

**Курсова работа**

**по**

**Програмни Среди**

**Изпълнена от:**

**Студент:**

**Даниел Благоев Богоев**

**Факултет:**

**ФКСТ**

**Специалност**:

**ИТИ**

**Група:**

**53**

**Факултетен** **№:**

**501216015**

**Тема на Проекта**

Личен медицински картон. Да се проектира и реализира програма, която да следи здравното състояние на потребителя. Да се предостави възможност за отбелязване на историята на посещенията при лекари и специалисти. Проверка за необходимост и напомняне за профилактични прегледи. Справки за честотата на изкараните заболявания и предстоящи прегледи.

**Детайлно задание**

За реализирането на програма, която да следи здравното състояние на

пациенти ще се реализира както Server, така и Client, постигайки Client–server model. Това осугирява сигурност / интегритет на данните съхранявани в

централен сървър. Client ще комуникира със Server чрез REST API като идентифицирането ще стане посредством authentication имплементация (Basic / OAuth). Данните ще се съхраняват на сървъра / backend в база от данни (MySQL database).

Разширявайки заданието на курсовия проект - приложението ще представлява

регистър за множество от болници, тоест за един пациент ще се показва

историята от посещение във всички болници, а не от една.

Client ще представлява десктоп приложение, чрез което потребителя ще може да се впише в системата - доктор или пациент. Като в зависимост от вписаната

роля, потребителя ще има различни възможности за работа с програмата.

Докторът ще може да вписва посещения/диагнози **към определена болница за определен пациент**. Той има цялостен контрол върху данните свързани с посещения при   
него, но не и контрол върху данни за други доктори.

Пациентът ще може единствено да преглежда своите данни - посещения /

диагнози / бъдещи задължителни посещения / резервации. Той ще вижда

данните структурирано - за всяка болница, доктор към болницата и данни за

посещението.

Администраторът ще може да създава болници към приложението, а болницата ще може да вписва доктори към нея. По този начин е възможно

цялостния контрол да става през Client приложението. Възможно е и един доктор да е вписан към няколко болници.

Backend / Server ще представлява web application, който ще изложи REST API за използване с цел реализиране на Client описан по-горе както и база от данни за съхраняване на данните.

Чрез REST API ще може да се:

1. Установява автентичност (login)
2. Създаде нов потребител от доктор
3. Регистриране на нова болница
4. Вписване на нов доктор към болница
5. Извличане на конкретна информация за пациента - болести / диагнози / посещения и т.н.
6. Извличане на конкретна информация принадлежаща на доктора - неговите пациенти спрямо всяка болница, в която е вписан
7. Извличане на конкретна информация за болницата - доктори / пациенти
8. Вписване на информация от доктора за пациента:
   * посещение

* диагноза
  + резервация

Всяка от гореизброените възможности ще бъде достъпна само в зависимост от

предоставените credentials - тоест докторът ще може да прави промени свързани с пациентите –

резервации, посещения, но няма да може да вижда информацията на пациента

към други доктори или болници,а пациента няма да може да прави подобни

промени или дори да вижда подробна информация за доктора. Тоест

scope-a на REST API възможностите е в зависимост от логнатия потребител.

1. **Функционалности**

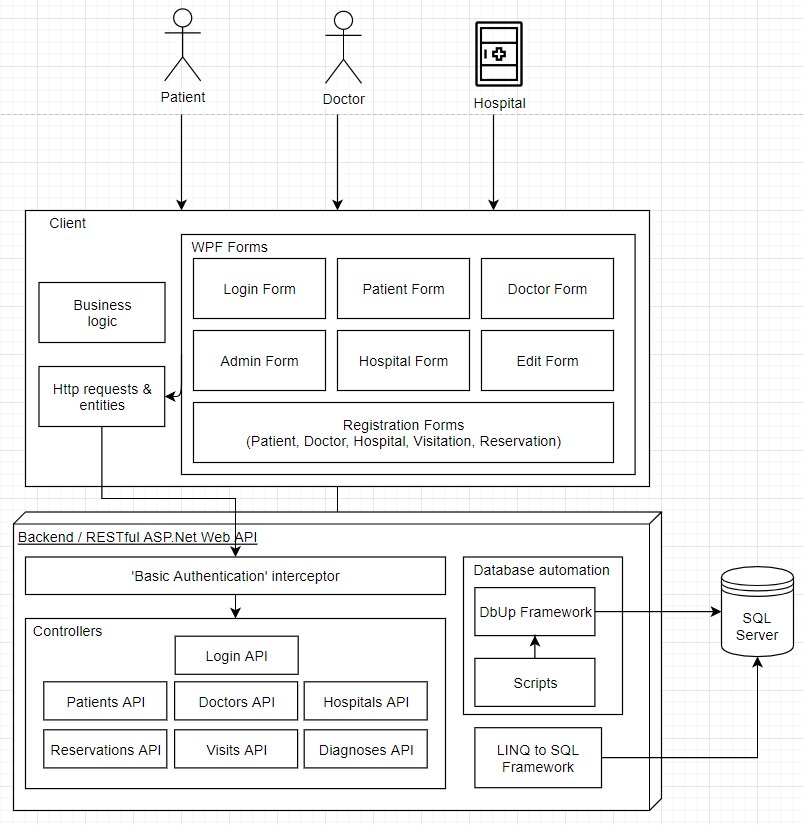
Остановените функционалности по време на разработването на приложението са следните:

* Сигурност (Authentication) при ‘login’ на всеки един от потребителите независимо от тяхната роля
  1. Пациент
* Възможност за извличането на собствена лична информация (имена, егн, адрес)
* Способност за обновяване/промяна на личните данни
* Извличане на информация за собствени резервации
* Извличане на информация за собствени диагнози
* Извличане на информация за предишни посещения
* Преглед на докторите в системата и техните специализации с цел посещение  
  1. Доктор
* Възможност за извличане на собствена лична информация (имена, егн, специализация и т.н.)
* Възможност за обновяване/промяна на личните данните
* Разглеждане на история на посещения от пациенти при него
* Разглеждане и създаване на резервация за конкретен пациент за конкретна дата и час
* Разглеждане и създаване на диагноза за конкретен пациент
* Възможност за обновяване/промяна на данните към посещения/резервации/диагнози
* Регистриране на нов пациент в системата, в случай на посещение от нерегистриран човек  
  1. Болница
* Преглед на лична информация на болницата (име, локация и т.н.)
* Възможност за обновяване/промяна на данните към болницата
* Възможност за преглед на всички доктори работещи в болницата
* Възможност за преглед на всички посещения на доктори работещи в болницата
* Възможност за преглед на всички резервации към доктори работещи в болницата
* Възможност за регистрация на нов доктор като работещ в болницата  
  1. Администратор
* Ексклузивни права за преглед, промяна на пълната информация в системата, включително и регистриране на нова болница, доктор, пациент. Има достъп до всяка от функционалностите описани за другите потребители през REST API.

1. **Архитектура**

Гореизброените функционалности са постигнати чрез Client – Server архитектура.

Клиента (UI) представлява WPF application, който си комуникира със сървъра – backend RESTful API.

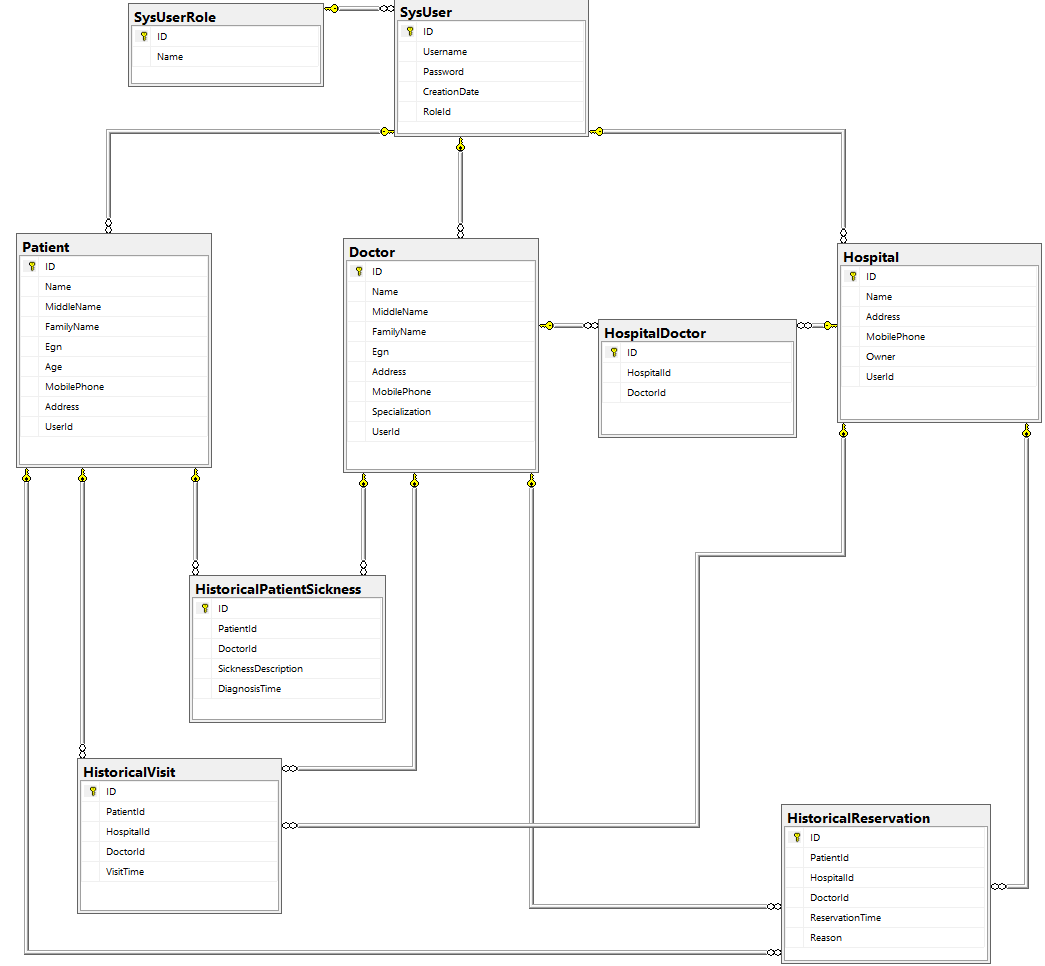


1. **Използвани технологии**

За реализацията на UI – е използвана библиотеката Design Materials. Тази библиотека позволя по-стилно предстване на UI – елементите в ХAML конструкцията на WPF формите, които допринасят за по-добър user experience на приложението.

1. **База данни**

ER схема:



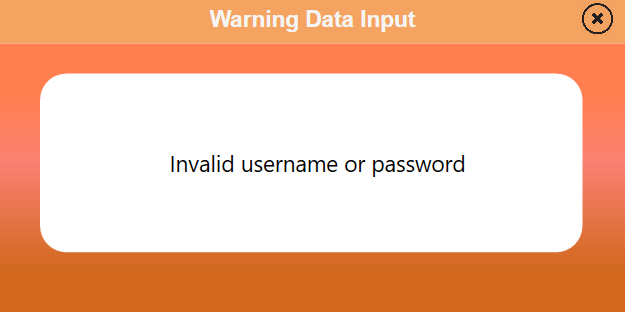
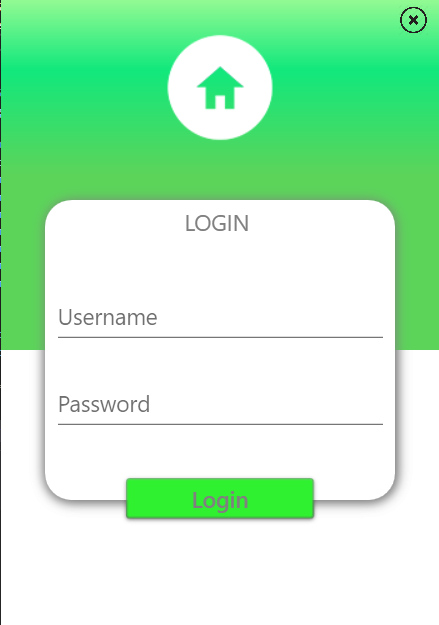
1. **Имплемeнтация**

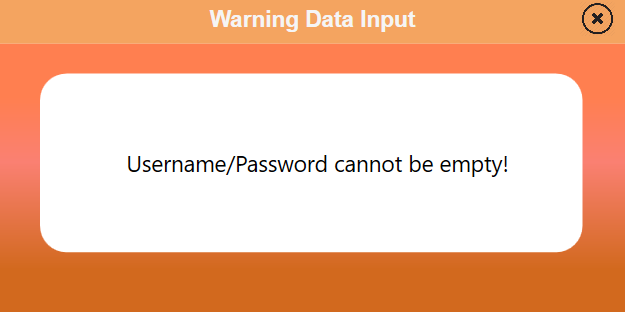
Приложението има няколко основни форми, защото е базирано на роли - Admin, Patient, Doctor и Hospital. Още с влизането на user-a се ауторизира дали той е админ, пациент, доктор или болница.

Всяка от главните форми показва различни информация в зависимост от типа роля, която има вече логналият се потребител.

Login формата изглежда по следният начин:

Има направена проверка дали потребителят е попълнил полетата Username/Password. Ако не ги е попълнил се изпизва съобщение за грешка, което генерирано чрез форма, която е проектирана за визуализацията на съобщения(информационни, съобщения за грешка и т.н.) . Когато потребителят е въвел грешно потребителско име или парола също се извежда съобщение за грешка. След затварянето му и въвеждане на коректни данни за идетификация на потребителя програмата отваря една от главните форми, в зависимост от ролята на потребителя.

