**Документация на СА**

**Приложение за запознанства**

**изготвена от:**

**Пламена Топалова, ФН: 471219077**

**Илиян Дрянов, ФН: 471219041**

**Николай Каймакански, ФН: 471219072**

**Чавдар Горанов, ФН: 471219040**

Съдържание

[1.Въведение 3](#_Toc94455529)

[1.1.Обхват 3](#_Toc94455530)

[1.2.Общ изглед 3](#_Toc94455531)

[2.Архитектурно представяне 4](#_Toc94455532)

[2.1.Особености на случаите на употреба(Use Case view) 5](#_Toc94455533)

[2.2.Особености на логическия изглед(logical view) 5](#_Toc94455534)

[2.3.Особености на изгледът на процесите(process view) 5](#_Toc94455535)

[2.4.Особености на изгледът за разработка (development view) 6](#_Toc94455536)

[2.5. Особонести на физическият изглед(deployment view) 6](#_Toc94455537)

[3.Use Case изглед 6](#_Toc94455538)

[3.1.Актьори 6](#_Toc94455539)

[3.2.Сценарии 6](#_Toc94455540)

[4.Логически изглед 7](#_Toc94455541)

[4.1. Component диаграма 7](#_Toc94455542)

[4.2. Class диаграма 8](#_Toc94455543)

[4.2.Диаграма на състоянията 9](#_Toc94455544)

[5.Изглед на процесите 9](#_Toc94455545)

[5.1. Диаграма на последователността 10](#_Toc94455546)

[5.2.Activity диаграма 11](#_Toc94455547)

[6.Основни цели и ограничения на избраната архитектура 11](#_Toc94455548)

[6.1.Функционални изисквания 11](#_Toc94455549)

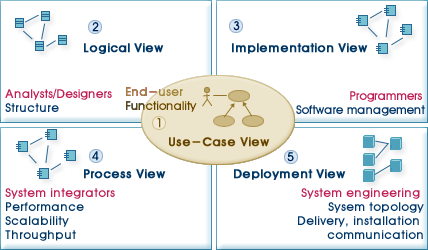
[6.2.Нефункционални изисквания 12](#_Toc94455550)

[6.3.Как Data-centered architecture style спомага за постигане на поставените изисквания 12](#_Toc94455551)

# 1.Въведение

## 1.1.Обхват

Настящата работа е общ преглед и обяснение на софтуерната архитектура на Приложение за запознанства. В работата се представят целите на архитектурата, избраните архитектурни стилове и случаите на използване, които са поддържани от системата. Описват се редица различни архитектурни възгледи, за да се изобразят всички аспекти на дизайна на системата, които се считат за архитектурно значими и важни за заинтересованите лица. Структурата на работата ще се основава на Philippe Kruchten’s “4+1” model view. Този модел ще позволи на всички заинтересовани лица да намерят лесно това, което им е нужно в софтуерната архитектура.



## 1.2.Общ изглед

Документацията съдържа следните 5 глави, описващи следното:

Глава 2 – представя избрания архитектурен стил, различните изгледи и тяхната употреба

Глава 3 – представя use case изгледът

Глава 4 – представя логическия изглед на системата

Глава 5 – представя изгледът на процесите

Глава 6 – представя основните функционални и нефункционални изисквания на системата, както и поставените цели, които оказват значително влияние върху избраната архитектура.

# 2.Архитектурно представяне

За приложението за запознанства ще използваме Data-centered architecture style, в частност Client-Server – N-tier architecture pattern.

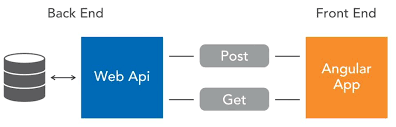
Какво трябва да знаем за този архитектурен стил:

* В този стил има централно хранилище на данните.
* Множество независими компоненти, които работят с това централно хранилище.
* Взаимодействието на отделните независими компоненти се осъществява чрез това хранилище.
* Всички компоненти сами проверяват за промените по данните в хранилището.
* Компонентите нямат пряка комуникация помежду си.
* Този стил позволява паралелна работа на различните компоненти.

