**Архитектурен проект**

Тема:

**Електронно здравно досие**

Група: 77

Проект №7

Автори:

Виктор Георгиев Георгиев, 471219028

Виктория Валентинова Терзиева, 471219067

Вяра Паскалева Паскова 471219036,

Елина Пламенова Календерова 471219051,

Мартин Христов Тодоров 471219013

Съдържание

[Въведение](#_heading=h.gjdgxs) **3**

[Use-Case View](#_heading=h.30j0zll) **3**

[Logical View](#_heading=h.3znysh7) **4**

[Process View](#_heading=h.2et92p0) **5**

[Deployment View](#_heading=h.tyjcwt) **6**

[Data View](#_heading=h.3dy6vkm) **7**

[Нефункционални изисквания](#_heading=h.1t3h5sf) **8**

# Въведение

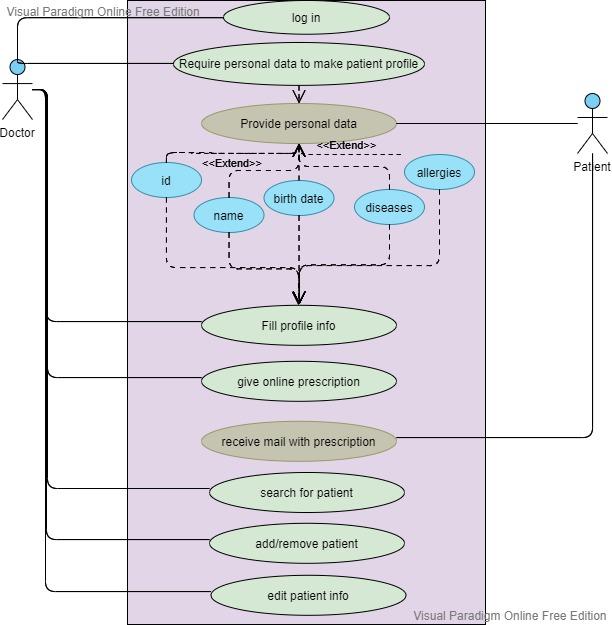
Тази документация предоставя общ преглед и обяснява архитектурата на системата “Електронно здравно досие”. Документацията определя основни сценарии (use-case) и актьори (actors), логическия изглед на архитектурата, изглед на процесите, изглед на данните, изглед на внедряването. Предоставя обосновка за архитектурните решения и описва различни нефункционални изисквания. Разглежданата архитектура е клиент-сървър.

Актьорите са:

* Лекар, който води здравните досиета и работи с приложението
* Пациент, който може да получи електронни рецепти и направления, благодарение на системата

# Use-Case View

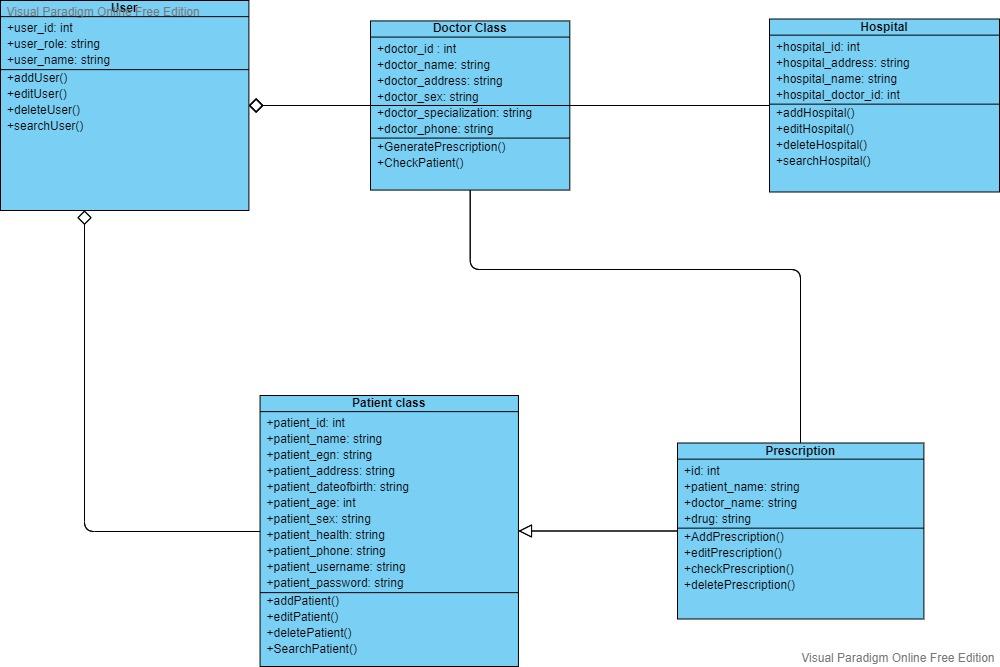
Use-Case View е предназначен за всички заинтересовани страни в системата, включително крайните потребители. Той описва набора от сценарии и / или случаи на употреба, които представляват някаква значима, централна функционалност на системата. Описвайки действащите лица и случаи на използване на системата, този изглед представя нуждите на потребителя и системата и се разглежда в детайли в следващото ниво на проектиране на системата.