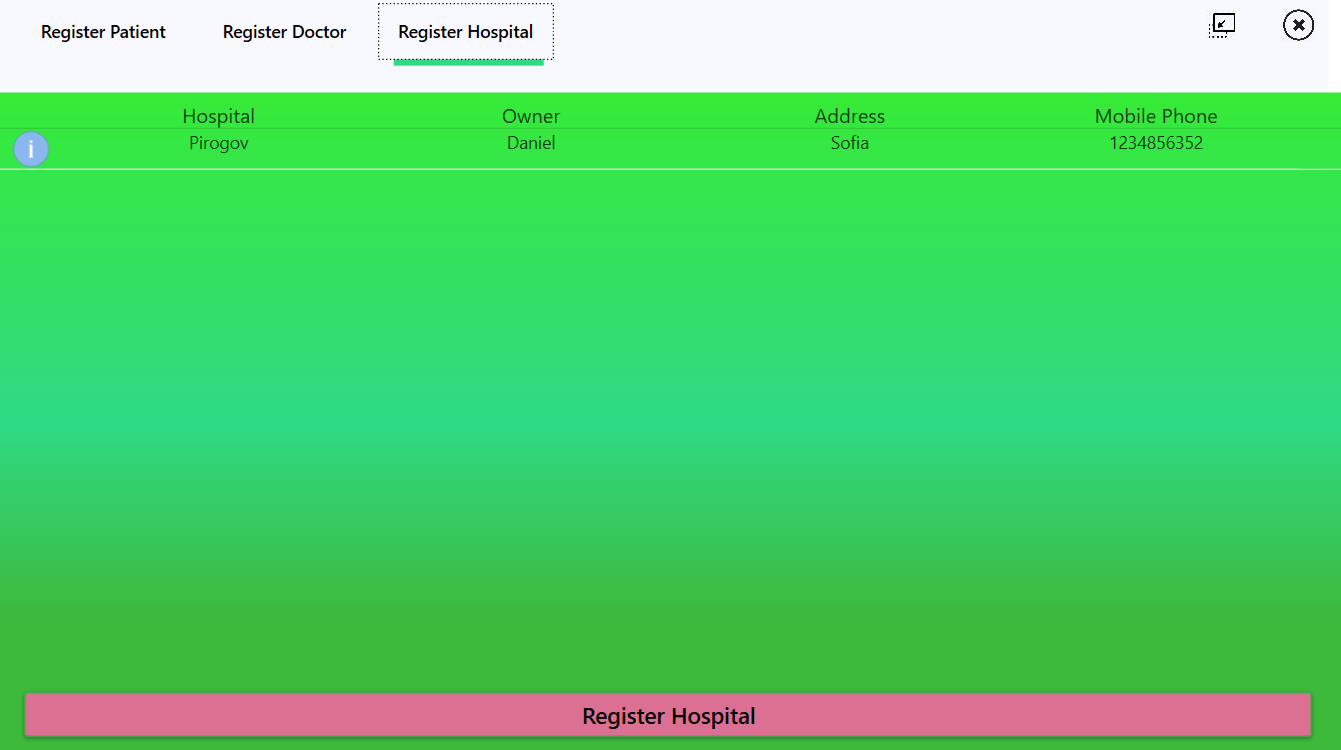


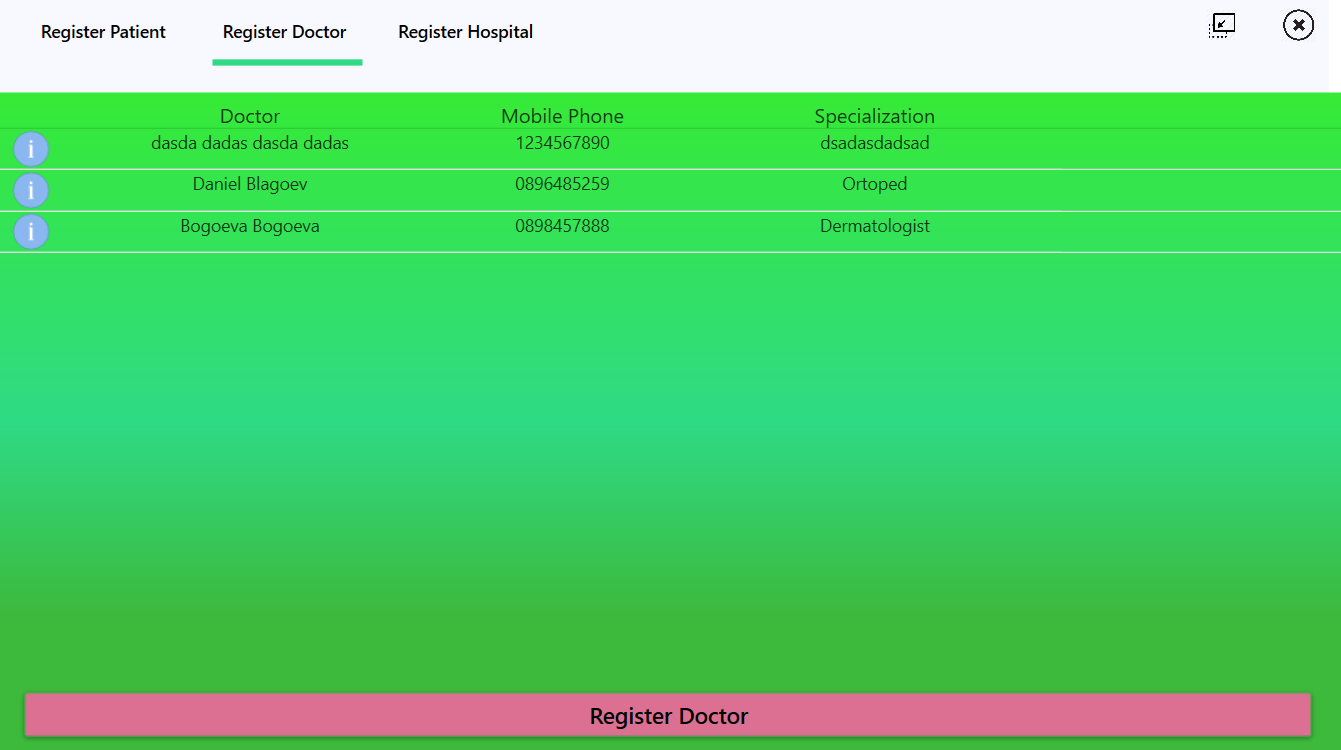


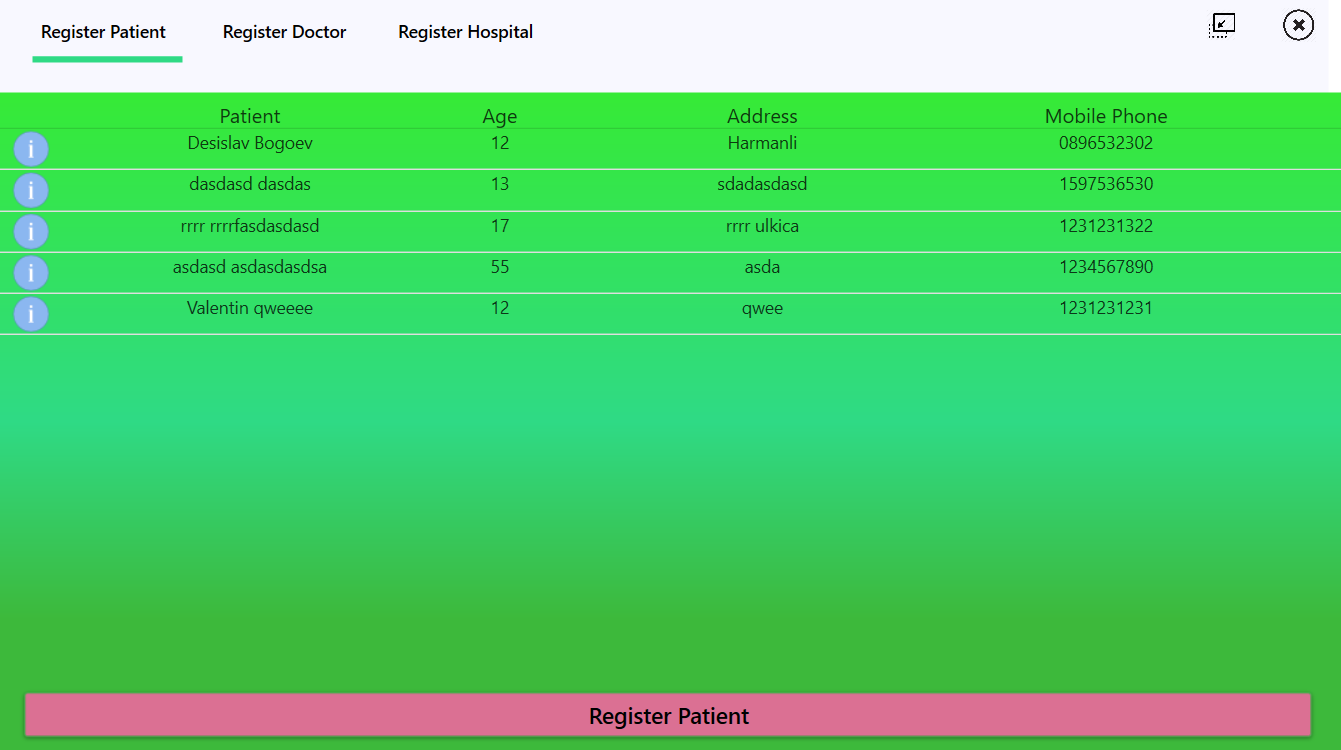
Ауторизацията и верификацията на потребителят, който иска да влезе в системата, като се подават Username-а и Password-a, като се кодират с Base64 в следният формат (Username:Password) и се прави POST HTTP Request, на който му се подават BasicAuthorization – Username:Password (Base64 Еncoded). Ако резултатът е успешен се връща като json – Guid-то нa потребителя и ролята в базата от данни, които се използват за следващите извиквания на главната форма и подформите, които ще са свързани с тях.