Какво трябва да знаем за този архитектурен шаблон:

* Функциите за представяне, обработка и управление на данни са логически и физически разделени.
* Всяка от тези функции се изпълнява на отделна машина или отделни клъстери, така че всяка да може да предоставя услугите с най-висок капацитет, тъй като няма споделяне на ресурси.
* Това разделяне прави управлението на всяка функция поотделно по-лесно, тъй като работата по едната не засяга останалите, изолира всички проблеми, които могат да възникнат.

Какво ще използваме за нашето приложение? : NET, Angular, Entity Framework, HTML, CSS, TypeScript, C#.



Стилът допринася за изпълнението на основни архитектурни цели и изисквания, които имаме, като:

* Мащабируемост - може да се променя за отделните нива
* Разширяемост - може да се разширява по всякакъв начин според изискванията
* Сигурност - всяко ниво може да бъде защитено отделно и по различни начини
* Лесен за управление – предоставя възможността да се управлява всяко ниво поотделно, като се добавя или променя всяко ниво, без да се засягат другите нива.

Избраният модел за описване на архитектурата на приложението за запознантсва е базиран на 4+1 изгледа, които се използват за описание на системата от гледна точка на различни заинтересовани страни. Те са логически изглед(logical view), изглед за разработка(development view), изглед на процесите(process view), изглед на внедряване/физически изглед(deployment view) и допълнителният един е случаи на употреба(use case view). Ще разгледаме какво описва всеки от изгледите, за кого е предназначен и кога се използва.

## 2.1.Особености на случаите на употреба(Use Case view)

Този изглед е предназначен за всички заинтересовани страни в приложението, включително и крайният потребител в качеството си на заинтересовано лице. В него се описват действащите лица и целият набор от различни случаи на употреба(сценарии), представляващи важни и основни функционалности на системата. Сценариите представят изискваният и нуждите на крайният потребител на системата, като също така служат за отправна точка за тестове на прототип на архитектурата. Те се представят използвайки UML диаграми, в частност с use-case диаграма. Случаите на употреба се разглеждат също и по-нататък в проектирането на системата.

## 2.2.Особености на логическия изглед(logical view)

Логическият изглед е предназначен за разработчиците и проектантите на приложението, съответно заинтересованите от него лица го използват главно по време на проектиране и разработка на системата. Изгледът представя функционалността, която системата предоставя на крайните потребители, т.е. описва структурата на системата, всички функционални и бизнес изисквания на системата. В тази “гледка” системата се разделя на основни кампоненти. За представяне на изгледа се използват UML диаграми, в частност диаграми на класове, диаграми на състоянието и компонентни диаграми.

## 2.3.Особености на изгледът на процесите(process view)

Изгледът на процесите е предназначен за системните интегратори. Този изглед представя всички динамични аспекти на системата, като описва системните процеси и това по какъв начин те си комуникират. “Гледката” е фокусирана върху самото поведение на системата по време на нейното изпълнение, като се отнася до производителността, интегрирането, паралелната обработка и още различни нефункционални изисквания. За представянето на изгледа се използват UML диаграми, в частност: Sequence диаграми, Communication диаграми и Activity диаграми.

## 2.4.Особености на изгледът за разработка (development view)

Изгледът за разработка, както си подсказва и името му, е предназначен за разработчиците на приложението (т.е. програмистите на приложението). Този изглед е известен още като изглед за изпълнение и се занимава със софтуерното управление, като той представя цялата система от гледна точка на програмиста. За описване на изгледа се използват UML диаграми, в частност Package diagram.

## 2.5. Особонести на физическият изглед(deployment view)

Физическият изглед е предназначен за системните инженери и лицата занимаващи се с внедряване, както става ясно, че той представя системата от гледна точка на системния инженер. Този изглед може да се нарече изглед за разполагане и описва разполагането на софтуерните компоненти на физическия слой и физическите връзки между тези компоненти. За описване на изгледа се използват UML диаграми, в частност : Deployment диаграми.