Тази диаграма представя функционалните възможности на софтуера. Лекарят може да влезе в свой собствен профил, където се държи цялата информация за неговите пациенти. Когато дойде нов пациент, лекарят добавя нов профил за пациента, като попълва с необходимите предоставени данни. При всеки преглед лекарят може да редактира тези данни и да добавя бележки за състоянието на пациента. В случай на необходимост, той може да издаде онлайн рецепта/направление, която се изпраща по мейл до пациента. Така се спестява ходенето до лекарския кабинет, когато това е допустимо. Лекарят има възможност да използва търсачка, за да намира бързо и лесно даден пациент. Ако пациент се премести при друг лекар, може да бъде изтрит.

# Logical View

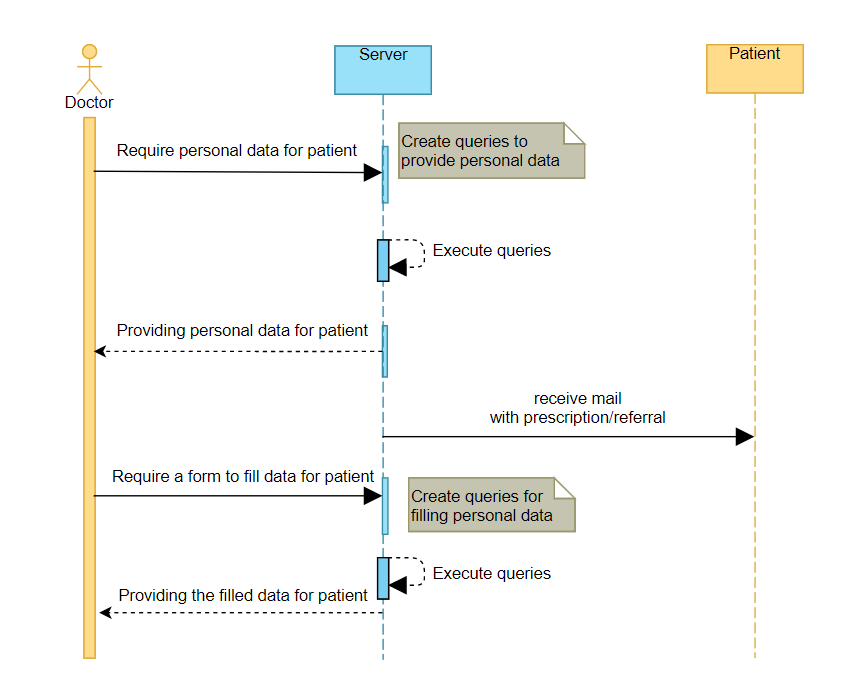
Логическият изглед се отнася до функционалността, която системата предоставя на крайните потребители. Основната цел на логическия изглед е да се дефинират компонентите, които изграждат системата и да се определят връзките, чрез които те ще комуникират и взаимодействат помежду си. Главният фактор за вземане на решения зад дефинирането на системните компоненти е необходимостта да се изолират компонентите, които е вероятно да се променят от останалата част на системата. Чрез ясно дефиниране на връзките на тези компоненти и скриване на техните вътрешни реализации от останалата част на системата, въздействието на очакваните промени може да бъде сведено до минимум.



Класът User управлява всички операции на потребителите на системата.Той съдържа основното поведение на потребителите(актьорите)и техните атрибути.Doctor класа управлява всички операции свързани с издаването рецепти на пациентите.Patient класа управлява операциите свързани с регистрирането на нов пациент в системата .Hospital класа – свързан с регистрирането на болница в системата.Prescription класа – включва процесите свързани с рецептите, издадени от лекарите.Класовете са свързани с подходящи връзки.

