**ДИЗАЈН И АРХИТЕКТУРА НА СОФТВЕР**

Домашна 2

**ПРОФЕСОР: ТИМ:**

Љупчо Антовски Јорданче Ефтимов 211069

Никола Анѓеловски 211071

Марија Аврамоска 211004

Стефани Ефтимовска 211062

* **КОНЦЕПТУАЛНА АРХИТЕКТУРА**

1. Целта на нашата Веб апликација е да обезбеди детален преглед на винарската индустрија во Република Македонија. Корисниците ќе имаат можност да истражуваат и добиваат инфомации за различни винарии, нивните производи, локацијата на која се наоѓаат, искуства на посетителите и други релевантни информации. Главната идеја е да се олесни процесот на истражување на винарии и производи.
2. Веб апликацијата ќе биде интерактивна и лесна за користење. Корисниците ќе имаат пристап до следниве функционалности:

* Пребарување на винарии: Корисниците ќе можат да прегледуваат листа со винарии, филтрирана според различни критериуми како што се град или работно време. За секоја винарија ќе има детален преглед со информации како што се адреса, контакт, искуства и слично.
* Интерактивна мапа: Апликацијата ќе има интегрирана мапа на Република Македонија која ќе ги прикажува локациите на сите винарии. Корисниците ќе можат да кликнат на одредена област на мапата и да добијат листа со винарии во тоа подрачје.

1. Клучни засегнати страни и нивни улоги:

* Посетители- главни корисници на апликацијата кои пребаруваат винарии
* Винарии- сопственици на винариите кои можат да додаваат инфрмации за нивните производи

1. Концептуален дијаграм



* **ИЗВРШНА АРХИТЕКТУРА**
* ***Слоеста Веб архитектура***- оваа архитектура овозможува поделба на одговорности помеѓу различнитеслоеви што ја прави апликацијата оддржлива.

1. Презантациски слој- овој слој е одговорен за тоа како ќе се прикажат информациите до коирсниците, и како ќе се обработат нивните акции.

-Користени технологии: HTML, CSS и JavaScript за изградба на корисниччи интерфејс, картографска библотека Google Maps API за прикажување на мапата, Canva за приказ на UI/UX дизајн.

1. Бизнис логика- овој слој ги содржи сите функционалности и бизнис правика на апликацијата. Ја вклучува логиката за пребарување и прикажување на винарии, обработка на кориснички барања и вршење на додатни бизнис процеси.
2. Податочен слој- овој слој манипулира и чува податоци. Одговорен е за комуникација со база на податоци и управување со податоците кои се користат низ целата апликација.

-Користени технологии: PostgreSQL database

* ***Микросервисна архитектура***- користи многу мли, автономни сервиси што работат заедно и комуницираат помеѓу себе преку API.

1. Modularity: апликацијата се состои од мали сервиси, секој одговорен за одредена функционалност, како што се микросервис за винарии, за корисници и слично.
2. Scalability and Reusability: секој микросервис ќе биде деплоиран и скалиран независно, овозможувајќи грануларна контрола на перформансите. Повторно искористување имаме на сервисите во различни делови од апликацијата.
3. Independent Development: се овозмжува побрз разбој и воведување на нови функционалности во апликацијата преку работа на различни тимови независно еден од друг.
4. Technological Diversity: секој микросервис се развива со различни технологии, што го олседнува изборот на правилната технологија за секоја специфична функционалност.
5. Rapid Development Cycle: Бидејќи сервисите се независни, нивната имплементација, тестирање и воведување можат да бидат изведени брзо и ефикасно.

-*Структура на микросервисната архитектура:*

1. Сервис за винарии: овој сервис содржи информации за винариите, нивните производи и локации.
2. Сервис за мапа: ја прикажува мапата и интегрира со картите.
3. API Gateway: Главна точка за влез во сисемот, што мнипулира со внатрешните и надворешните барања и ги прследува кон соодветниот сервис.
4. База на податоци: нерелациона абза на поатоци која се користи од сервисите за чување на информации.
5. Дизајн на база на податоци: нашата апликација користи Google Places API за да истражува винрии блиски до одредена локација во Македонија и потоа враќа детални информации за секоја винарија, како име, адреса, телефонски број, работно време и слично. Базата на податоци е зачувана во CSV фолдер наречен wineries\_0.csv.

* **ИМПЛЕМЕНТАЦИСКА АРХИТЕКТУРА**

1. Pipe and filter ахритектура:

Кодот содржи класа Pipeline која е централен компонент, и таа управува со серија на филтри. Методот add овозможува доавање на повеќе филтри, додека методот execute се итерира низ филтрите, аплицирајќи го секој филтер на влезните податоци последователно. Секоја функција за филтрирање врши специфична трансформација на влезните податоци. Овие филтри се независни и можат да се комбинираат на различни начини користејќи ја класата Pipeline. Податоците, вчитани од CSV датотека поминуваат со користење на методот execute. Филтерот filter\_duplicates отстранува дупликати врз основа на колоните Id и Name. Филтерот business\_status мапира статус на бизнис вредности, додека филтерот filter\_working\_hours ја обработува колоната Opening hours форматирајќи ги работните часови и е справува со специјални знаци. Филтрите се споени заедно, што овозможува модуларна и повторлива структура. Резлтатот е трансформиран како DataFrame(‘result’) ој подоцна се зачувува во нова CSV датотека.

* **ДИСТРИБУИРАНА АРХИТЕКТУРА**

1. ***Скалабилност***

* Load Balancing: Користиме load-balancer кој распределува барања од корисниците кон повеќе инстанци од апликацијата. Ова овозможува рамномерно оптоварување на серверите и подобри перформанси.
* Кеширање: Се користи кеш за складирање на често користени податоци како што се информациите за винариите ин самите локациии. Ова ни помага да го намалиме оптоварувањето врз базата на податоци и да овозможиме брзо проследување на информаци до корисниците.
* Дистрибуирани бази на податоци: податоците се чуваат во дистрибуирани бази на податоци за осигурување на скалабилност и отпорност на грешки.

1. ***Отпорност на грешки***- овие стратегии и технологии работат заедно за да обезбедат отпорност на грешки и да овозможат скалабилност на аплиакцијата.

* Стратегии за управување со грешки- користиме стратегии за управување со грежки како retry механизам за повторен обид на барања, circuit breakers за спречување на надокнадно фрлање на грешки и fallback механизми за користење на алтернатвни логики кога главниот сервис е недостапен.
* Редунрантност- секој од сервисите е развиен да биде самостоен и да може независно да функционира. Ако еден севрис не работи, останатите сервиси продолжуваат да функционираат.
* Механизмми за резервно копирање- редовно се прави backup на податоците и конфигурациите.