# 3.Use Case изглед

Този изглед представя различини сценарии на употреба, всеки в различен контекст , представяйки взаимодействието между отделните компоненти на приложението. Тези компоненти в случая се наричат и се разглеждат като актьори в различните сценарии на употреба в приложението. В точка 3.1. ще разгледаме кои са актьорите(компоненти) в нашето приложение за запознанства, а в точка 3.2. ще представим един такъв сценарий на употреба, който включва всички актьори.

## 3.1.Актьори

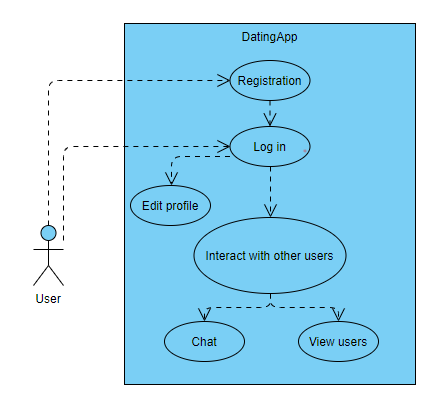
Актьорите в приложението за запознанства не са много различни потребители, а всъщност е фокусирано върху един потребител: потребител(“клиент”) .Ще разгледаме всеки от тях поотделно:

* **Потребител(“клиент”)** – “клиентите” са потребители, които взаимодействат с различни интерфейси на системата, с цел запознанство с други такива потребители.

## 3.2.Сценарии

В приложението за запозснаства има различни сценарии на употреба. Например сценарий за вход в системата, и т.н.

Ще разгледаме use case диаграма, включваща нашите актьори – “клиенти”,при употреба на нашето приложение за запознанства.

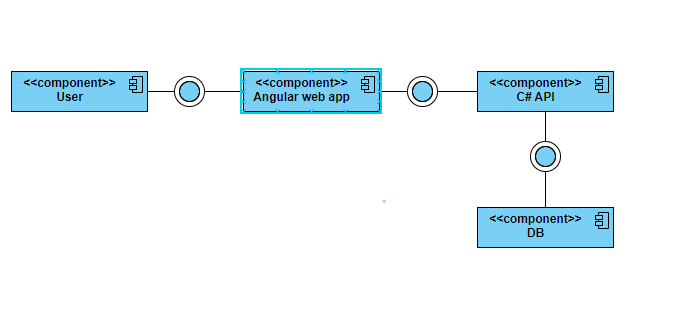


Фиг.1.(UML use-case diagram)

# 4.Логически изглед

Една от основните задачи на този изглед е декомпозирането на системата на приложението на основни изграждащи я компоненти и предстаявеното на това как те ще си комуникират. Комуникацията става чрез предоставяне на различни интерфейси от един компонент на друг. Ще опишем това декомпозиране чрез UML диаграми.

## 4.1. Component диаграма



Фиг.2.(Component диаграма)

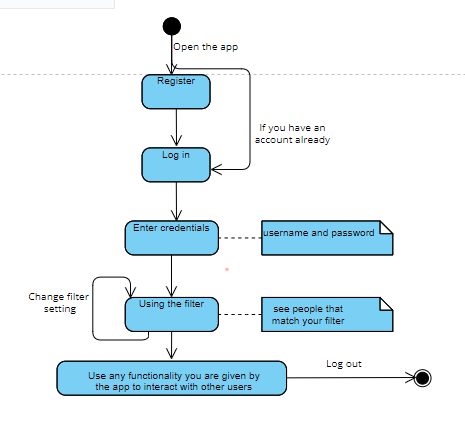
## 4.2. Class диаграма

Както се вижда на фиг.3(UML class diagram), разделяме системата на 6 архитектурно значими класа: AppUser, Message, Photo, UserLike, Group, Connection. Връзките между класовете, техните методи и полета са представени на диаграмата.



