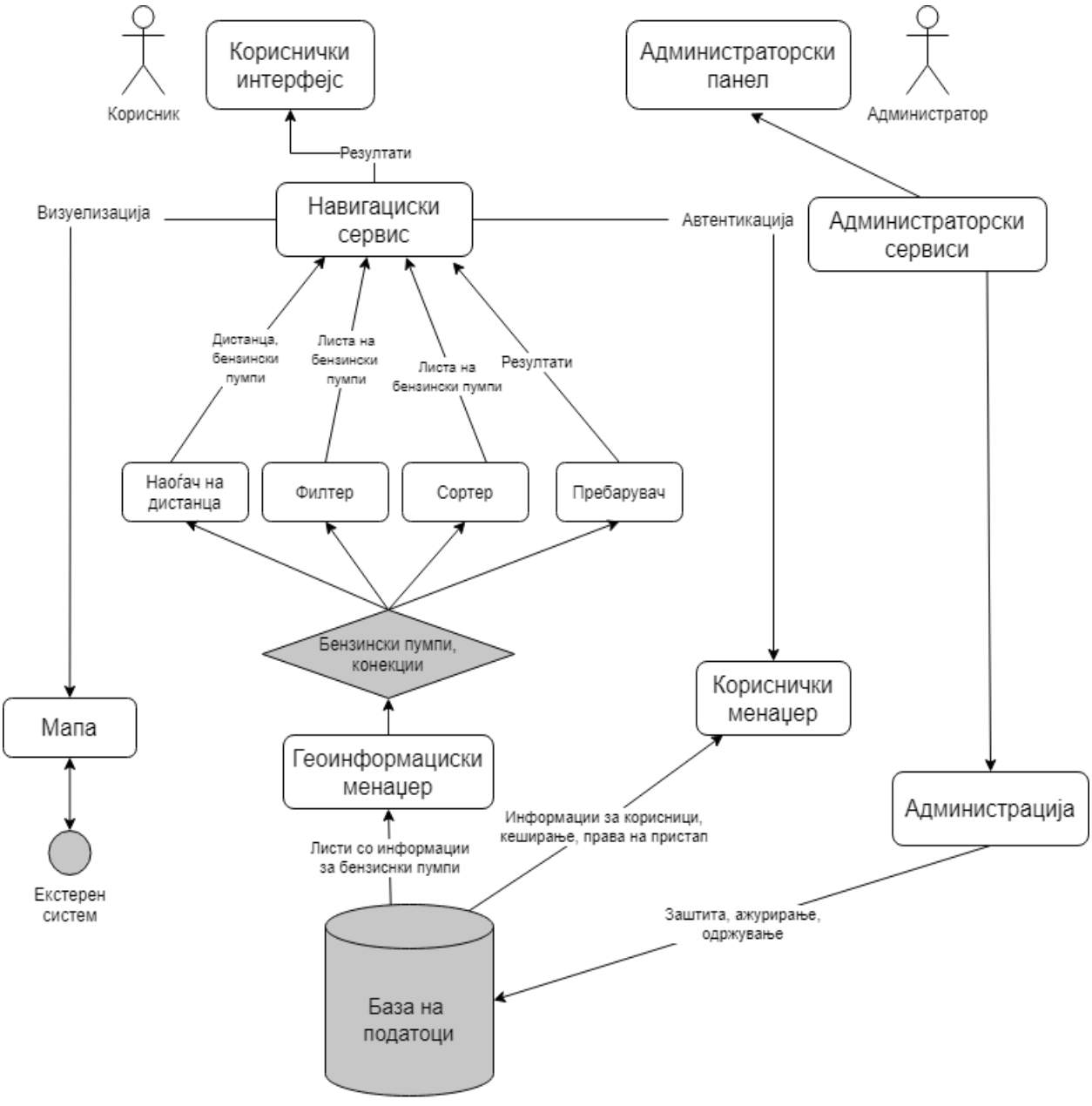
- База на податоци
- Корисник
- Администратор
- Филтрирање
- Приватност
- Бензинска пумпа
- Територија на Република Северна Македонија
- Видови на нафта
- Име
- Опис
- Слика
- Работни часови
- Локација
- Координати
- Контакт
- Водич за користење
- Сервиси
- Кориснички интерфејс
- Ажурирање
- Конекција
- Сортирање
- Одбирање
- Кеширање
- Рестартирање
- Автентикација
- Авторизација
- Најава
- Регистрација
- Достапност
- Доверливост
- Безбедност
- Портабилност
- Информации (за моментален корисник)
- Повратни информации
- Пребарување
- Додавање/Одземање

1.1.2 Категоризација

Шест главни категории за групирање на клучните концепти:
Податоци, Функции, Засегнати страни, Систем, Хардвер, Абстрактен Концепт

Податоци	***Функции***	***Засегнати страни***	***Систем***	***Хардвер***	***Абстрактен концепт***
База на податоци	Филтрирање	Обичен корисник	Сервиси	Глувче	Приватност
Бензинска пумпа	Ажурирање	Администратор	Веб прелистувач	Тастатура	Контакт
Територија на РСМ	Ажурирање		Драјвер	Монитор	Кориснички интерфејс
Видови на нафта	Сортирање				Достапност
Име	Одбирање				Доверливост
Опис	Кеширање				Безбедност
Слика	Рестартирање				Портабилност
Работни часови	Автентикација				Водич за користење
Локација	Авторизација				
Координати	Најава / Регистрација				
Информации (корисник)	Конекција				
Информации (повратни)	Пребарување				
	Додавање / одземање				

1.1.3 Главна шема



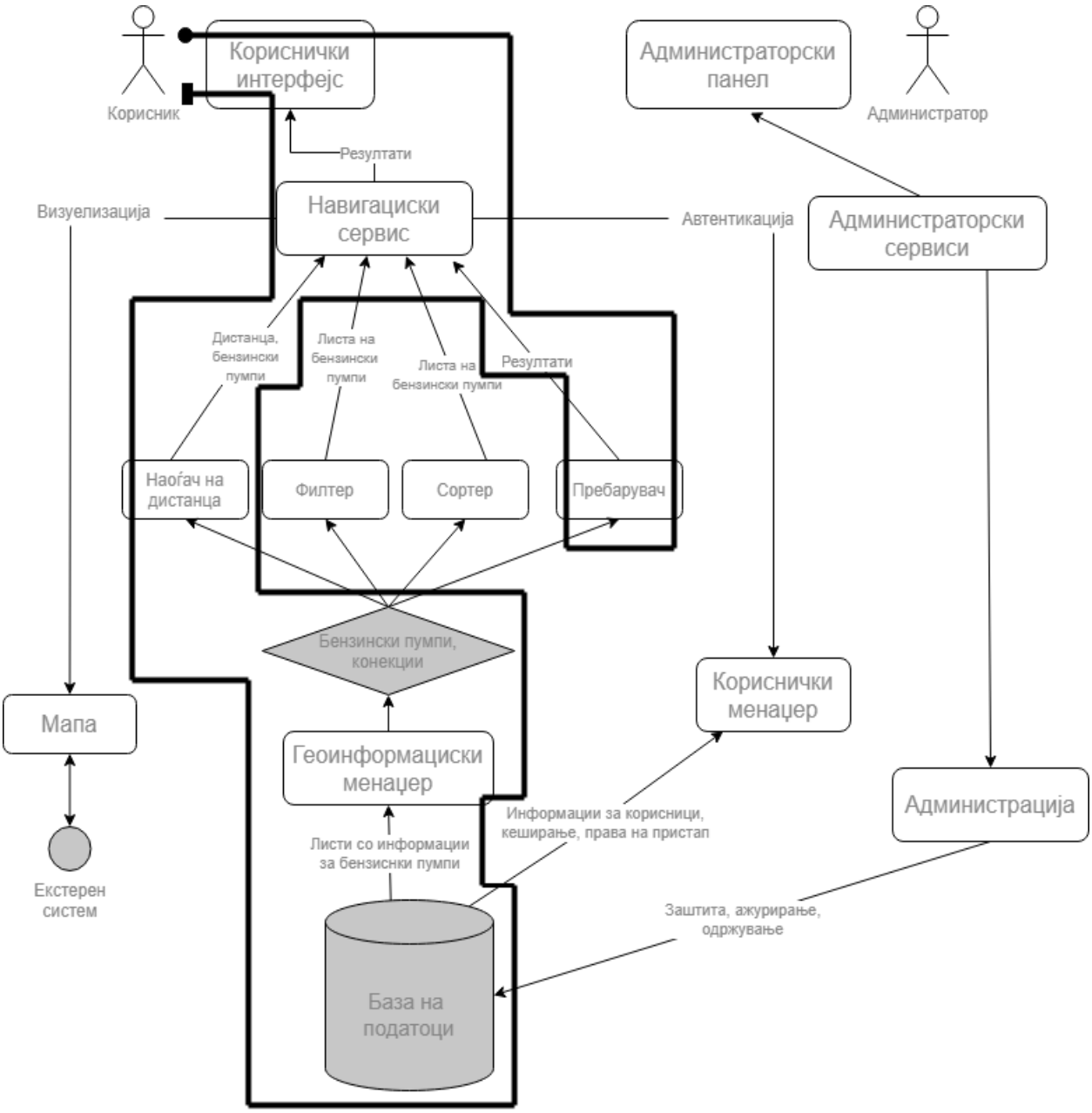
Слика 1: Шема на концептуална архитектура