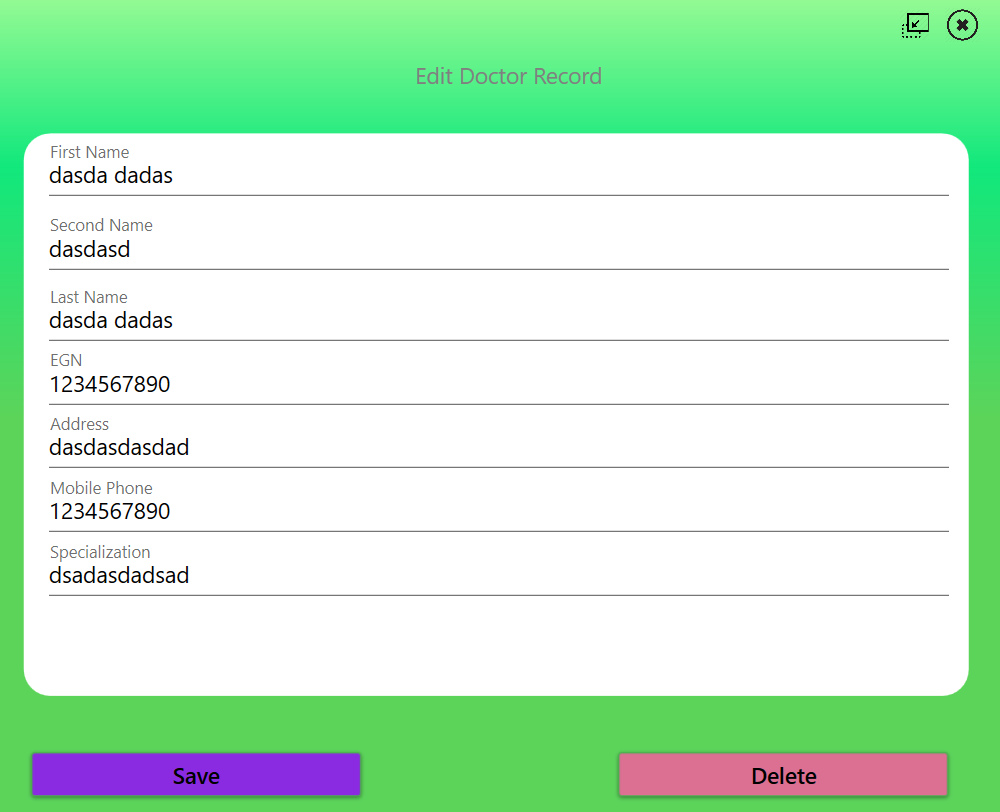
**Admin Form**

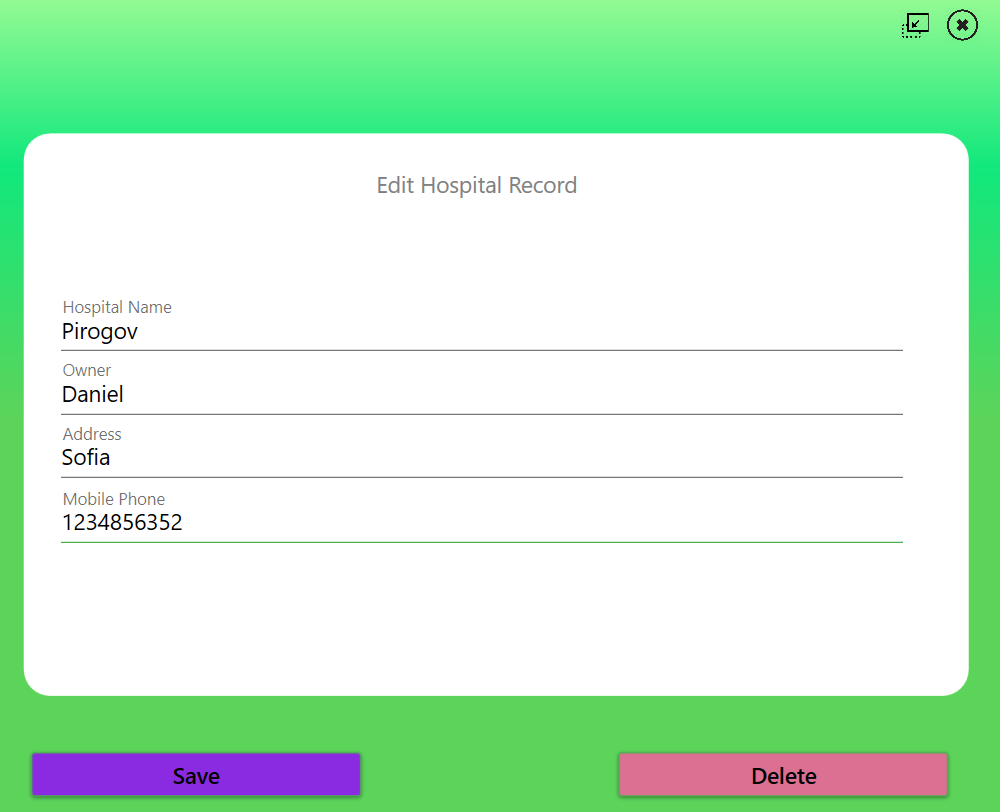
Тя има три таба – Register Patient, Register Doctor, Register Hospital. Потребителите с Admin роля, могат да виждат всички пациенти, доктори и болници. Те могат да променят коригират или да изтриват информацията на за всеки един от горе изброените потребители ако се наложи. Във всеки от табовете има списък съответно с всички пациенти, доктори или болници. За да се визуализира на екрана пълната информация за всеки от потребителиете(Patient, Doctor, Hospital) администратора трябва да кликне върху Info бутона. Така, той ще може да види информацията за всеки един от потребителите, същевременно ще може да коригира неговите данни или да го изтрие. Визуализацията на всички данни става с друга форма, която е конструирана за едитване на потребителските данни. Визуализацията на списъците с пациенти, доктори и болници става чрез извикването на GET HTTP заявка. Всеки от листовете може бъде филтриран по колони в низходящ или възходящ ред.За модификация на всеки пациент, доктор или болница се прави PUT HTTP заявка, която ако не е успешна се вузуализира съобщение на екрана, което е генерирано с чрез специално проектирана форма за визуализация на съобщения, в противен случай се визуализира съобщение съобщаващо, че пациента, доктора или болцата са модифицирани успешно. Има опция, позволяваща на администратора да изтрие който и да е от потребителите. В този случай отново се отваря Edit формата, която позволя да се изтрие съответният потребител, като му се появява съобщение за потвръждение дали наистина е сигурен, че иска да изтрие този запис от базата данни. При потвръждаване се прави DELETE HTTP заявка, която изтрива записа от базата данни, където е използвано каскадно изтриване.

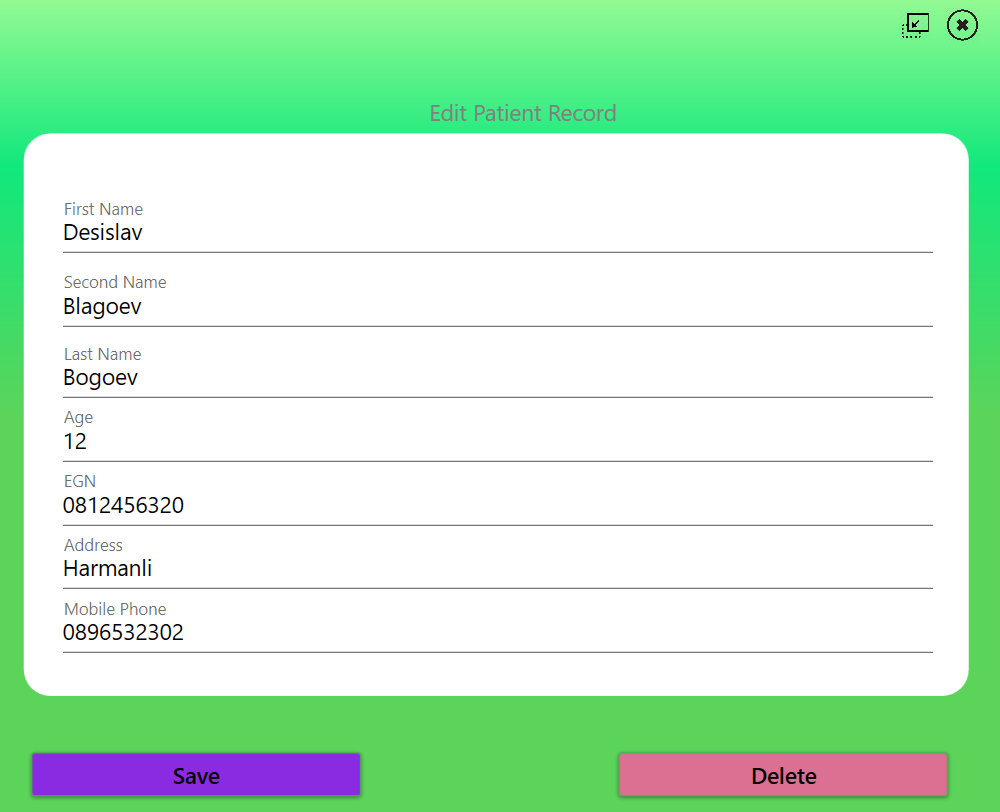


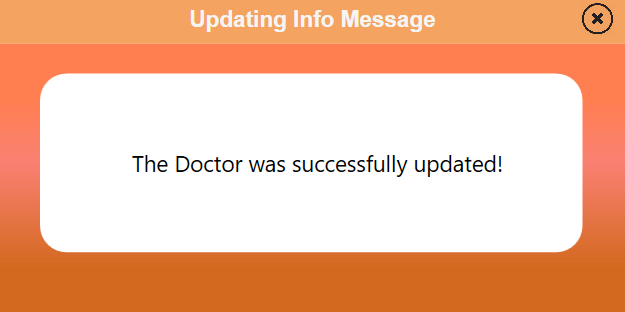
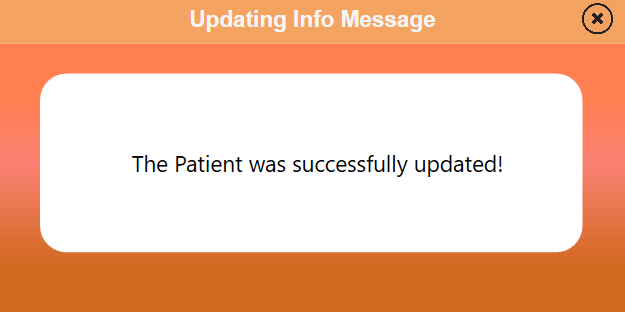


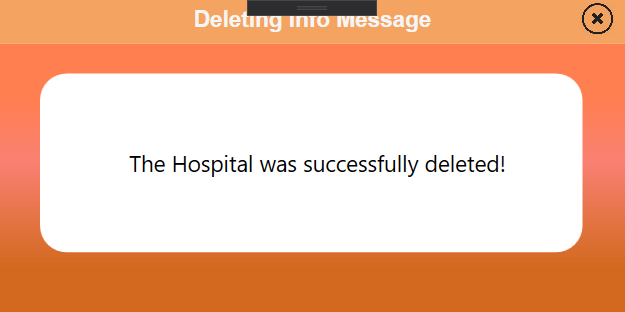
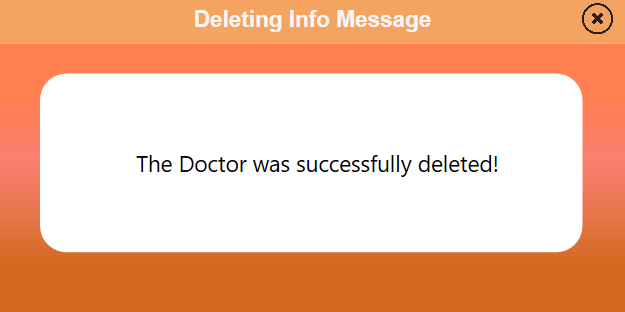