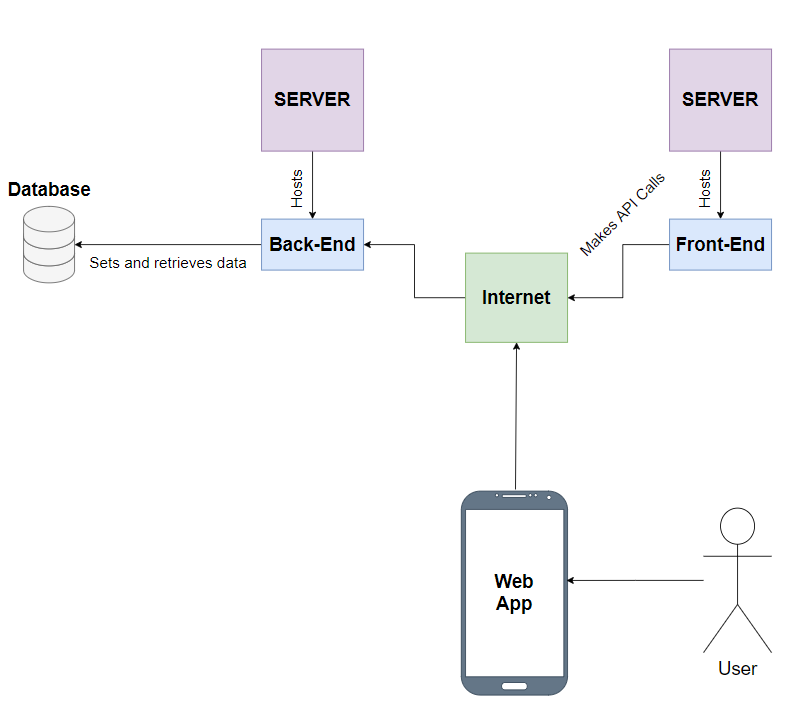
# Process View

Изгледът на процесите представя динамичните аспекти на системата, обяснява системните процеси и начина, по който те комуникират, и се фокусира върху поведението на системата по времето на нейното изпълнение.

Sequence диаграмата представя „поглед“ отгоре на последователността и времето, по което се изпълняват заявките към системата и съответно техните отговори. Основната комуникация се осъществява между доктор и сървър, а комуникацията с пациент, представлява получаване на информация под формата на мейл, съдържащ необходимите данни.

# Deployment View

Изгледът на внедряване изобразява системата от гледна точка на системен инженер. Той се отнася до топологията на софтуерните компоненти на физическия слой, както и физическите връзки между тези компоненти. Описва разполагането на софтуерите компоненти върху хардуера и показва разпределените аспекти на системата.

****

# Data View

Изгледът на данни представя модел на обработваната информация от базата данни, интегрирана в приложението с ролята на хранилище. Моделът изобразява таблиците на индивидуалните обекти, типовете данни във всяка една от тях и начинът по който те са логически свързани помежду си (релации). В този случай е представен начинът, по който са обвързани болниците, лекарите, пациентите и рецептите са обвързани. Между лекаря и болницата е създадена **many to one** релация, което означава, че към една болница могат да бъдат регистрирани множество лекари, но един лекар може да се регистрира само към една болница. Подобна е връзката между лекаря и пациента. По друг начин обаче седят нещата при връзката между рецептата и пациента. Тук е създадена **many to many** релация, тъй като на един пациент може да му бъдат предписани множество рецепти и една рецепта може да бъде предписана на множество пациенти. Подобен е случая и при лекарите, издаващи рецептите.

# 

# Нефункционални изисквания

* **Достъпност** – Приложението бива достъпно лесно чрез браузър, като потребителят (лекарят) трябва да влезе в своя профил
* **Разширяемост** – Създаденият софтуер позволява бъдещ растеж, могат да бъдат предвидени и създадени подобрения, без да се нарушават съществуващите системни функции. Могат да бъдат добавени нови функционалности, като вид администратори, които да използват софтуера, като те ще притежават различни роли и права за използване.
* **Използваемост** - Интерфейсът, който ще предоставя възможността на потребителя да комуникира със софтуера, е под формата на използваното устройство. Тъй като софтуерното решение ще представлява уеб приложение, то може да бъде достъпно от изключително голям брой различни устройства: настолни компютри, лаптопи, таблети, телефони и други. Множеството различни видове устройства позволява гъвкавост в използваемостта на софтуера - при налична интернет връзка може да бъде достъпен където и когато желае потребителя.
* **Сигурност** - Много важно изискване за една архитектура е сигурността.Като тя е свързана със степента на защита на данните от неправомерен достъп до тях, осигуряване на идентификация на потребители.Нивото на защита на данните на потребителите е високо, защото системата притежава всички необходими процеси за сигурност - Идентификация(Identification), Автентикация(Authentication), Оторизация(Authorization), Проверка(Audit) и Криптиране (Encryption).
* **Производителност** - Производителността определя колко бързо дадена софтуерна система или нейна част реагира на действията на определени потребители при определено работно натоварване. В повечето случаи този показател обяснява колко време потребителят трябва да изчака, преди да се случи целевата операция (страницата се изобразява, транзакцията се обработва и т.н.), като се има предвид общият брой потребители в момента. Изискванията за производителност могат да описват фонови процеси, невидими за потребителите, напр. архивиране. С цел нейното ускоряване, използваме server-client архитектурата, която позволява трудоемките процеси да бъдат изпълнени на сървър, а простите на потребителската машина.
* **Интероперабилност** - Оперативната съвместимост е способността за обмен на информация и комуникация с вътрешни и външни приложения и системи. Форматите на данни, транспортните протоколи и интерфейсите са ключовите атрибути за архитектурата на оперативно съвместими системи. Стандартизирането на форматите на данни, транспортните протоколи и интерфейсите са ключов аспект с цел нейното успешно проектиране. Използваните от нас REST архитектура и HTTP протокол предлагат точно тази услуга.