1.1.4 Компонентни одговорности

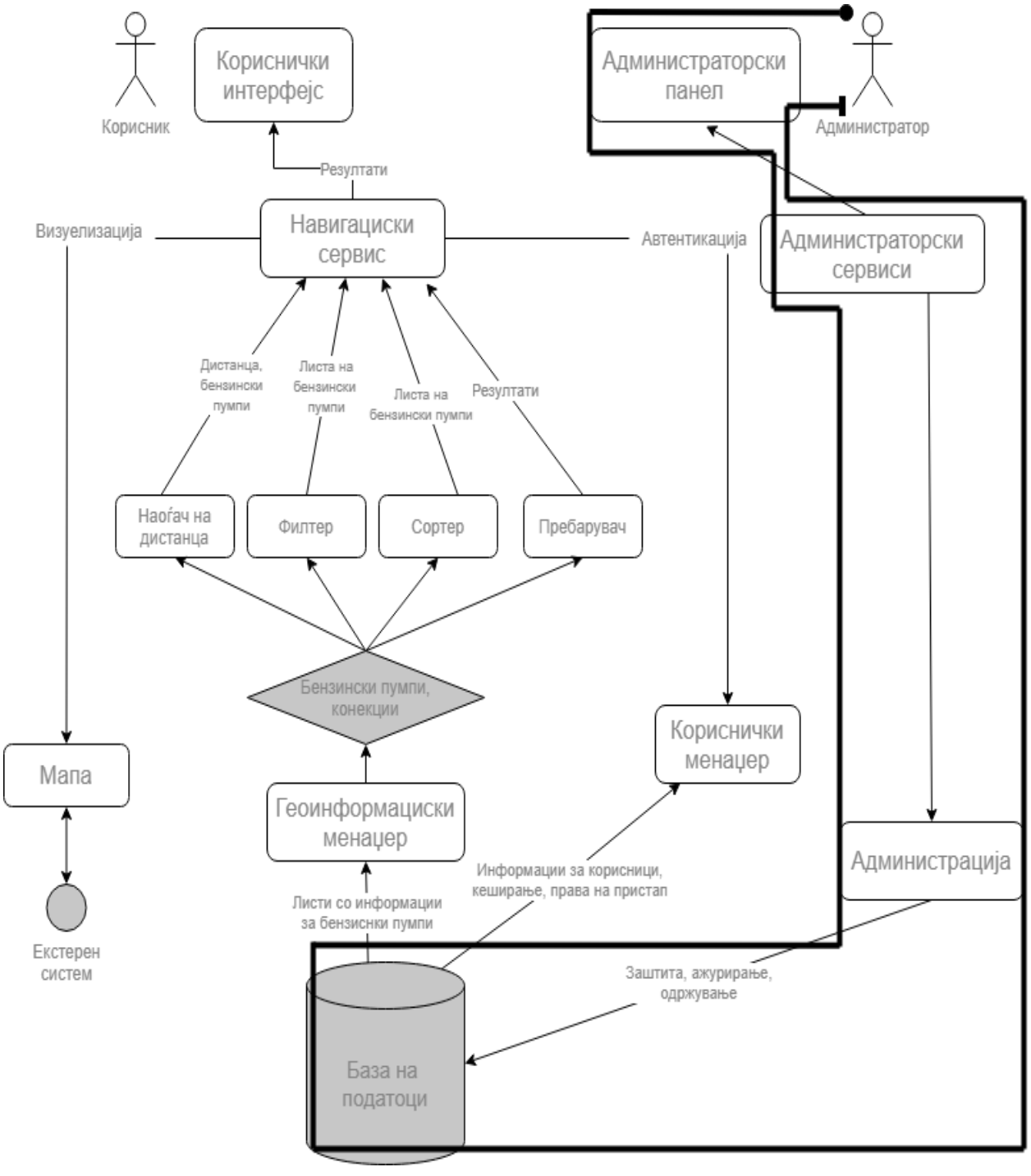
- **AppUI**
  - ShowPetrols
  - DisplayMap
  - ChoosePetrol
  - ShowPetrolInformation
  - LeaveFeedback
- **AdminPanel**
  - ListPetrols
  - ListConnections
  - ListTools
- **NavigationService**
  - Filter
  - AddInfo
  - Sort
  - DistanceComputed
  - ShowDistance
- **AdminService**
  - DatabaseAdministrationTools
  - OperatingSystemAdministrationTools
  - PetrolManipulation
  - UserManipulation
  - ServerManipulation
- **MapService**
  - DisplayPetrolLocation
  - DisplayNearByPetrols
  - DisplayCurrentUserLocation
- **DistanceFinder**
  - ComputeDistance
  - FindMapLocation

- **Filter**
  - FilterByName
  - FilterByTypesOfOil
  - FilterByWorkingHours
  - FilterByInfoInput
- **Sorter**
  - SortByName
  - SortByTypesOfOil
  - SortByWorkingHours
  - SortByCurrentUserLocation
- **Search**
  - SearchPetrols
- **GeoInformationManager**
  - SendPetrolList
  - ComputeErrors
  - ComputeSize
  - ComputeLength
- **UserManager**
  - RegisterUser
  - LoginUser
- **AdministrationManager**
  - AddPetrol
  - RemovePetrol
  - UpdatePetrol
  - AddUser
  - AddUserToRole
  - AddUserRoles
  - ControlProcesses
  - ListWorkingProcesses
  - TerminateProcesses
  - UpdateDatabase
  - ResetDatabase