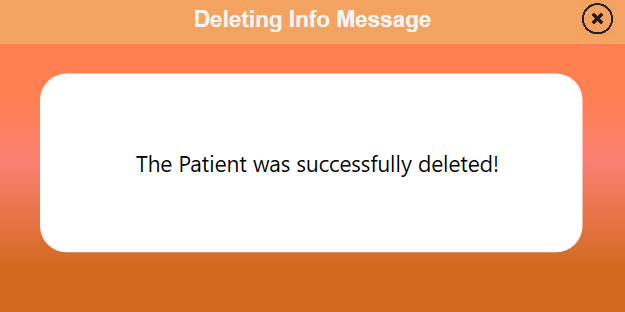




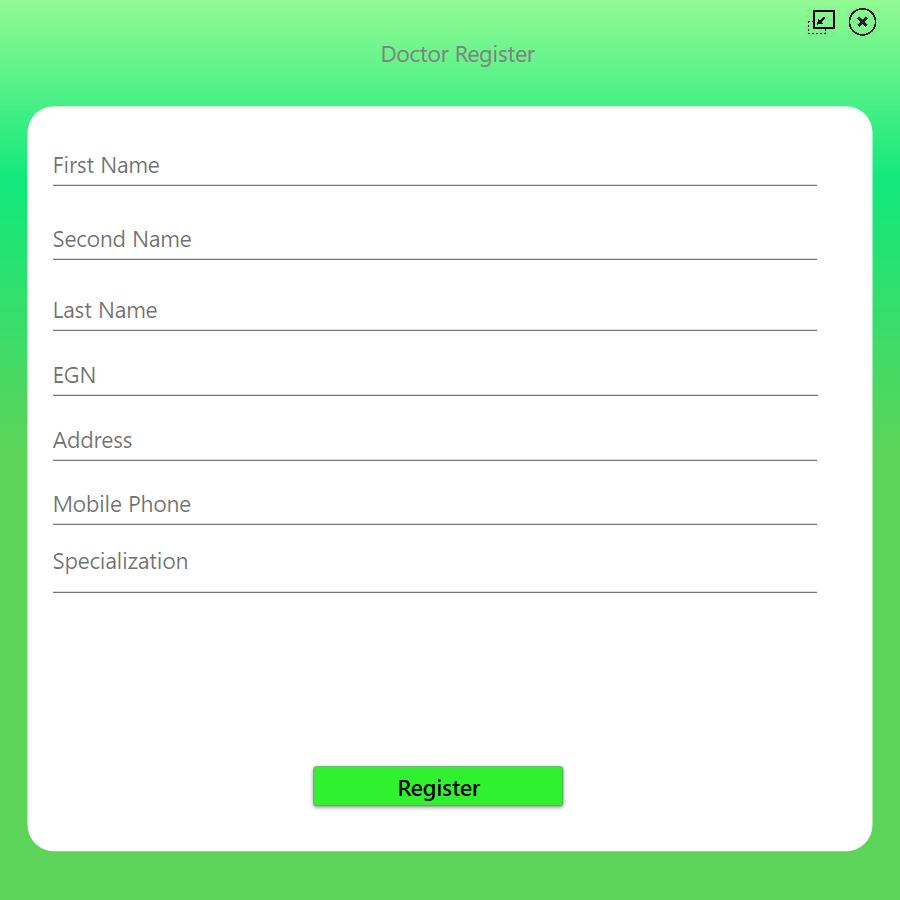


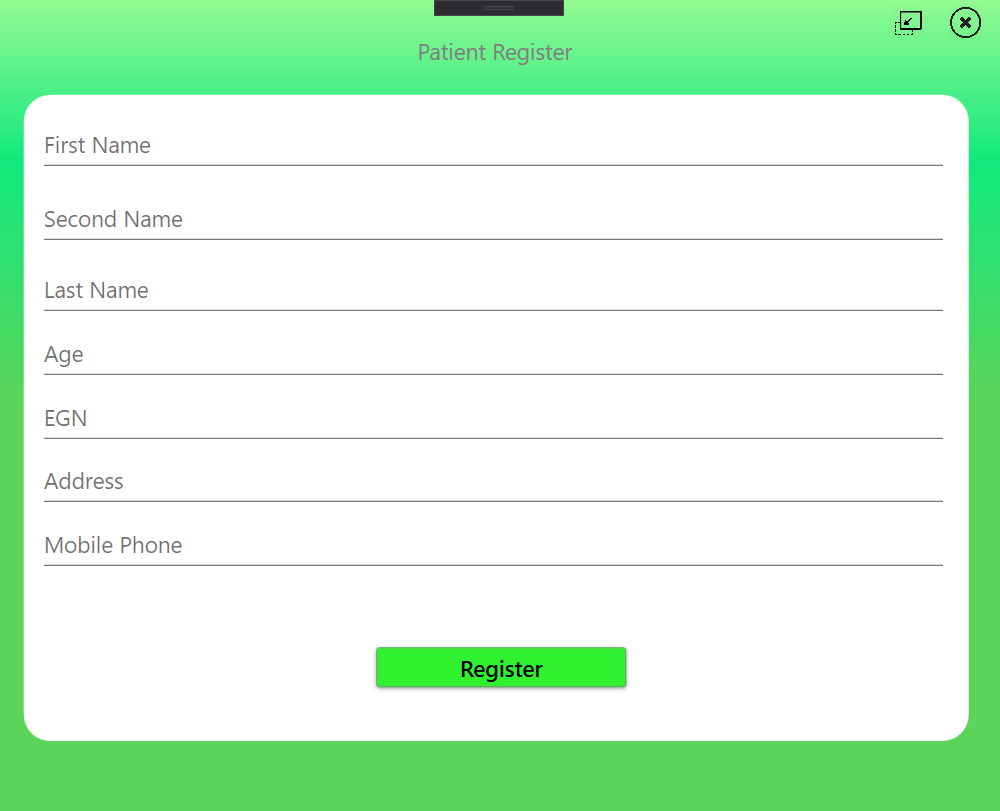


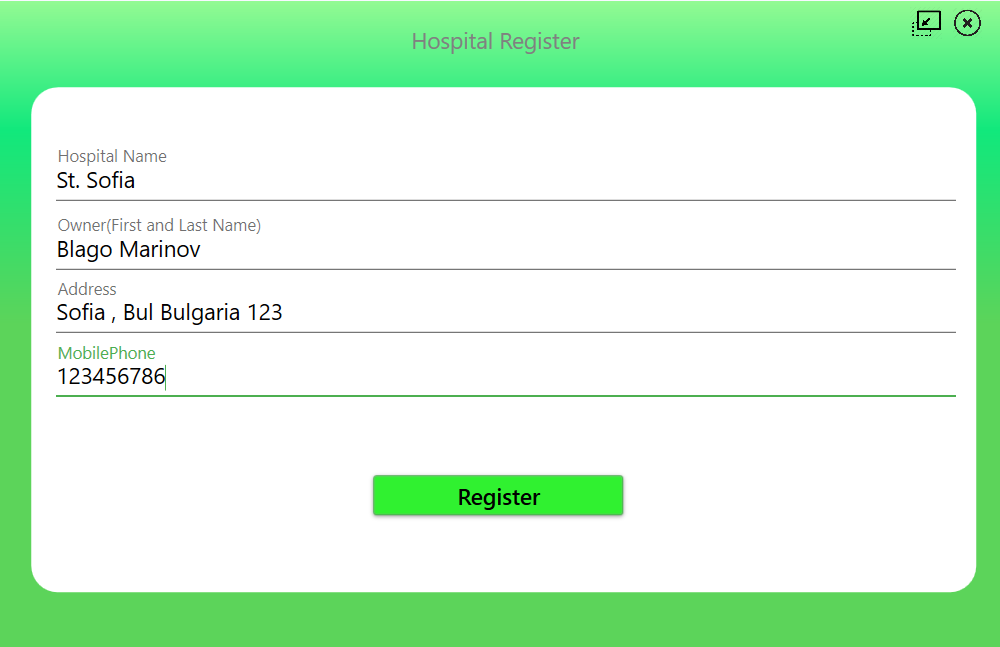


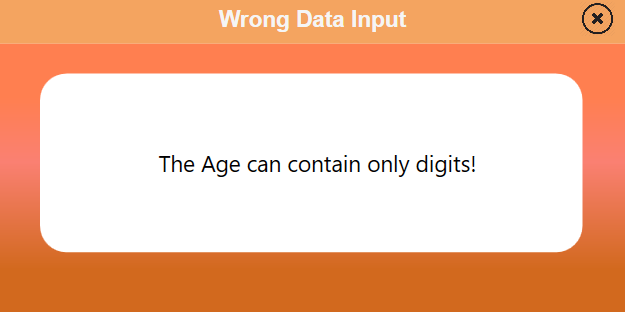
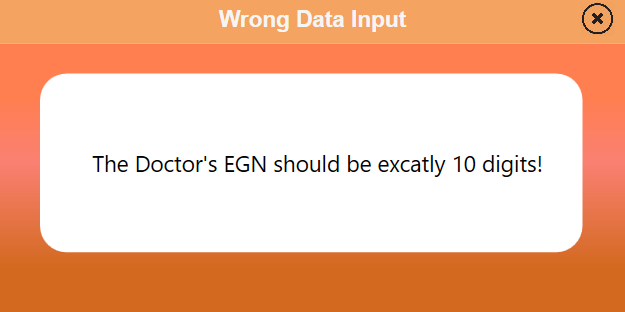
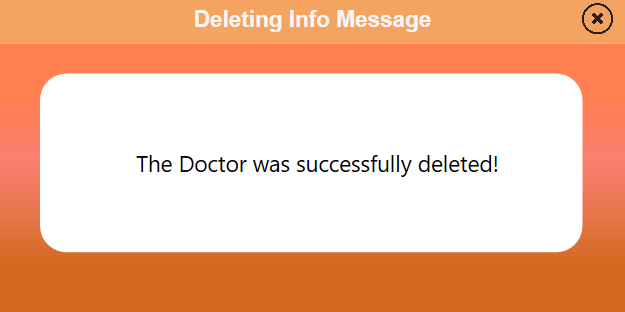
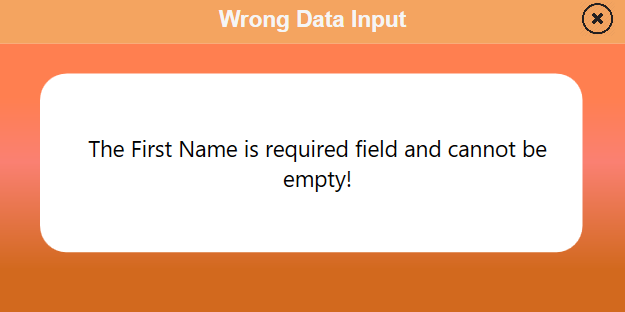
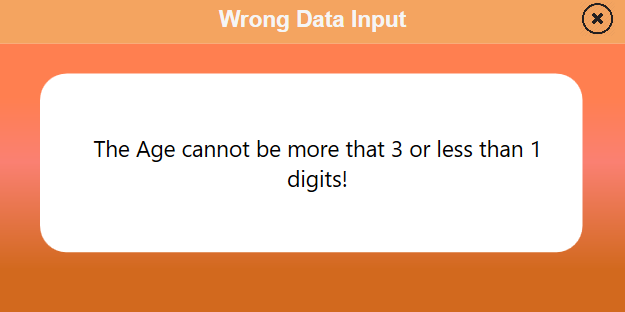
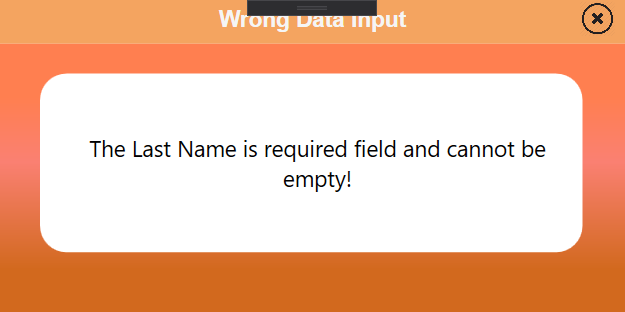


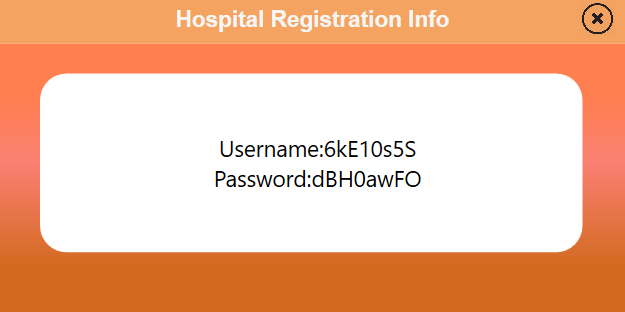
Всеки от табовете има бутон, който позволява на администратора да добави нов пациент, доктор или болница. Отваря се форма, която е проектирана, специално за регитрация съответно на пациент, доктор или болница. Има валидация на всички полета, които трябва да се попълнят – име, презиме, фамилия, ЕГН, години, адрес, мобилен телефон (за пациента) и съответно специализация за доктора, а за болницата – име на болницата, име на собственика на болницата, адрес и телефон на болницата. Когато някое от гореспоменатите параментри е празен или не са попълнени коректно данните се визуализира форма със съответната грешка, която дава пояснения ка потребителя какво и как трябва да бъде направено. При правилно попълнени данни съответно за всеки от гореспоменатите потребители се прави успешна регистрация както на потребител така и на пациент, доктор или болница в базата от данни. След като регистрацията е успешна на екрана се визуализира съобщение, със автоматично генерираните потребителко име и парла, които трябва да бъдат предоставени директно на пациента, доктора или болницата. Това се прави с цел защита на данните на всеки един от потребителите.

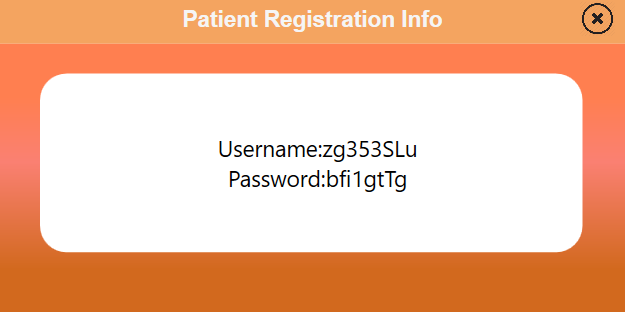


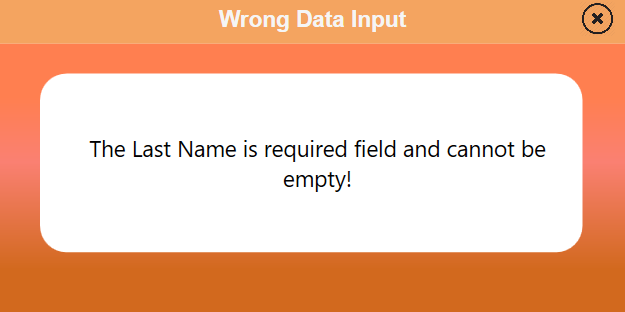
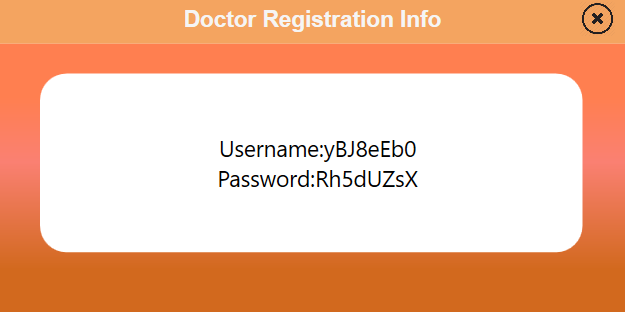


