

# Архитектурни стилови

## Data-flow Architectures

Во проектот се користат Pipes and filters. Концептуално филтрите земаат податоци од внесување и запишуваат податоци на излез. Филтрите не знаат ништо за другите филтри кои постојат (loose coupling).

Pipes се добри затоа што ќе ни помагаат при одржување на системот. Голема предност би била тоа што компонентите може повторно да се искористат, и исто така истите можат да се додаваат нови и да се отстрануваат постоечките.

## Network-Centered Style

Во овој проект ќе го користиме клиент-сервер моделот. Овој архитектурен стил покажува како податоците и сервисите се дистрибуиран на неговите сервери.

Карактеристика на клиент-сервер моделот е тоа што ја опишува поврзаноста меѓу програмите во апликацијата. Серверот ќе им овозможи функции или сервиси на клиенти кои пратиле барање за истите. Во нашиот случај серверот чека да биде исконтактиран од корисникот. Корисникот испраќа барање и чека да биде опслужен.

Клиент-сервер архитектурата ќе ни овозможи улогите на компјутерскиот систем да бидат дистрибуирани низ повеќе независни компјутери/мобилни уреди кои се препознаваат меѓусебно во мрежата. Со оваа архитектура ќе ни се олесни одржувањето на системот. Со овој модел ќе ни се овозможува системот да се премести, поправи, надогради, или да се релоцира опслужувачот, доколку клиентите имаат доволно знаење, и немаат влијание од направената промена.

Корисниците пристапуваат до нашите сервиси преку своите компјутери, користејќи пребарувач за да испратат барање до нашиот опслужувач. Таа програма може да го препрати барањето од неговата програма - клиент, која ги чува податоците, која пак испраќа барање до опслужувачот со бази на податоци на друг сервис, за да ја врати информацијата на корисникот. Состојбата се враќа до базата на податоци на нашиот сервис, која за возврат ја испраќа назад до пребарувачот-клиент, прикажувајќи му го резултатот на корисникот.

Чувањето на податоците ќе биде централизирано со што обновувањето на тие податоци ќе ни биде многу полесно за разлика од P2P мрежите.

## **Data-centered Architectures**

Затоа што системот има за цел да го постигне квалитетот на интеграбилност на податоците, се користи Data-centered Architecture. Имаме една централизирана “data store” која што комуницира со голем број на клиенти. Протоколи кои ни се важни за овој стил се протоколите за комуникација, дефиниција и манипулација со податоци.

## **Component Architecture**

Друга архитектура која ќе биде применета во нашиот проект е Component архитектурата. Архитектурата базирана на компоненти се фокусира на разградување на дизајнот на поединечни функционални или логички компоненти кои претставуваат добро дефинирани комуникациски интерфејси кои содржат методи, настани и својства. Обезбедува повисоко ниво на апстракција и го дели проблемот на потпроблеми, секој поврзан со партиции на компонент

## **Multilayer architecture**

Во проектот имаме применето и N-tier architecture. Се поврзува со Веб апликацијата којашто ја градиме повеќеслојната софтверска архитектура сè уште има презентациски слој и слој на податоци. Едноставно се дели и го проширува слојот на апликацијата. Овие дополнителни аспекти во апликацискиот слој се во суштина различни услуги. Ова значи дека нашиот софтвер сега треба да биде поскалабилен и да има дополнителни димензии на функционалност.