1.1.5 Модел на однесување



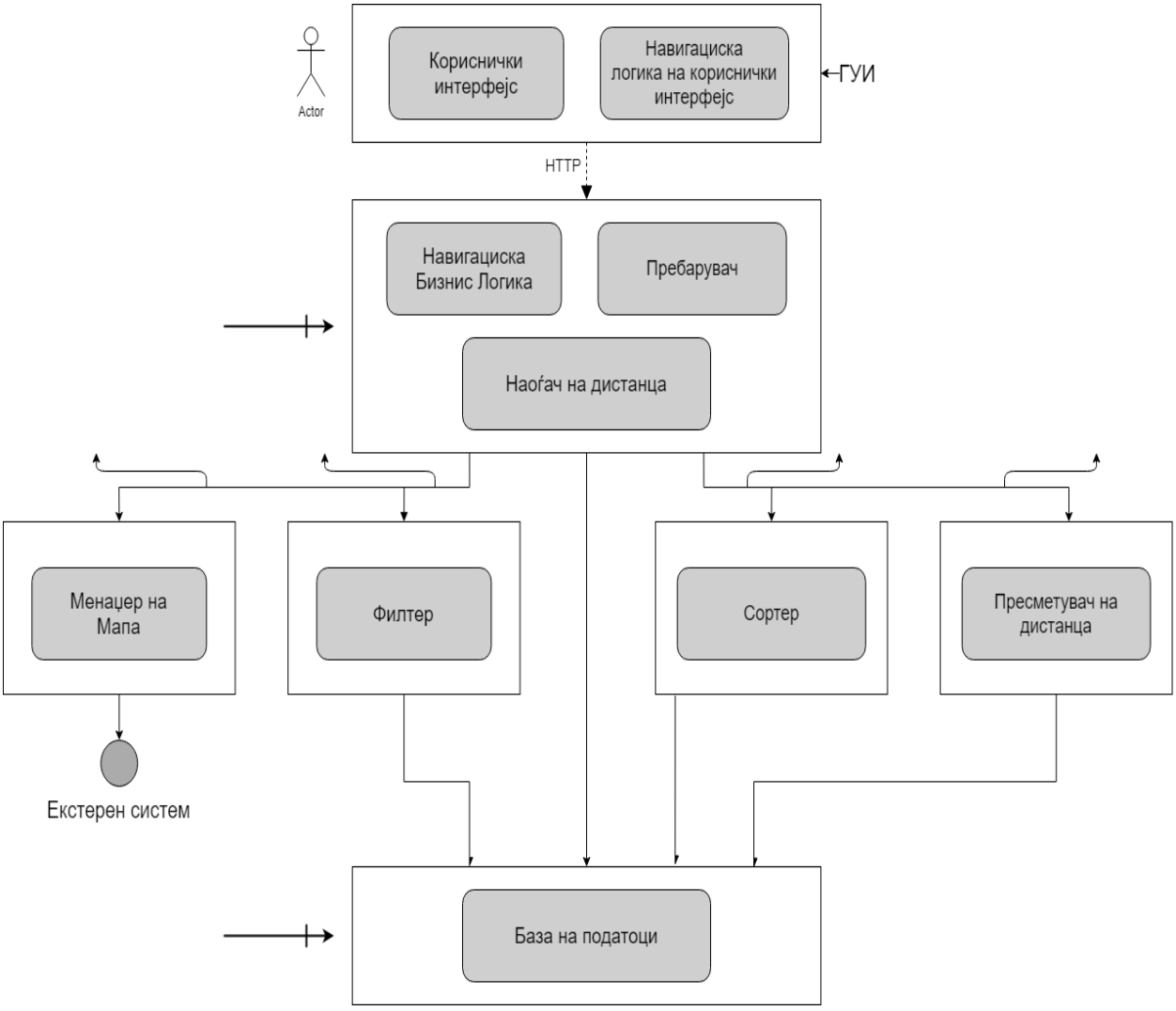
Слика 2: Модел на однесување на стандарден корисник