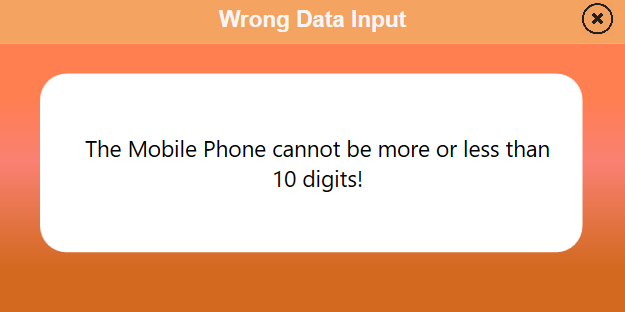








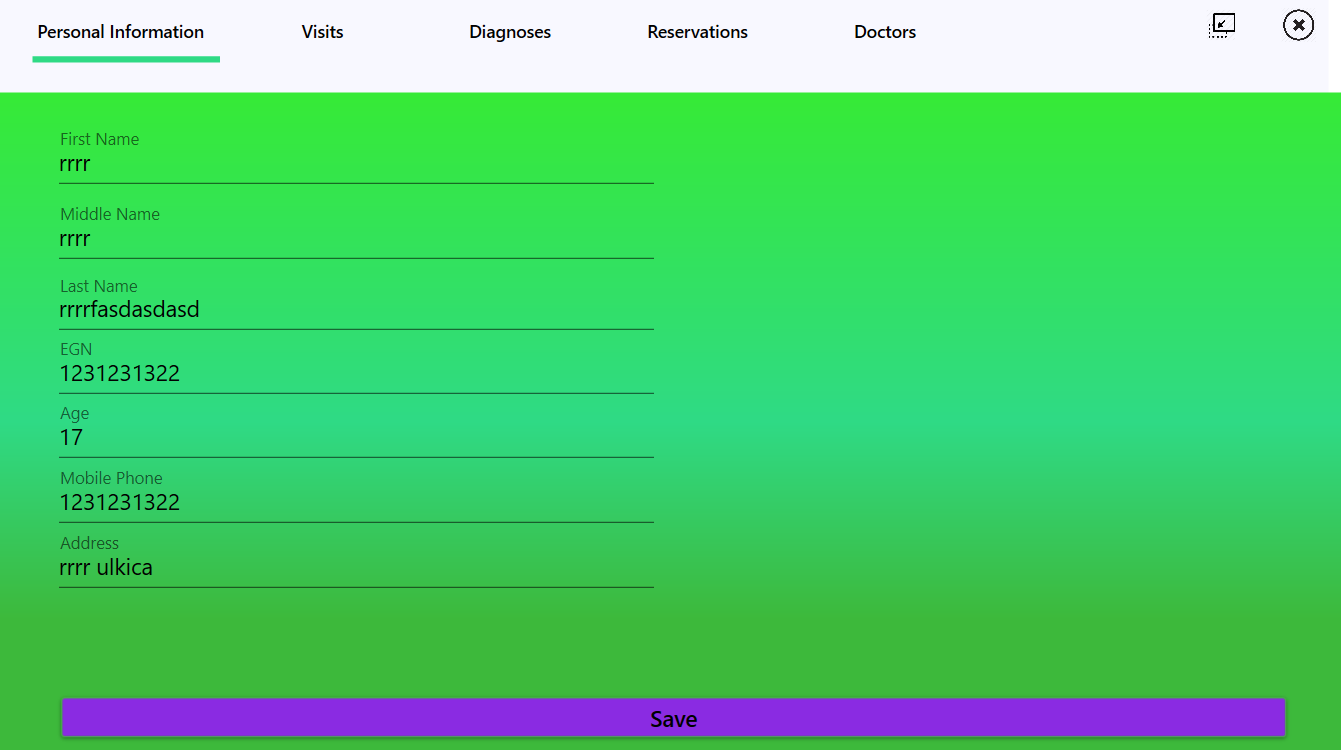


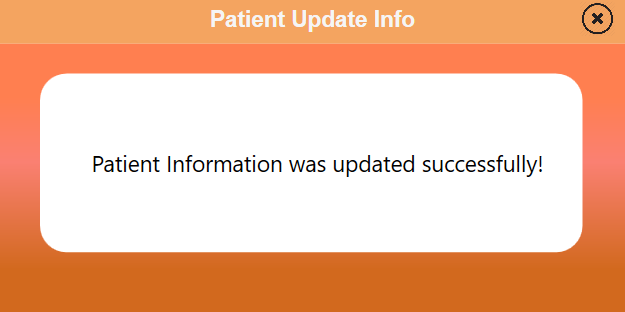


**Patient Form**

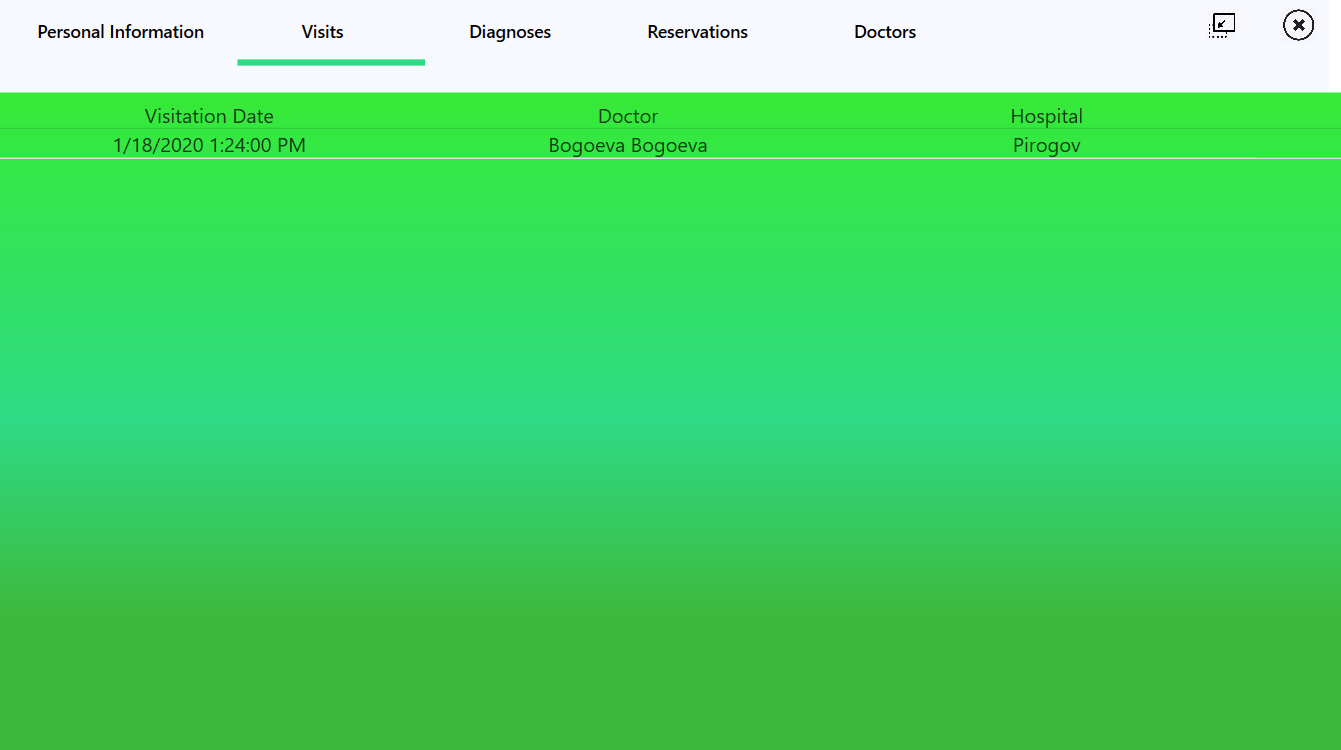
Формата за пациент се отваря когато се потребител с роля Patient се впише в системата. Има следните табове – лична информация за самият пациент, посещения при лекарят, диагнози, които е имал самият пациент, резервации, които са направени/оредени с различните лекари, и списъск със всички доктори в системата.

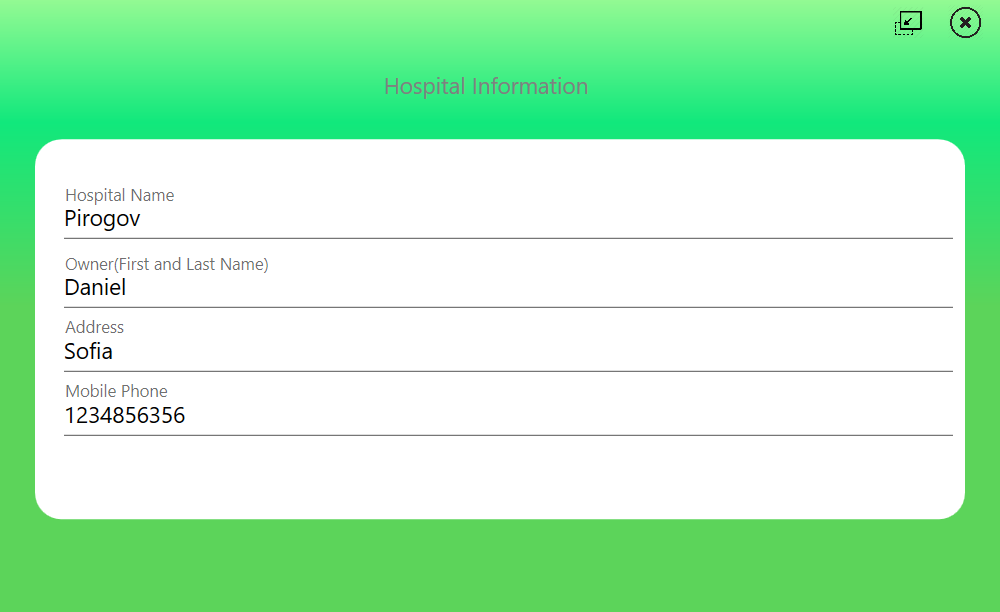
В първият таб пациентът може да коригира личните си данни и може да ги запази като натисне бутонът Save. При натискането на бутонът се прави POST HTTP заявка, която модифицира записът в базата данни за текущият логнал се потребител. Пациентът не може да промени своите username и password, зкоито са автоматично генерирани от системата. Могат да бъдат модифицирани само личните му данни с цел, че докторът е объркал някои от тях при въвеждането им в системата.





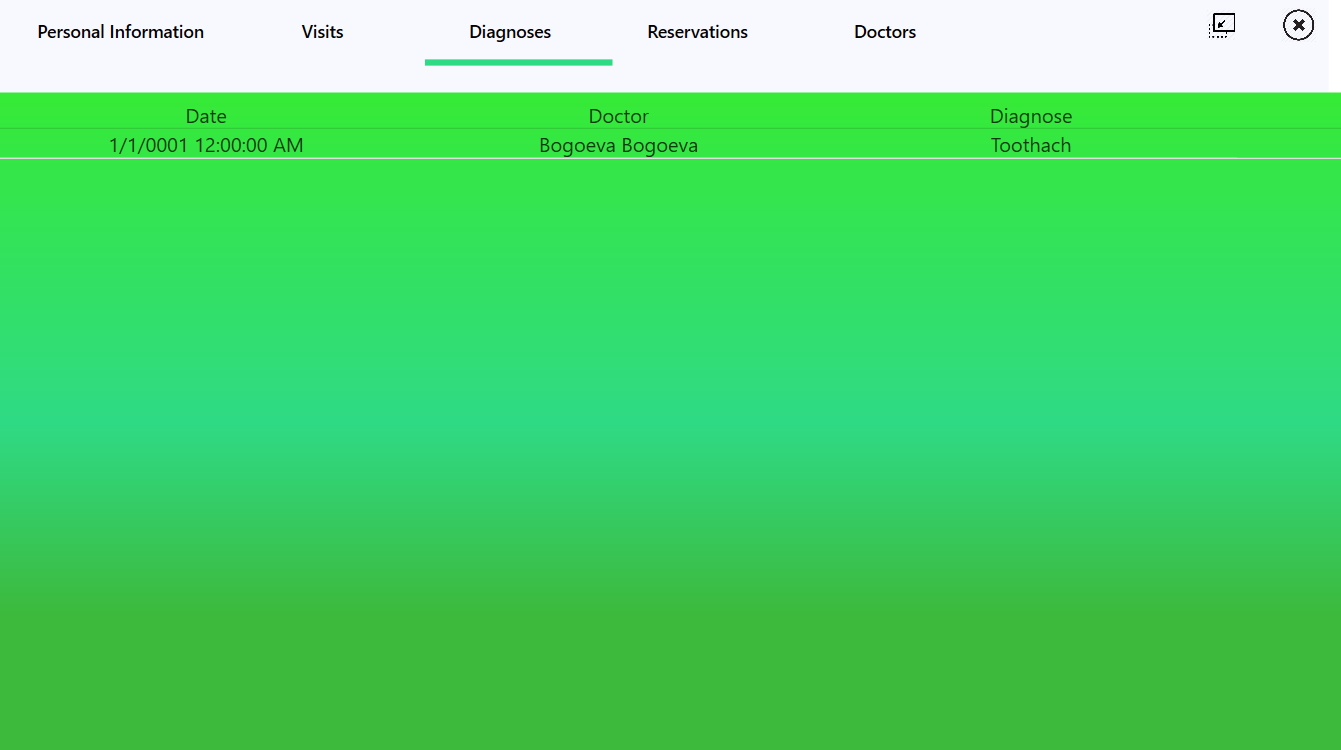
Във вторият таб пациентът може да види списък със всички посещения при всички доктори, при които е бил на посещение. В този таб се визуализират дата, часа, докторът и в болницата, в която се е извършил прегледът. При кликане на доктор или болница пациентът може да види по-подобробна информация за самият доктор – име, презиме, фамилия, телефон, адрес, специализация, а за болницата – име, собственик, адрес и телефон в случай , че иска да се свърже с болницата или докторът.



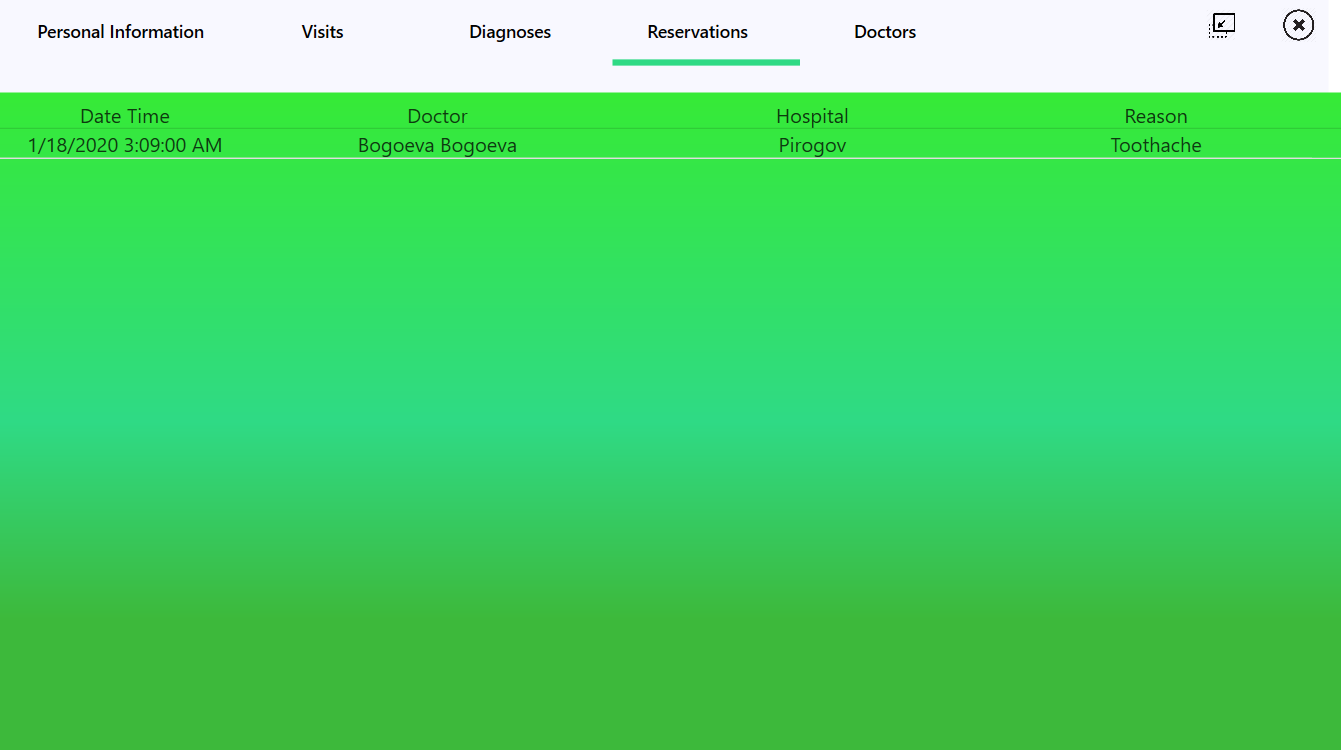




В третата секция се съдържа информация за всички диагнози, които са му били съставени. Сътветно в този таб са представени датата, часът, докторът, болницата, където се е извършило дигнозтицирането и съответно диагнозата, която му е била съставена. Тук също може да се види по-подробна информация за самият доктор и болница, като се кликне сътоветно върху определен доктор или болница.