Фиг.3.(UML class diagram)

## 4.2.Диаграма на състоянията



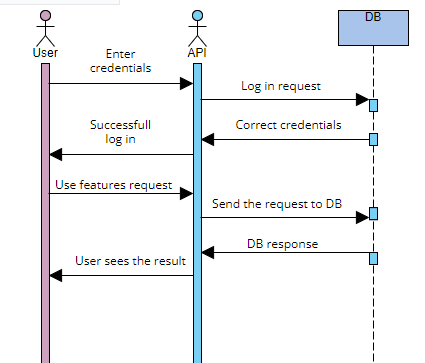
Фиг.4.(State диаграма)

На фиг.4.(Диаграма на състоянията) се описват състоянията и действията при употреба на приложението за запознанства, през които се преминава.

# 5.Изглед на процесите

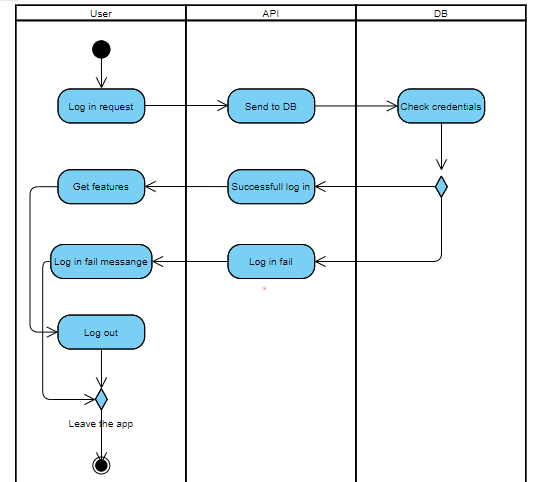
Този изглед се фокусира върху поведението на системата по време на нейното изпълнение. Както знаем, системата има определен набор от изпълними процеси, отговарящи за реализирането на определени задачи. Последователността във времето на изпълнението на тези задачи се представя на фиг.5.(Диаграма на последователността) и на фиг.6.(Activity диаграма).

## 5.1. Диаграма на последователността



Фиг.5.(Диаграма на последователността)

## 5.2.Activity диаграма



Фиг.6.(Activity диаграма)

# 6.Основни цели и ограничения на избраната архитектура

## 6.1.Функционални изисквания

Системата поддържа роля потребител(“клиент”). Функционални изисквания за тази роля са както следва:

* Създава профил в сайта
* Има възможността да редактира своя профил в сайта
* Има възможността да харесва снимките на другите потребители като него
* Има възможнастта да качва свои лични снимки и има опцията да избира коя от тях да използва като профилна снимка.По този начин не се налага всеки път, когато потребителят променя своята снимката да я качва отново
* Има възможността да използва системата за съобщения, която му позволява да комуникира с други потребители(чат)
* Има достъп до страница с потребителите, които харесва и също така страница с потребителите, които харесват него
* Потребителят ще получава в реално време известия, ако получи съобщение от друг потребител
* Може да използва филтър, благодарение на който може да търси други потребители по критериите - възраст и пол

## 6.2.Нефункционални изисквания

Приложението има следните нефункционални изисквания:

* Ефективност – системата да преизползва определени ресурси, за които това е допустимо;
* Скалируемост – системната архитектура позволява лесно преконфигуриране на отделните модули при нарастване на обема на потребителски заявки с цел запазване на определената производителност;
* Сигурност(Security) – защита на обменяните данни
* Мащабируемост(Scalability)
* Разширяемост(Extensibility)
* Възможност за поддръжка(Maintainability)

## 6.3.Как Data-centered architecture style спомага за постигане на поставените изисквания

Data-centered architecture style се отличава със своите по-добри условия за сигурност(Security), поради централизираното управление на данните при този стил. Поради независимостта на компонентите лесно може да се добавят нови такива, което значи, че системата ще бъде лесно мащабируема в случай на увеличаване броят на потребителите й, така и условието за мащабируемост(Scalability) е изпълнено.

**Хранилище за управление на версиите – GitHub:**

https://github.com/reffera12/DatingApp.git