Слика 3: Модел на однесување на администратор

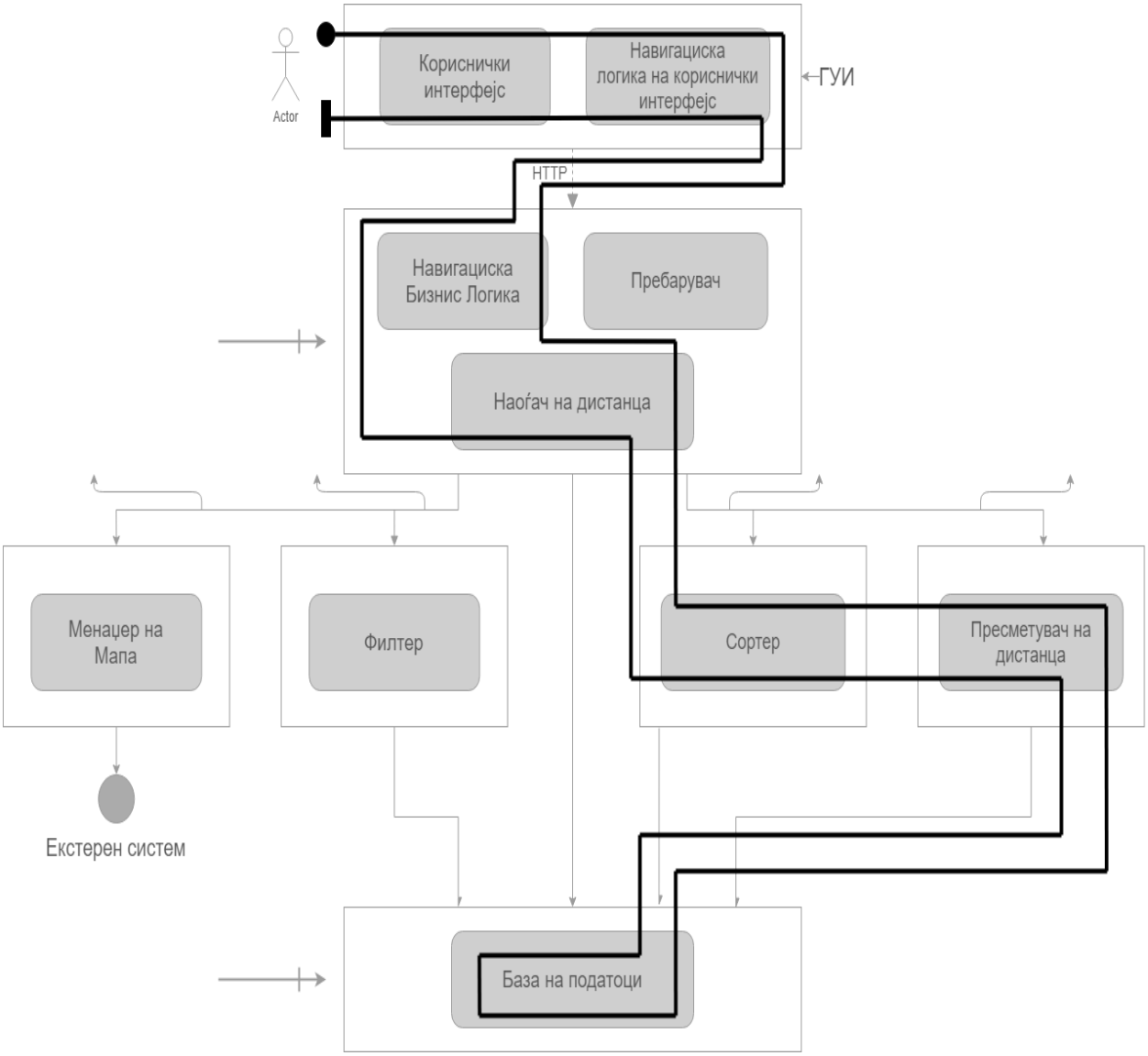
1.2 Извршна архитектура

1.2.1 Главна шема

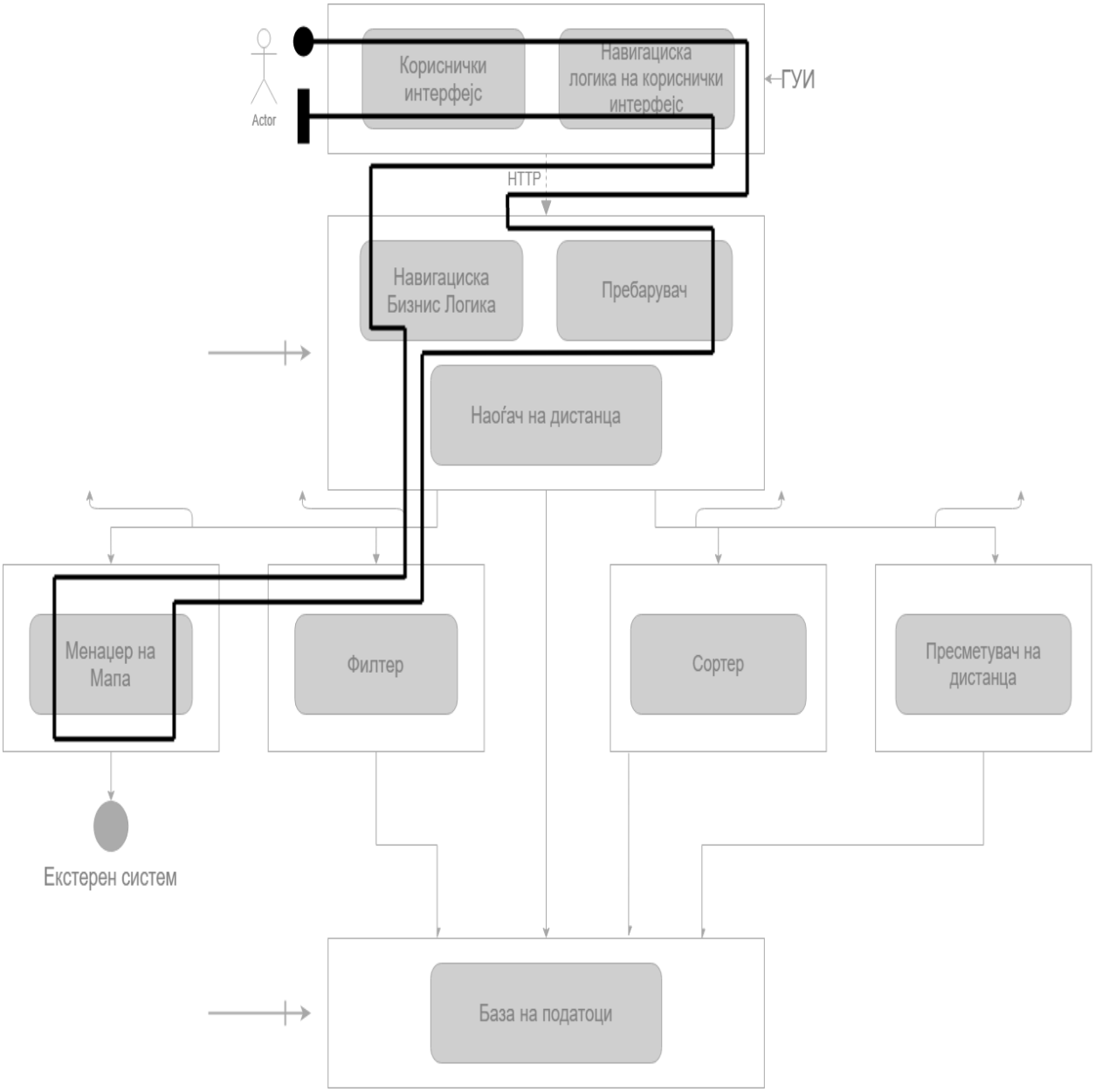


Слика 4: Шема на извршна архитектура

1.2.2 Модел на однесување



Слика 5: Модел на однесување на сортирање и пресметување на дистанца за дадена локација



Слика 6: Модел на однесување за прикажување на внесената локација на корисникот на мапа

1.3 Имплементациска архитектура

1.3.1 Пондериран метод на бодување

Група на критериуми	Критериум	Код
Функционалност	Логирање и пристап	F0
Функционалност	Права на дозвола и управување со дозволи	F1
Функционалност	Управување со содржина	F2
Функционалност	“Кошничка со продукти” одлика	F3
Функционалност	Управување со каталог на производи	F4
Функционалност	Карактеристика за споредба на производи	F5
Функционалност	Опција за пребарување	F6
Функционалност	Опција за сортирање	F7
Функционалност	Опција за филтрирање	F8
Функционалност	Опција за внесување податоци во форма	F9
Функционалност	Опција за оставање повратна информација	F10
Квалитет	Точност при пресметки	F11
Квалитет	Адаптабилност / Квалитет на изворен код	Q0
Квалитет	Доверливост	Q1
Квалитет	Безбедност	Q2
Квалитет	Технологија / Програмски јазик	Q3
Употребливост	Пријавување на грешки	U0
Употребливост	Кориснички интерфејс / Едноставност	U1
Продавач	Референци	V0
Цена	Цена на инсталација и имплементација	C0

	Критериумско пондерирање			SW Резултат на пакет			
Код	**AVG**	**RW**	**WSM**	**ASP.NET**	**Java Spring**	**Django**	**Ruby on Rails**
F0	5%	3	5%	5	3	3	1
F1	5%	3	5%	4	4	2	2
F2	5%	1	2%	3	3	3	3
F3	5%	1	2%	4	1	2	3
F4	5%	3	5%	5	3	2	1
F5	5%	5	8%	4	4	2	2
F6	5%	5	8%	4	4	2	2
F7	5%	5	8%	4	4	2	2
F8	5%	5	8%	4	4	2	2
F9	5%	3	5%	4	4	2	2
F10	5%	3	5%	3	3	1	4
F11	5%	5	8%	3	5	1	1
Q0	5%	4	7%	5	4	2	2
Q1	5%	3	5%	4	4	2	2
Q2	5%	2	3%	5	3	2	2
Q3	5%	1	2%	4	5	2	1
U0	5%	2	3%	4	5	1	1
U1	5%	1	2%	5	3	2	2
V0	5%	1	2%	3	3	3	1
C0	5%	4	7%	4	4	4	4
			Просек	**4.05**	3.65	2.00	2.00
			Пондериран Метод на бодување	**4.17**	3.95	2.34	2.03

- Врз основа на овој критериум ја избираме *ASP.NET* како развојна околина

1.3.2 Апликациски компоненти

Правиме 1-1 мапирање на концептуалните компоненти од концептуалната архитектура, со одредени исклучоци:

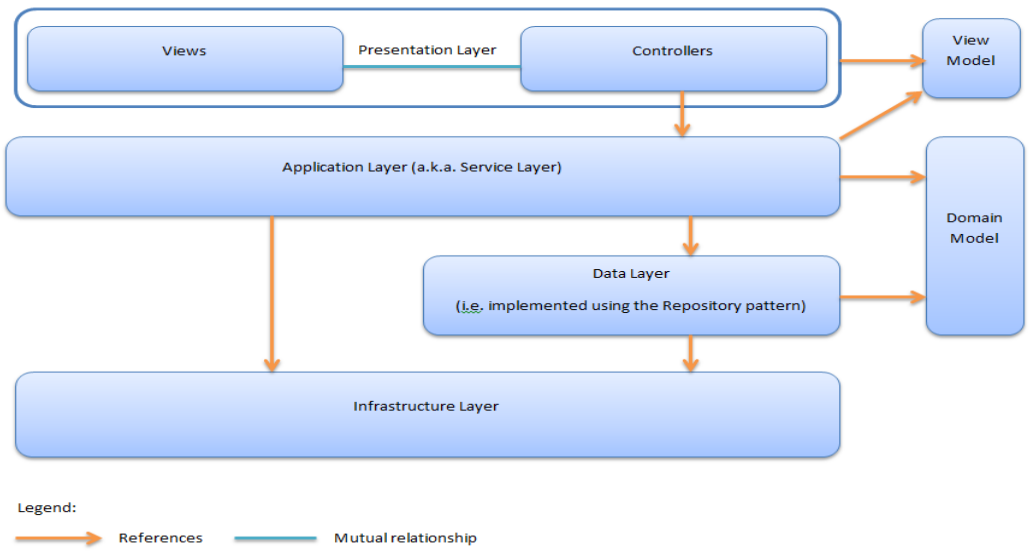
- **AppUI**
  - ShowPetrols
  - DisplayMap
  - ChoosePetrol
  - ShowPetrolInformation
- **AdminPanel**
  - ListPetrols
  - ListConnections
  - ListTools
- **NavigationService**
  - Filter
  - AddInfo
  - Sort
  - DistanceComputed
  - ShowDistance
- **AdminService**
  - PetrolManipulation
  - UserManipulation
  - ServerManipulation
- **MapService**
  - DisplayPetrolLocation
  - DisplayNearByPetrols
  - DisplayCurrentUserLocation
- **DistanceFinder**
  - ComputeDistance

- **Filter**
  - FilterByName
  - FilterByTypesOfOil
  - FilterByWorkingHours
  - FilterByInfoInput
- **Sorter**
  - SortByName
  - SortByTypesOfOil
  - SortByWorkingHours
  - SortByCurrentUserLocation
- **Search**
  - SearchPetrols
- **GeoInformationManager**
  - SendPetrolList
  - ComputeErrors
  - ComputeSize
  - ComputeLength
- **UserManager**
  - RegisterUser
  - LoginUser
- **AdministrationManager**
  - AddPetrol
  - RemovePetrol
  - UpdatePetrol
  - AddUser
  - AddUserToRole
  - AddUserRoles
  - UpdateDatabase
  - ResetDatabase

1.3.3 Инфраструктурни компоненти

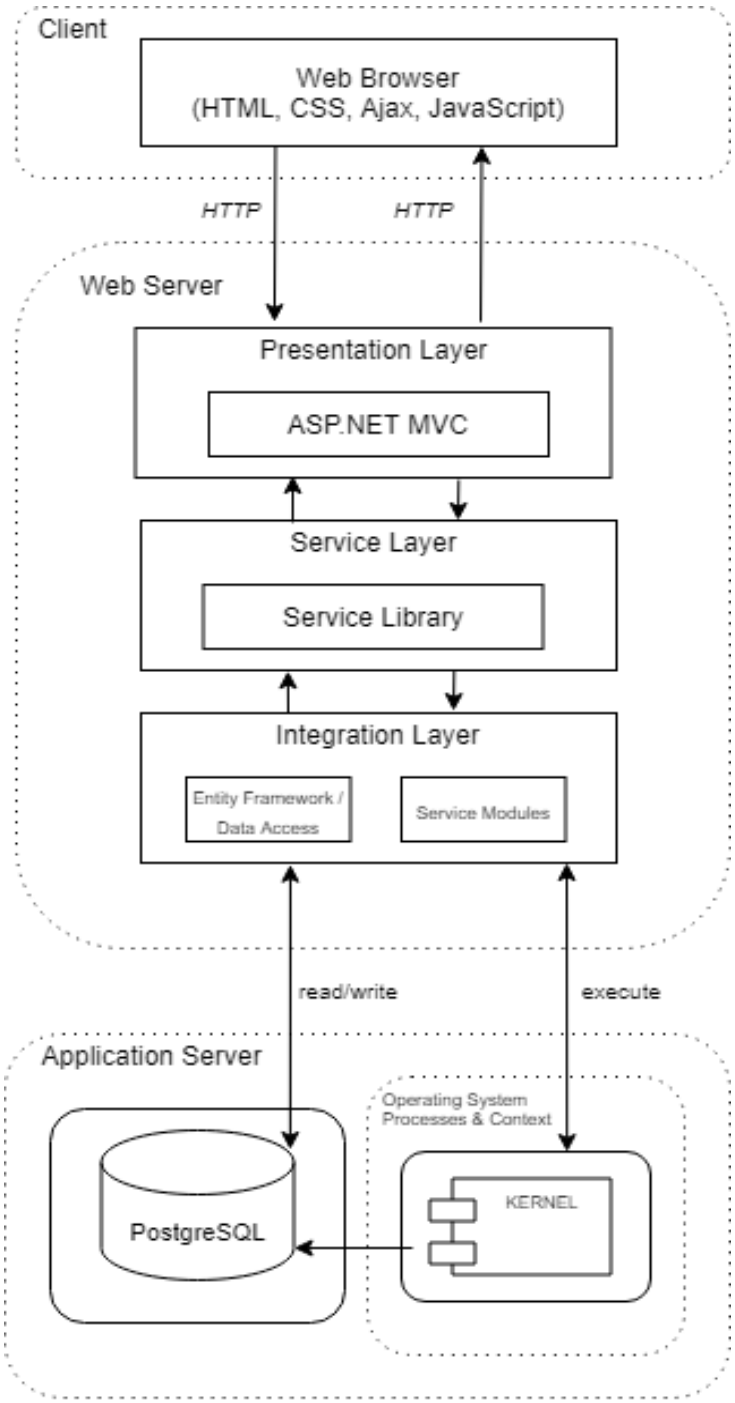
Врз основа на пондерираниот метод на бодување, како и други анализи на архитектурата, се одлучува да се работи со следните инфраструктурни компоненти:

- *Python* и *Bash* скрипти за филтрирање на податоците и сместување во база
- *ASP.NET* како главна компонентна развојна околина имплементација на секој од поодделните веб слоеви, вклучително, бек-енд и фронт-енд
- *PostgreSQL* база на податоци
- Лично-развиени алгоритми за пресметување на дистанца, сортирање, филтрирање и останати функционалности коишто ги нуди апликацијата
- *C#*, *JavaScript*, *JQuery*, *Razor*, *HTML*, *XML*, *PostgreSQL*, *Python*, *CSS* технологии за развој
- *HTTP* протокол за комуникација на апликациски слој
- Компонентно разделување
- *CLR* (Common Language Runtime)
- *UPC* (Unified Programming Classes)
- Web Services (APIs)