В четвъртият таб могат да е визуализирана информцията, която показва вече направени резервации при различни доктори в различни доктори. Вече направените резервации показват часът, датата, докторът, болницата и причината, поради която отива пациентът при определеният доктор. Пациентът отново може да види по-подорбна информация за определен доктор или болница като кликне на името на болницата или докторът.



В петият таб се съдържа информация за всички налични доктори. Визуализирана е информацията за доктора – Име и Фамилия, специализация и мобилен телефон. Когато се кликне върху името на доктора се отваря форма, която визуализира по-детайлна информция за избраният лекар.

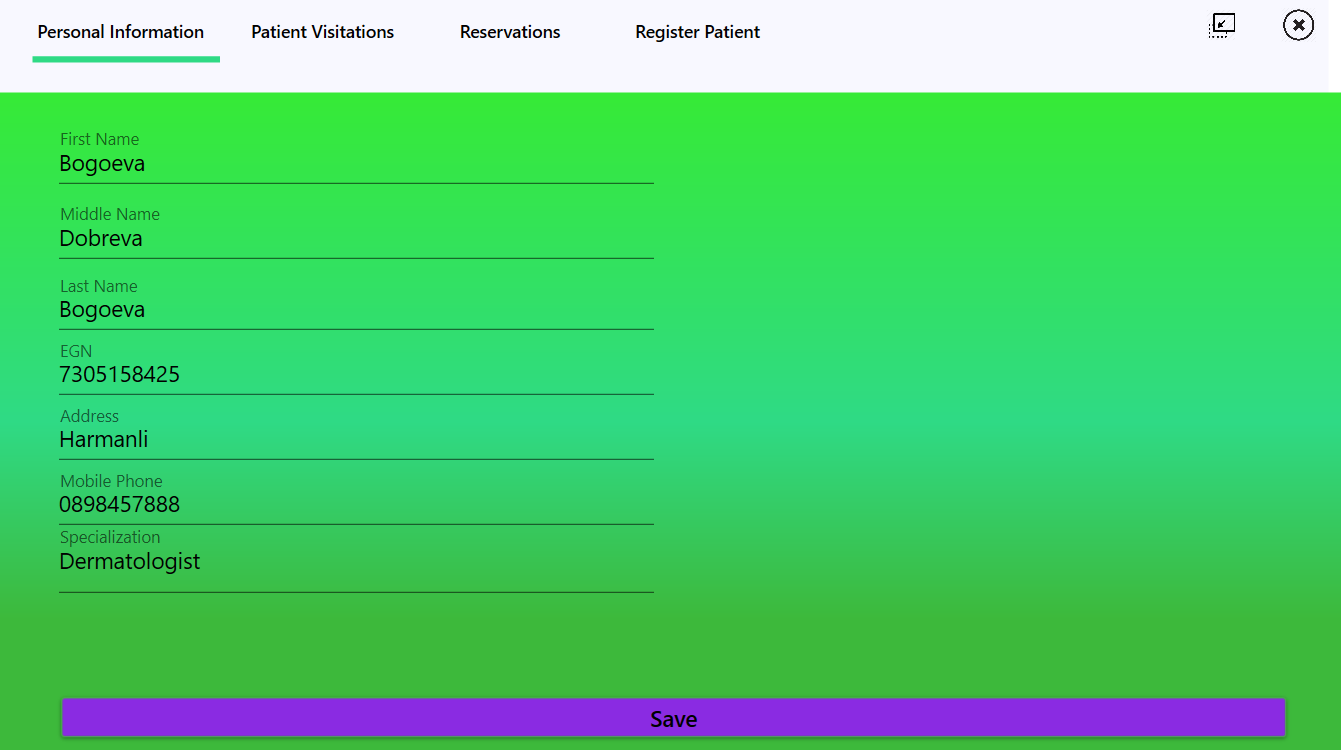
Всеки от гореспоменатите листовете може бъде филтриран по колони в низходящ или възходящ ред. Всички данни визуализирани в табовете се вземат чрез GET HTTP заявка сътовено за пациент, посещения, резервациии, диагнози и доктори.

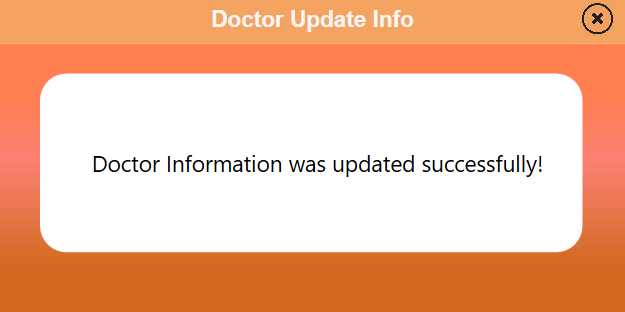


**Doctor Form**

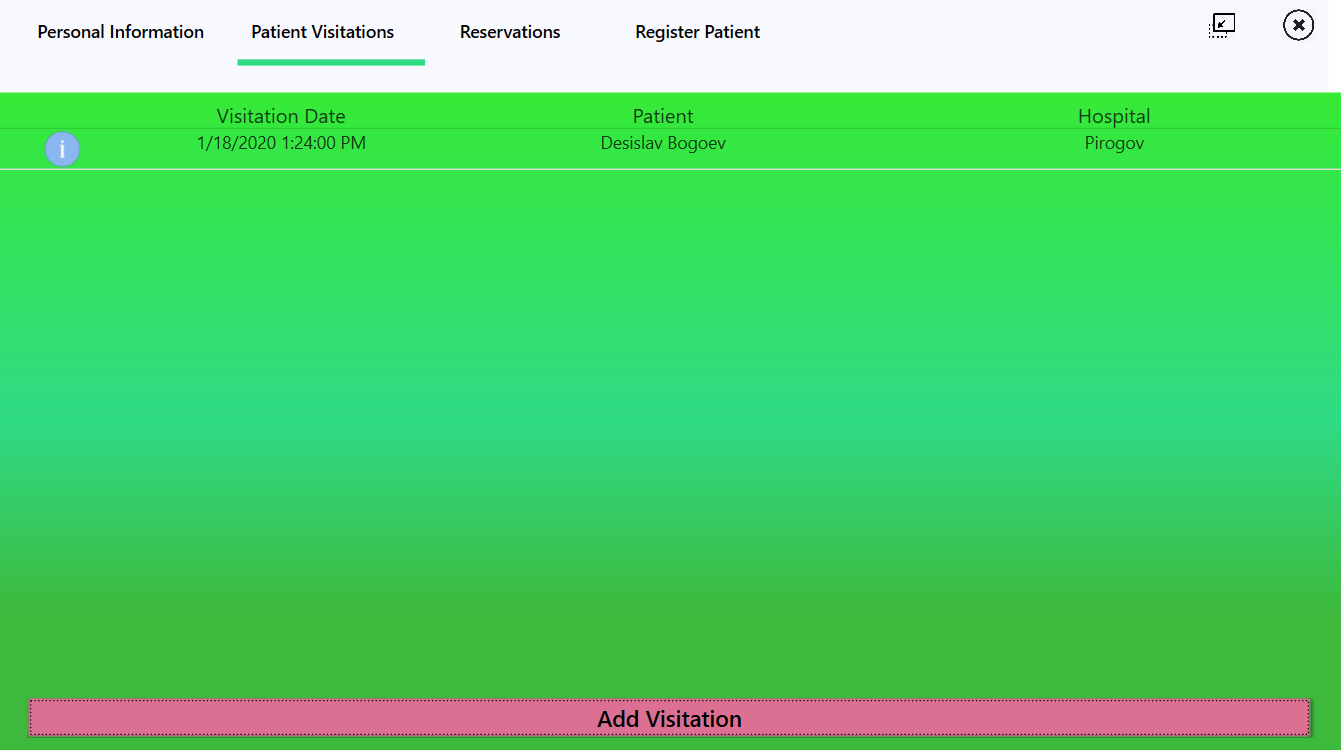
Тази форма се визуализира на екрана, когато потребител с роля Doctor, се впише в системата. Има следните табове – лична информация за самият доктор, посещения на различни пациенти, резервации, които са направени/оредени с различните пациенти, и регистрация на пациент.

В първият таб се визуализират личните данни на вече логналият се доктор в системата. Те се вземат чрез GET HTTP заявка, която връща информацията за докторът. Докторът има възможност да коригира своите данни. След като е готов с корекцията на данните и натисне бутонът Save се прави PUT HTTP заявка, която модифицира записът в базата данни. Има направена валидация за всяко от едно от полета, които може да коригира докторът. Ако е са въведени неправилни данни или някое от полетата е празно при опит за модификация се визувализира форма за съобщение , която дава индикация каква е грешката и какво трябва да се направи. При успешна модификация на записа се изпивсва съобщение, което информира потребителят, че промяната на данните е успешна. Това отново се постига, че формата , която е създадена специало за съобщения.