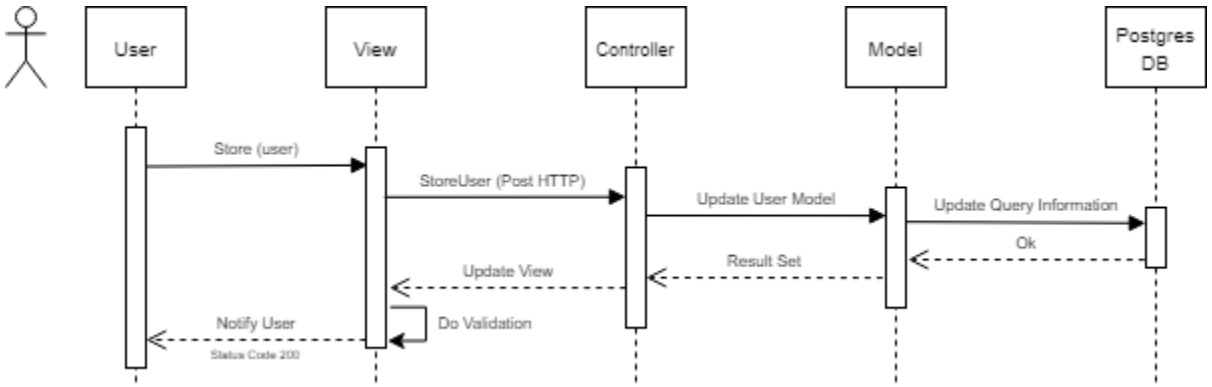
Слика 7: Инфраструктура на ASP.NET развојна околина

1.3.4 Главна шема

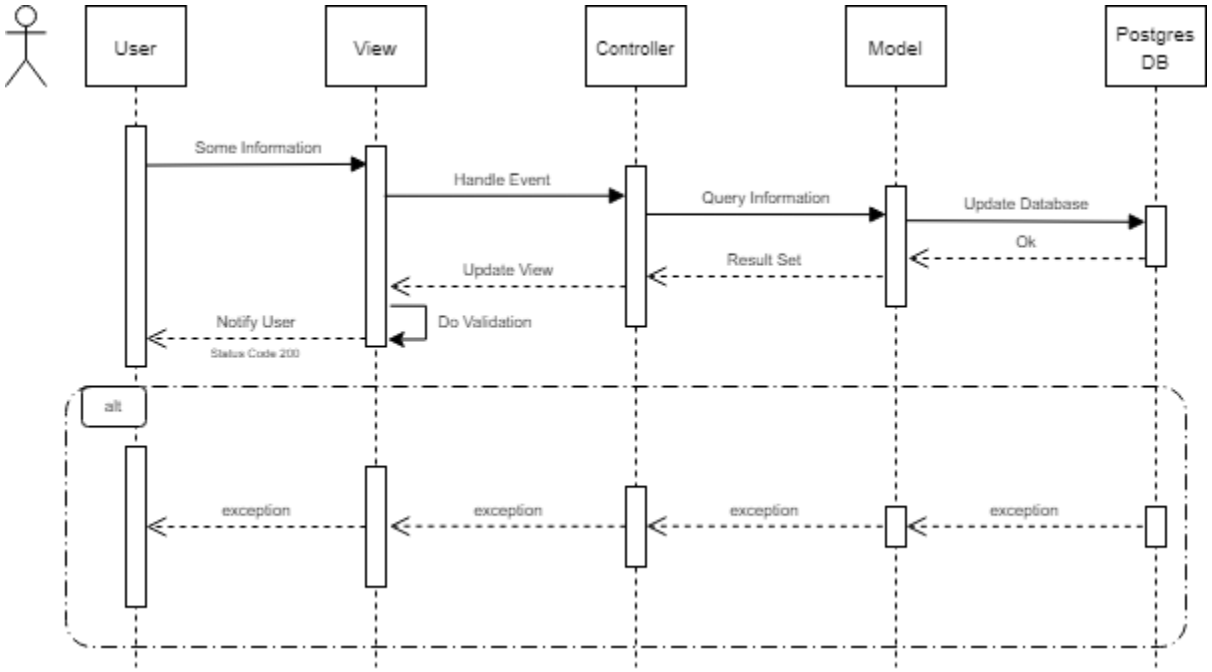


Слика 8: Шема на имплементациска архитектура

1.3.5 Модел на однесување



Слика 9: Модел на однесување (Секвенцен дијаграм) на регистрација на нов корисник во системот



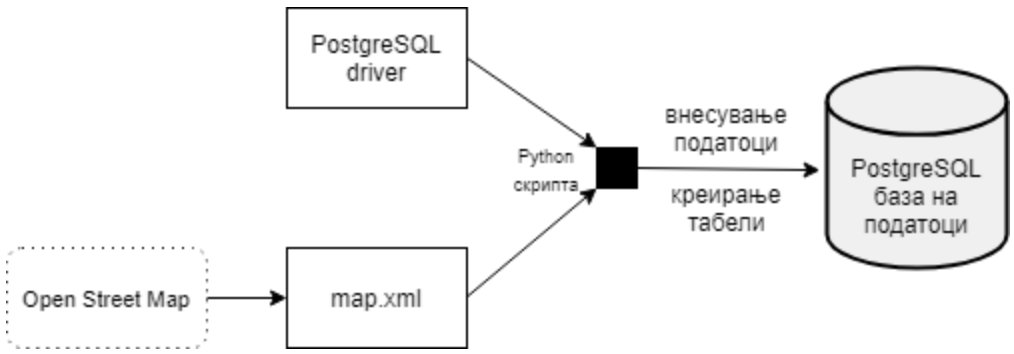
Слика 10: Абстрактен модел на однесување на имплементациска .NET архитектура

2. Архитектурни стилови

Апликацијата има хибридна архитектура и како главни подархитектури во неа спаѓаат: Архитектура на податочен проток (Цевка и филтер), Слоевита тернарна MVC веб архитектура, Податочно центрирана архитектура (База на податоци), Норификациска архитектура, Дистрибуирана архитектура, Микросервисна архитектура со контејнеризација

2.1 Цевка и филтер

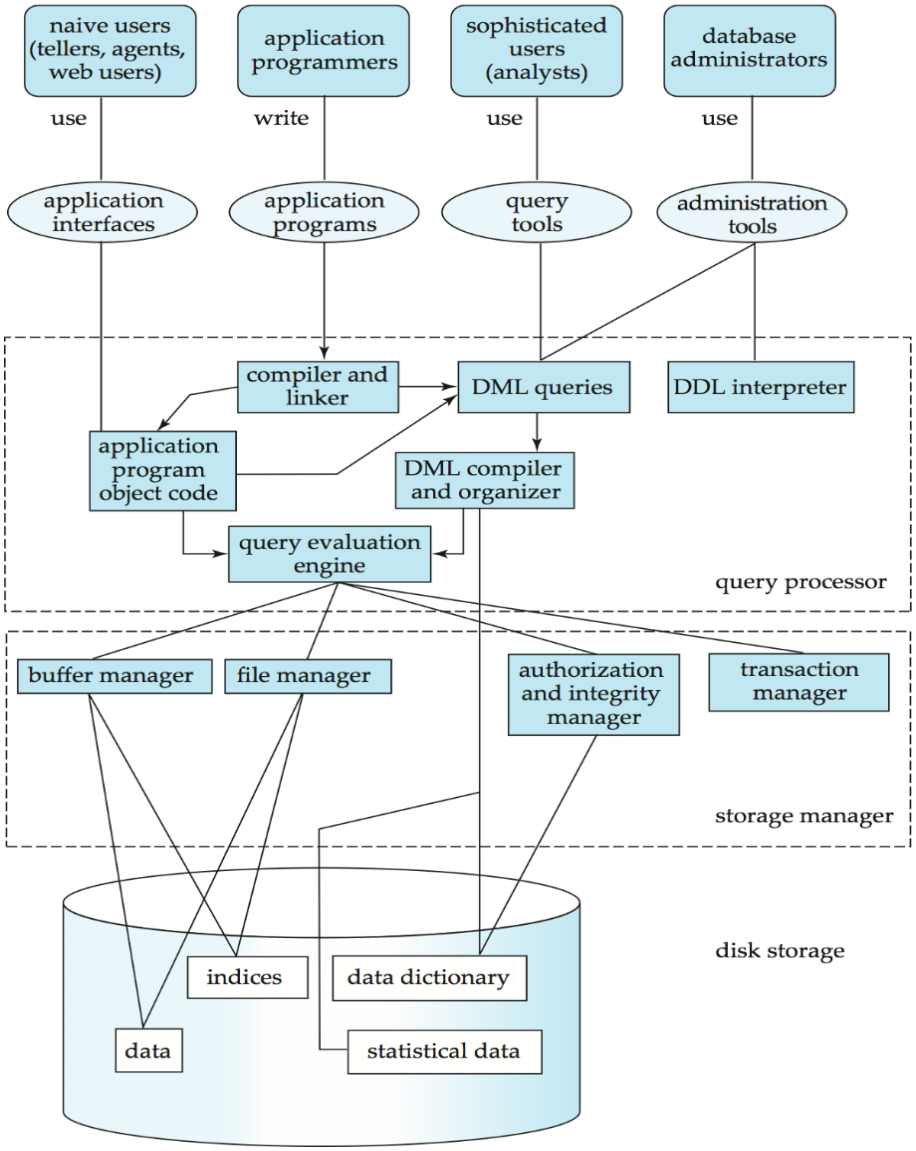
Апликацијата користи *Python* скрипти за филтрирање на податоците и нивно сместување во базата на податоци која е *PostgreSQL*. Исто така, инстанцирањето, креирањето како и правењето на табелите во базата на податоци е напревено преку истата технологија. Иницијалниот извор на податоците се преземени од www.openstreetmap.org којшто е веб страница со достапни геоинформации во некаков структуриран формат во којашто е вклучена територијата на цел свет. За потребите на нашата апликација и со цел кореспондирање со технологиите што се искористени во неа, ние ги преземавме податоците за целата територија на Република Северна Македонија во *.xml* формат.



Слика 11: Цевка и филтер архитектура на апликацијата

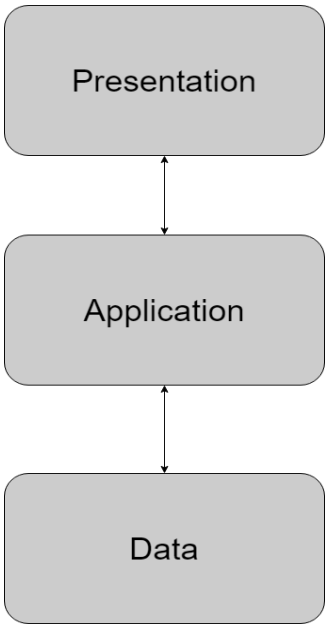
2.2 База на податоци

Апликацијата користи PostreSQL релациска RDBMS база на податоци. Таа се користи за организација, структурирање, чување, сместување, вршење операции и групирање на податоците за бензински пумпи и корисници. Базата има структура слична на подолу прикажаната:



Слика 12: База на податоци архитектура

2.3 Трислојна MVC веб архитектура



Апликацијата користи трислојна веб архитектура како абстрактна слоевита архитектура. Тоа е еден вид на клиент сервер архитектура со 3 главни подслоеви: презентациски, апликациски, податочен слој.

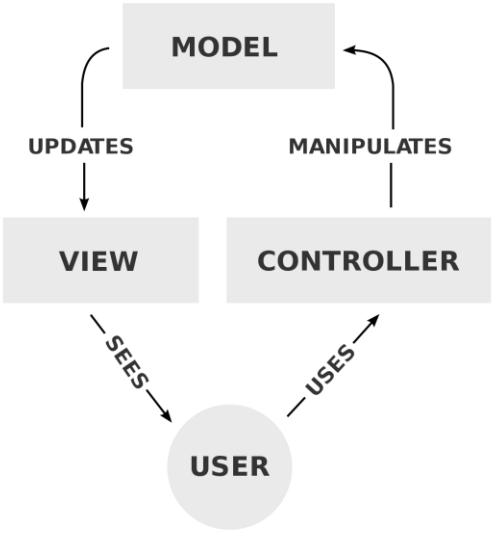
Презентацискиот слој го претставува корисничкиот интерфејс, односно погледите и интеракцијата со корисникот.

Апликацискиот слој го претставува бизнис слојот, односно делот во којшто се имплементира бизнис логиката и соодветното справување со барањата на корисникот.

Податочниот слој е делот којшто се грижи за чување и огранизација на колекциите од податоците.

Слика 13: Абстрактна трислојна архитектура

MVC веб архитектура



Слика 14: MVC веб архитектура

MVC е кратенка за "*Model - View - Controller*". Презентацискиот слој одговара на *View* делот, апликацискиот на Controller, а податочниот на *Model* делот. Во "*Find your petrol*", модели се *User* и *Petrol*, контролери се *AppUIController*, *NavigationController*, *AdminController*, *MapController*, *DistanceController*, *FilterController*, *SorterController*, *SearchController* и *UserManagerController*. Погледи се сите HTML страници кои се прикажуваат на страната на клиентот. Такви се Petrol/InformationInput.html, Petrol/AdminService.html, Petrol/Map.html итн.

2.4 Нотификациска архитектура

Нотификациска архитектура е онаа каде што информации и активности се пропагирани преку нотификациски механизам.

Во апликацијата има повеќе места каде што е имплементирана нотификациската архитектура. Тука спаѓаат: справување со настани во контролерите преку корисничкиот интерфејс, *JavaScript* проверки на страната на клиентот, Тригери во базата на податоци како и проверки на истата за конзистентност.

Справување со настани е една од функционалностите на секоја веб апликација. Тоа е она што ја прави веб апликацијата интерактивна и интересна за сите видови на корисници. Оваа функционалност е овозможена од страна на контролерите, којшто се направени воедно за таа намена. При било каков настан на страната на клиентот со којшто повикува до серверот, се повикува некој контролер, којшто притоа со примена на некаква бизнис логика и комуникација со останатите слоеви од веб апликацијата враќа некоја повратна информација. Таа повратна информација е најчесто *HTML*, меѓутоа во зависност од барањето, има случаеви во којшто се враќа и на пример информации во формат на *XML* или *JSON*.

Тригери и проверки во базата на податоци се прават преку пишување на *PostgreSQL* скрипти во којшто најчесто во оваа апликација се користени креирањето на табели, ажурирањето на табели, бришење на табели, сортирање на податоци од табели и групирање. Бидејќи во *PostgreSQL* базата на податоци нема вграден *consistency checker*, како во многу други, искористен е екстерна алатка *pgcheck*, со цел проверка на интегритетот на податоците.