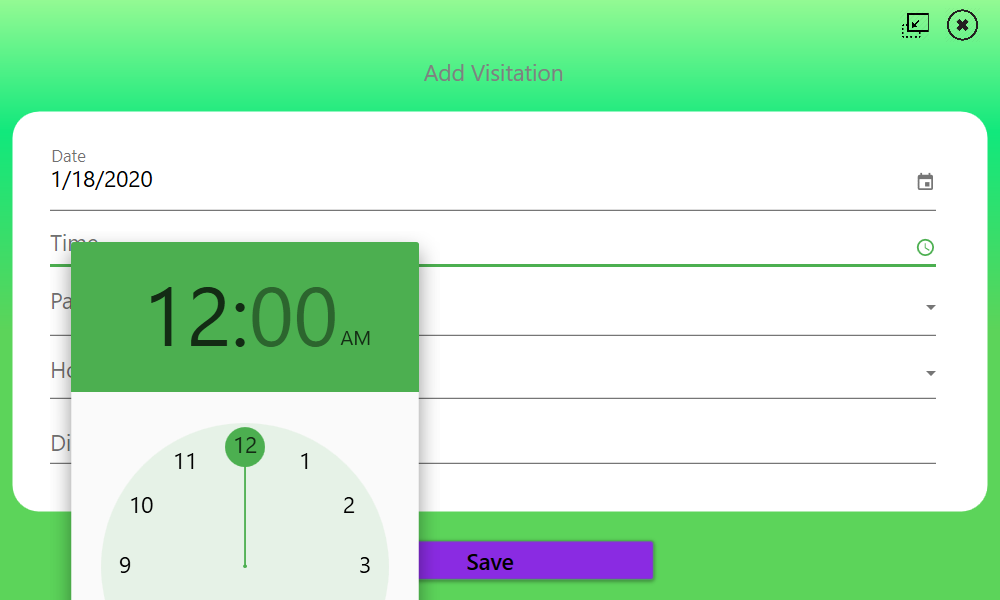


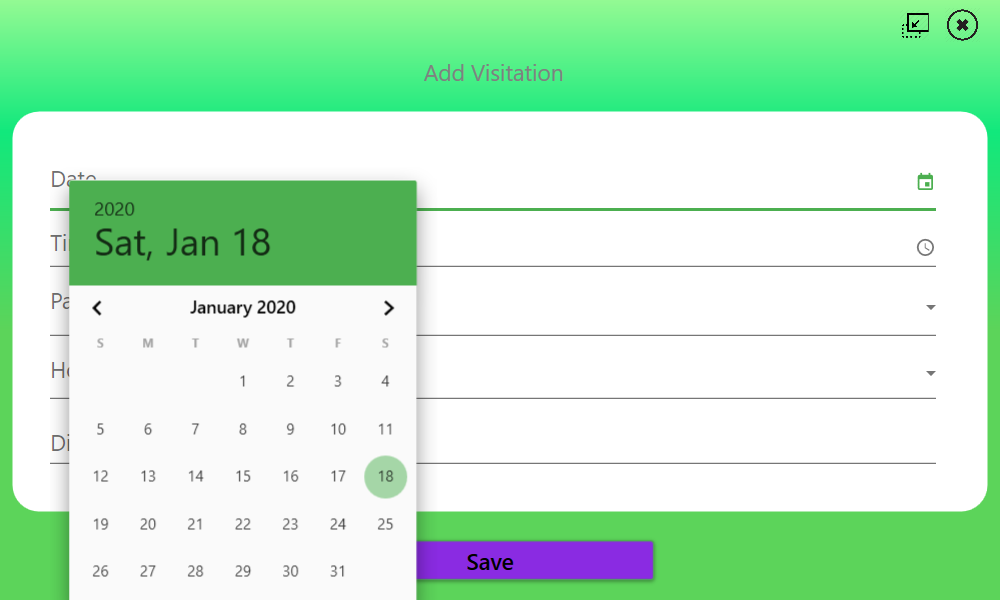


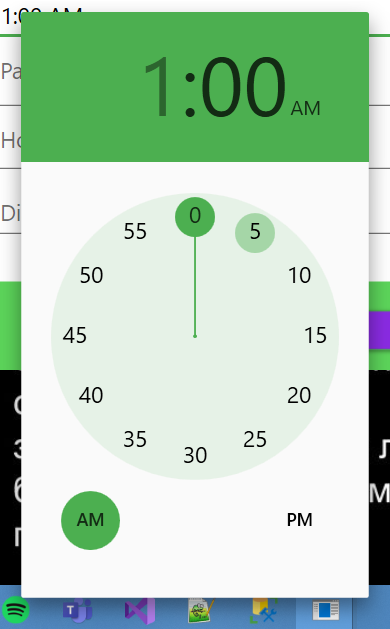
Във вторият таб се визуализира информацията за всички посещения при вписалият се вече в системата доктор. Информацията за тези записи се визуализират като се прави GET HTTP заявка, за да вземем необходимата информация. Докторът има опция да види по-детайлно всяко посещение като кликне върху info бутонът. По този начин се отваря Edit формата, която позволява на докторът да направи промяна върху съответният запис и кликне бутонът save. При кликането на този бутон се прави PUT заявка, която модифицира съответният запис. Ако е успешно се извиква формата за съобщения и информира потребителят, че записът е модифициран успешно, в противен случай се изписва определена грешка. Има направена валидация за всяко едно от полетата, които могат да се коригират. Също така Докторът има възможност да изтрие определено посещение, като кликне бутонът Delete. При натискането на този бутон се визуализира форма за съобщение за потвръждение дали наистина потребителят иска да изтрие този запис. При натискането на Yes се прави DELETE HTTP заявка, която изтрива записът, а при натискането на No се затваря съобщението и се остава във формата за корекции. Таблицата със всички записи може да се филтрира във възходящ и низходящ ред спрямо всяка колона. Докторът може да види по-детайлна информация за пациентът, на когото е бил направен прегледът и за болницата, където е бил прегледан. При създаването на посещение – избирането на болница или пациет става чрез dopdown поле, което съдържа всички регистрирани пациенти в базата данни, а за болницата се визуализират всички болници, в които работи този доктор.



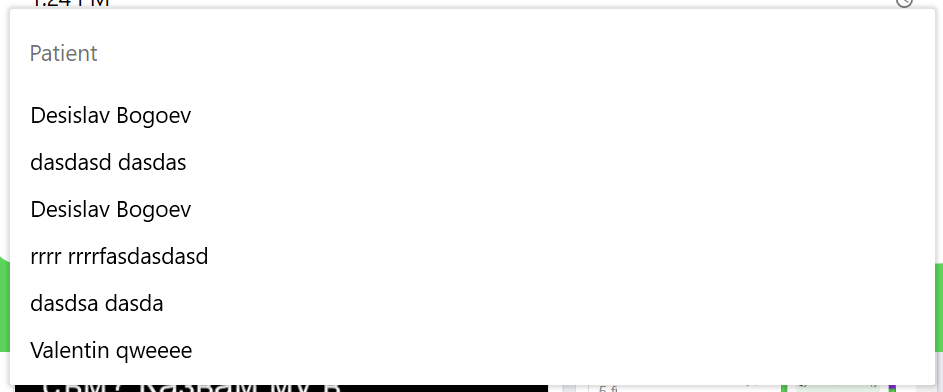


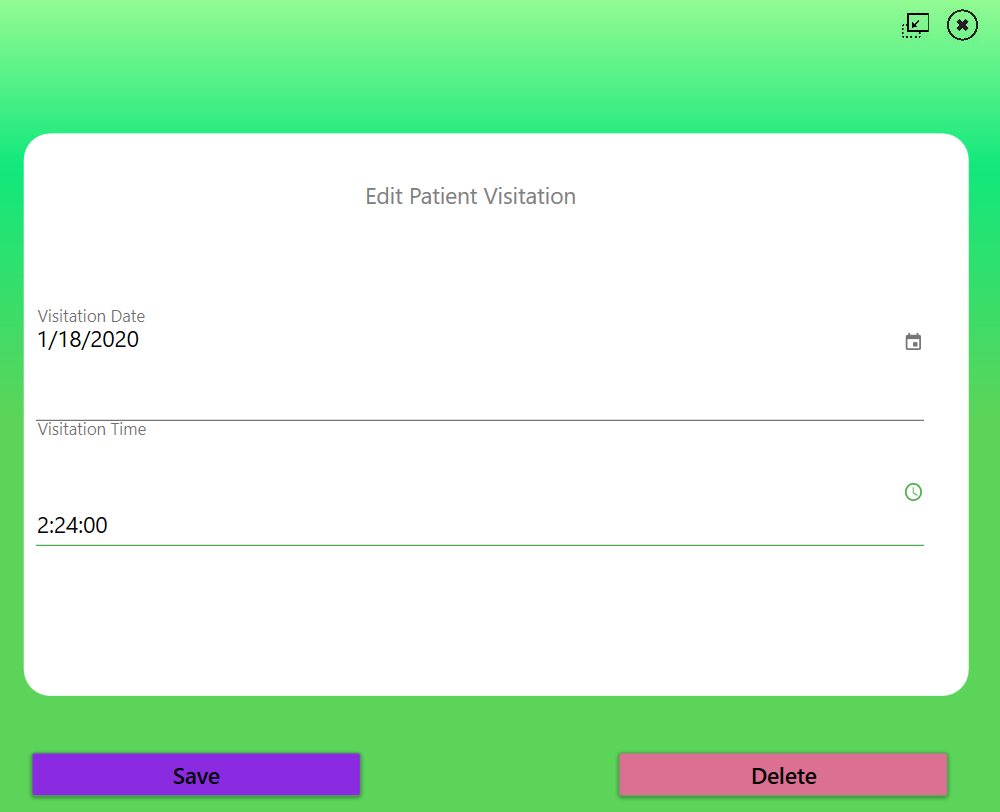


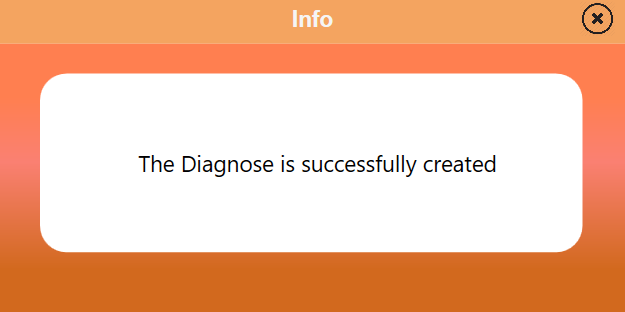
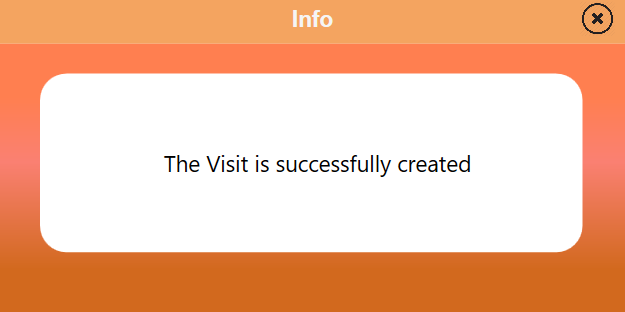


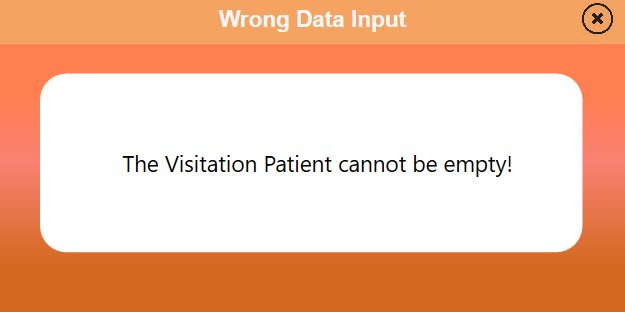
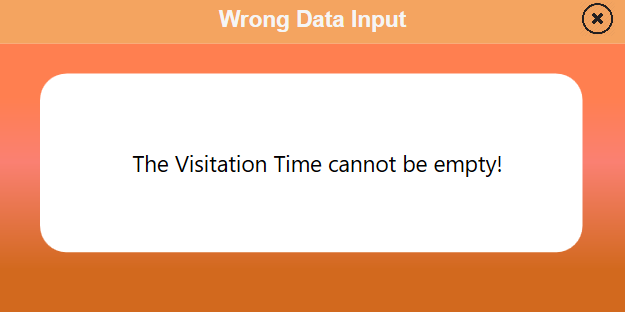


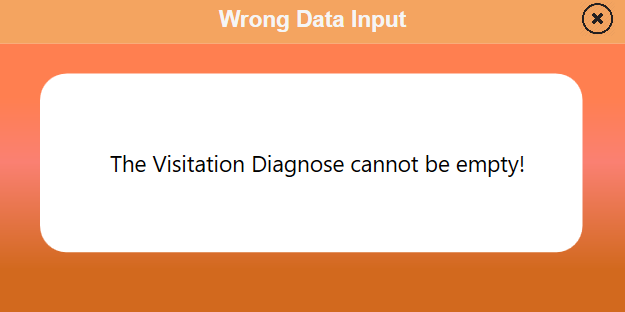


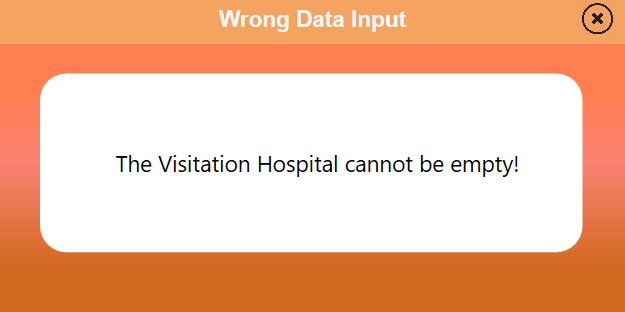


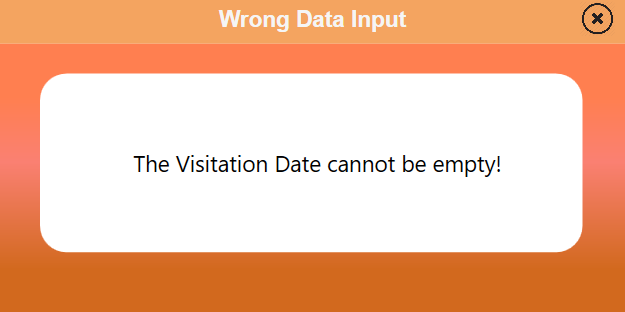






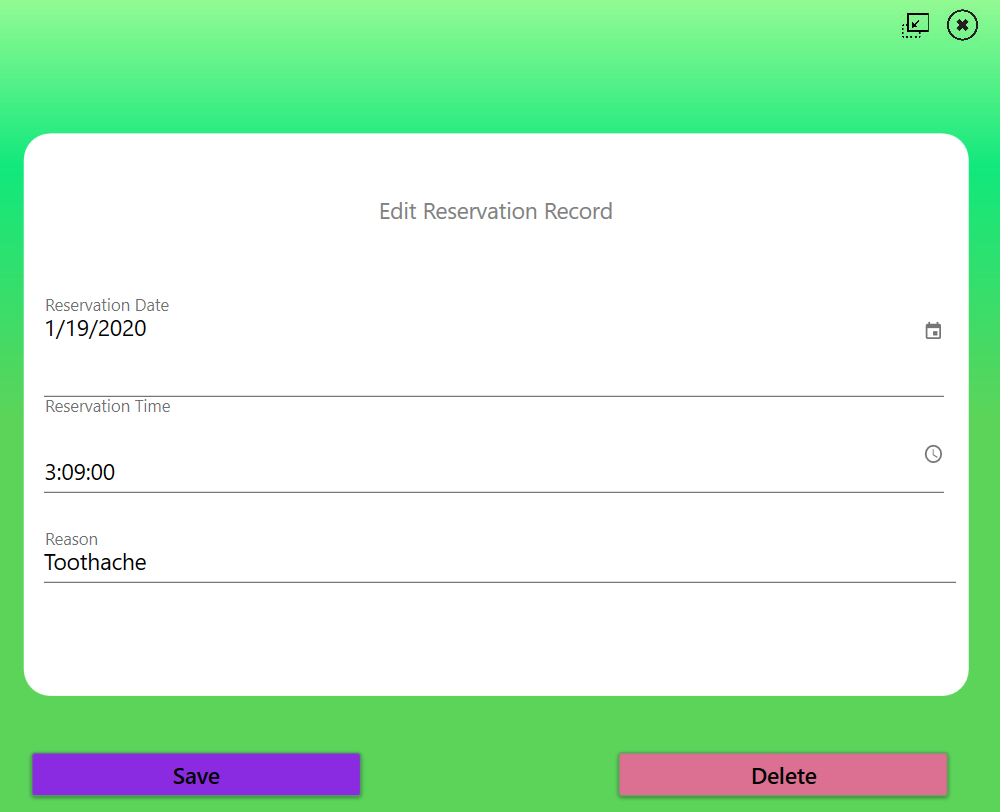


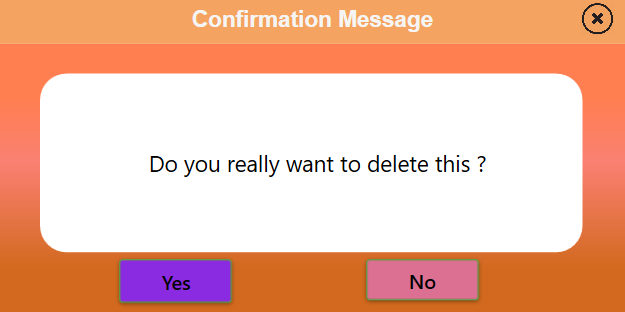
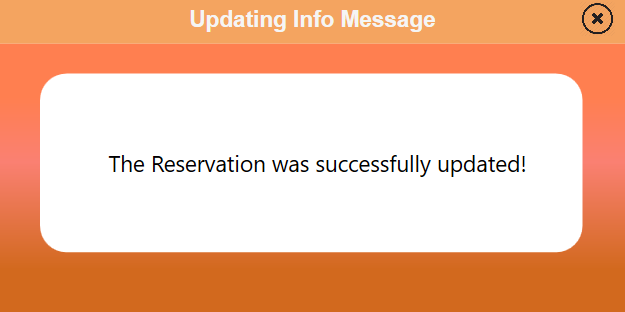