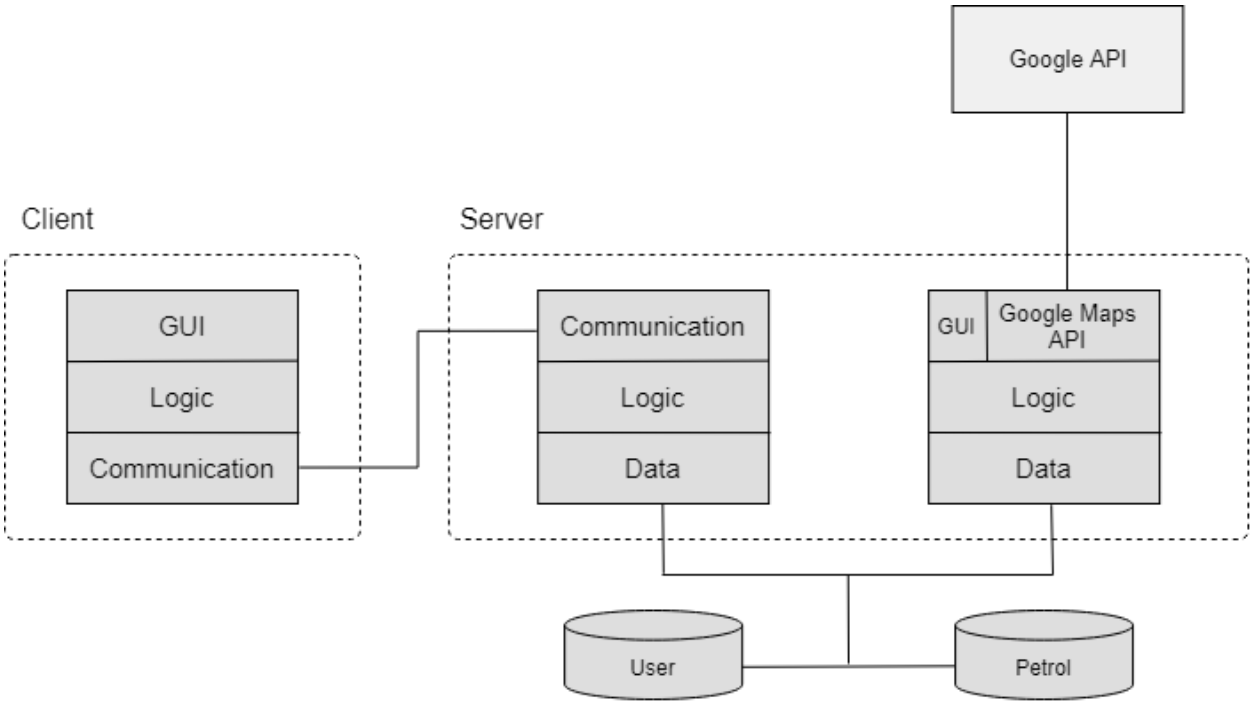
Покрај справувањето со настани на страна на серверот, имаме и имплементирано механизам за кратки проверки и покажување нотификации на страната на клиентот и неговиот веб прелистувач. На пример:

- Анимации за оставање повратни информации
- Анимации за привремено прикажување на копчиња,
- Анимирано упатство за користење,
- Постојани проверки за авторизација,
- Проверки при внесување на податоци во форма

2.5 Дистрибуирана архитектура

Апликацијата ги задоволува сите одлики на дистрибуираната архитектура. Таа е стандардна клиент-сервер ахитектура. Има сериализација: ги зачувуваме податоците и ги правиме статички, имаме членство во група: повеќе клиенти имаат истовремена комуникација со серверот, имаме лидер: серверот, кој задава задачи на процесирачките делови (хардверот) да обработува податоци којшто тој ги дава, имаме споделени податоци коишто се заштитени со синхонизација на процесите, со цел, справување со конкурентност, како и конфигуриран сервер со цел да може да се справи со секој можен вид на барања од страна на клиентот.

Исто така користиме синхрона архитектура бидејки имаме валидација на секој процес што се извршува на страна на серверот, се со цел, да не се прати одговор на клиентот со некаква грешка. Во оваа апликација приоритетот е безбедност и сигурност, а не поголема брзина на процесирање и поголема респонсивност, што прави асинхроната архитектура да се користи само во неколку ретки наврати.



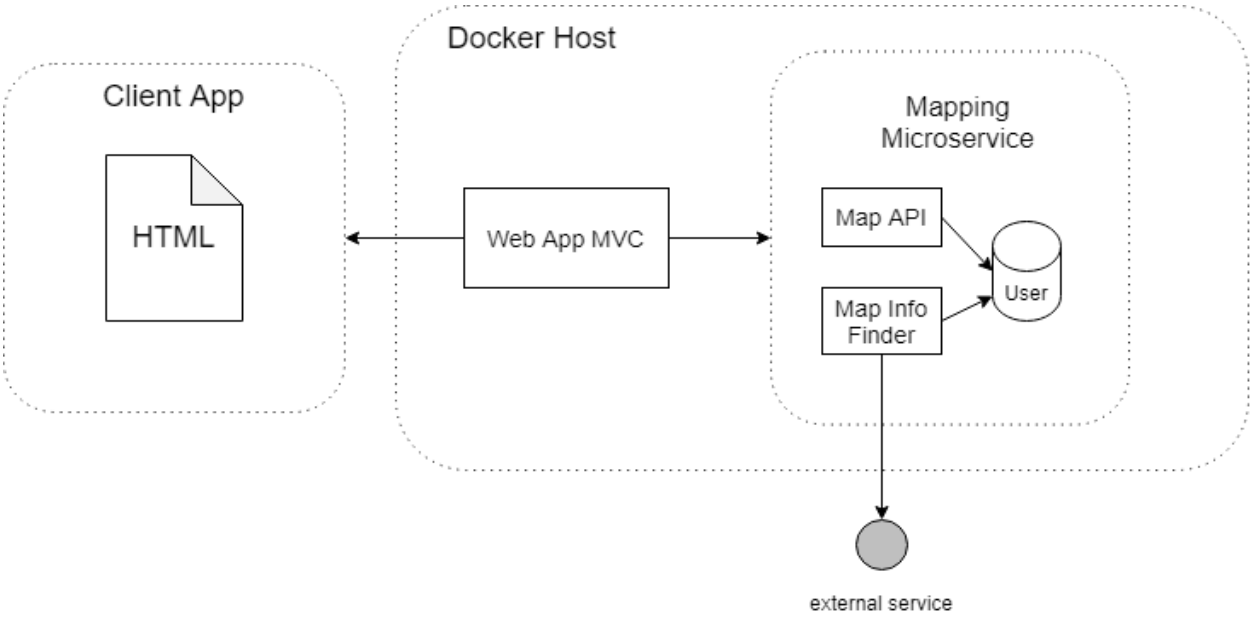
Слика 15: Абстрактно претставена веб архитектура на “Find your petrol”

2.5 Микросервисна архитектура со контејнеризација

За да може да се воспостави портабилност и реискористување на изворниот код, користените библитеки, фајлови и сите податоци, искористен е *Docker*. Искористена е функционалноста на *Visual Studio* и неговата поврзаност со *Azure* сервисите за контејнеризација на апликацијата со помош на *Azure Kubernetes Service (AKS).*

Имаме еден дел од апликацијата којашто користи стриктно микросервисна архитектура. Тоа е контролерот за мапа, којшто, покрај другото и комуницира со друг сервер (*Google API*), за добивање потребни информации. Целта е побрза обработка и подобра интеракција на страната на клиентот. Ова е имплементирано со тоа што контролерот за мапа е *REST API* контролер којшто враќа повратни информации во *JSON* формат, и тие се рендерираат и прикажуваат на страната на клиентот. Ова е воедно и единственото користење на асинхрона архитектура.

Апликацијата користи *OAuth 2.0* за автентикација и авторизација на корисници.



Слика 16: Микросервисна архитектура на прикажување на мапа