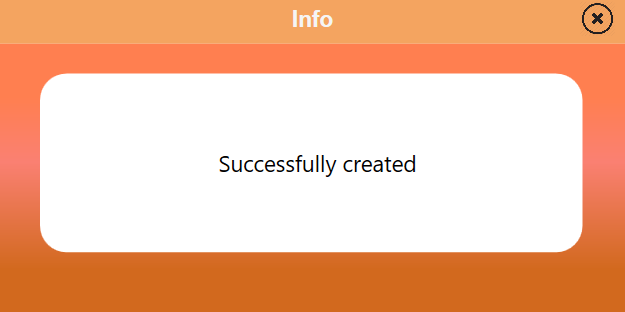


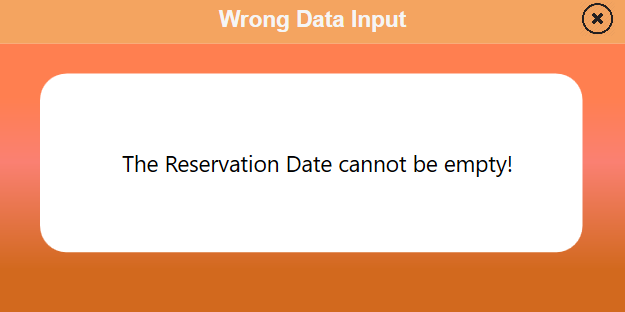
Друга опция, която предоставя Докторската форма е визуализацията на списък с всички резервации, които е направил докторът със съответният пациент в съответната болница. Визуализацията на тези резервации се постига чрез извикване на GET HTTP заявка, която ще върне наличните записи от базата данни и след това ще бъдат показани на дисплея. Списъкът със резерваци визуализира - часът, датата, пациентът, болницата и причината Докторът може да види по-детайлно информация за съответният запис като натиске инфо бутонът, който ще визуализира формата за корекции, на която има два бутона – един за запис и едни за изтриване. При едитването на резервация може да се модифицират часът, датата и причината, поради която се прави. При натискане на Save бутонът се прави POST HTTP заявка, която модифицира записа и извежда съобщение за успешно модифициране на записа в базата данни, когато е успешно и съобщение за грешка, когато не е . Докторът има възможност да изтрие определена резервация, като кликне бутонът Delete. При натискането на този бутон се визуализира форма за съобщение за потвръждение дали наистина потребителят иска да изтрие този запис. При натискането на Yes се прави DELETE HTTP заявка, която изтрива записът, а при натискането на No се затваря съобщението и се остава във формата за корекции. Таблицата със всички записи може да се филтрира във възходящ и низходящ ред спрямо всяка колона. Докторът може да види по-детайлна информация за пациентът, на когото е била направена резервацията за болницата, където ще бъде прегледан. Има Make a Reservation бутон, който отваря нова форма за резервация, в която има валидация за всяко едно от полетата, които ще се попъват. При натискането на Save се прави POST HTTP заявка, която създава нова резервация в базата данни и се визуализира прозорец, който информира потребителят, че резервацията е успешно направена, в противен случай се визуализира съобщение за грешка. При създаването на резервация– избирането на болница или пациет става чрез dopdown поле, което съдържа всички регистрирани пациенти в базата данни, а за болницата се визуализират всички болници, в които работи този доктор.

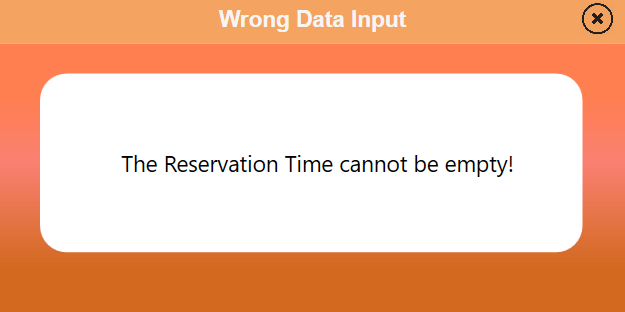


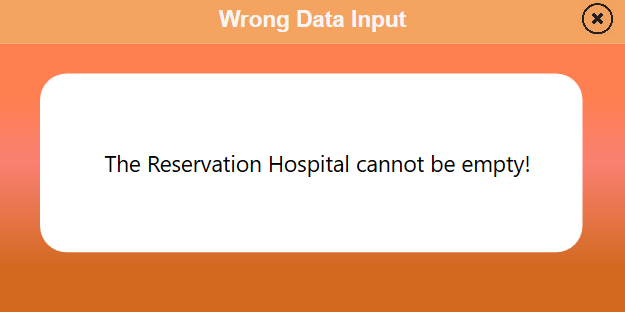


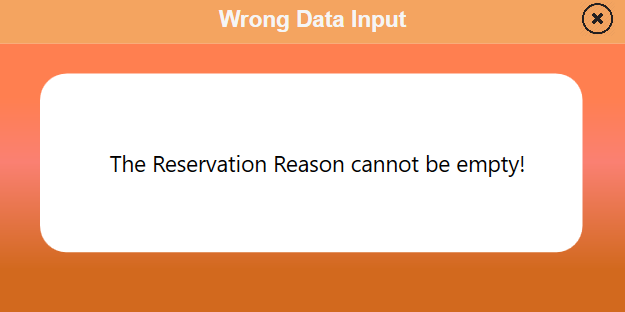
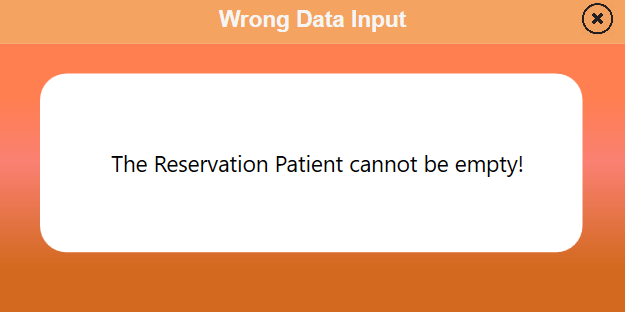


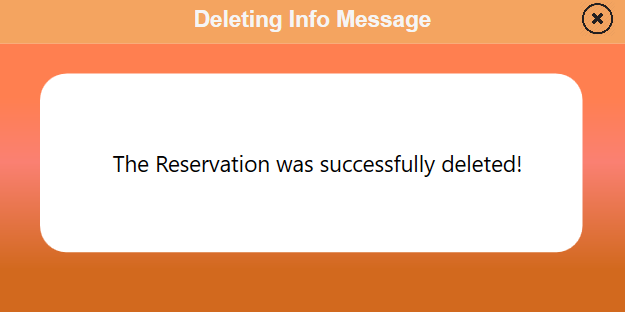




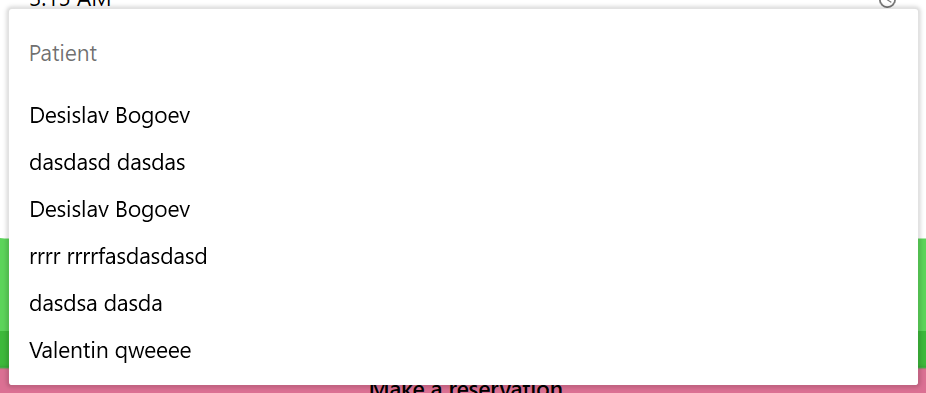




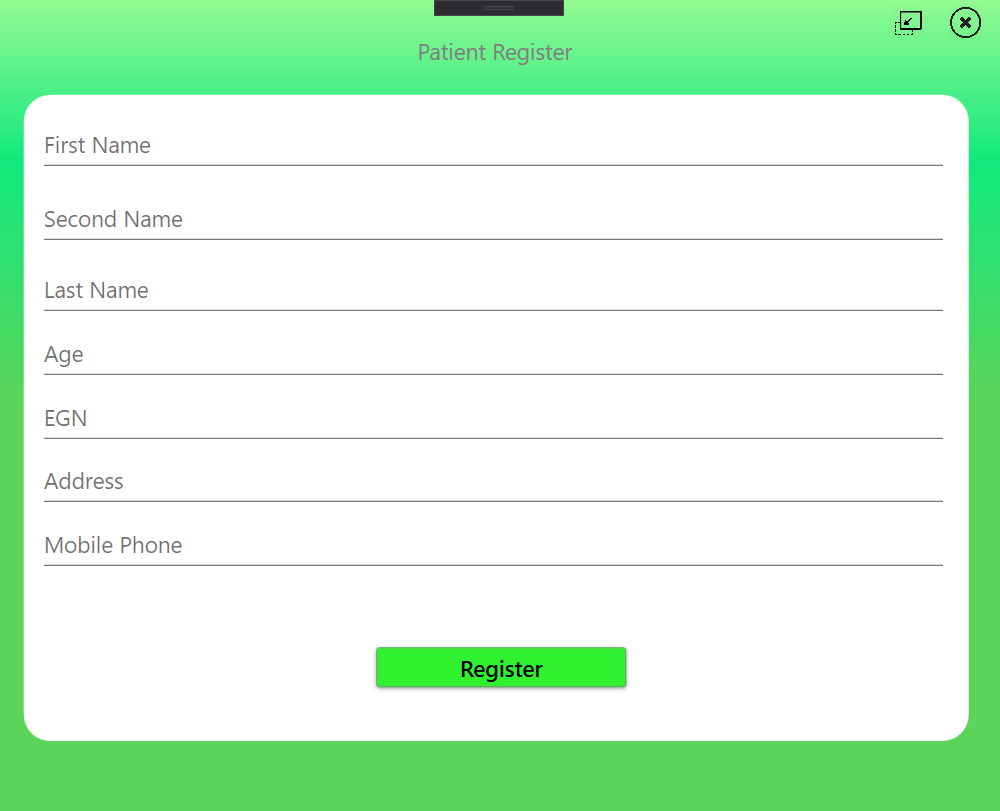


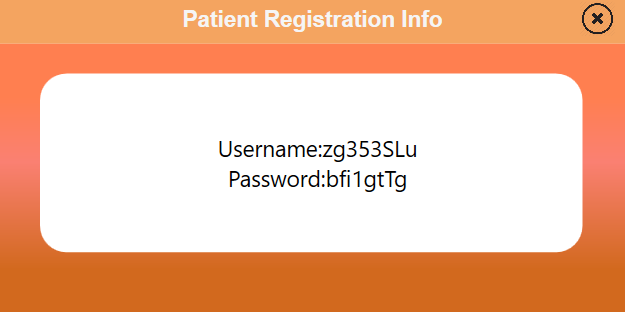


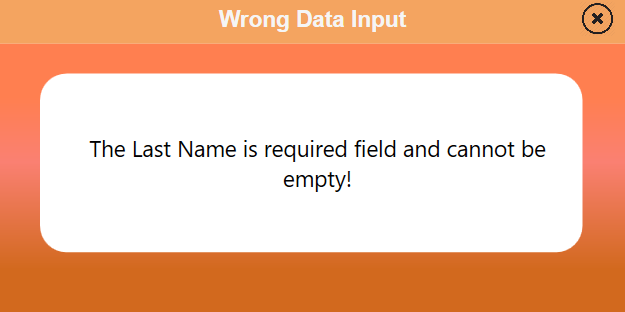


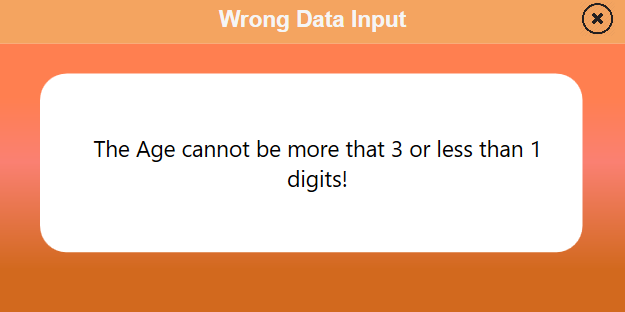


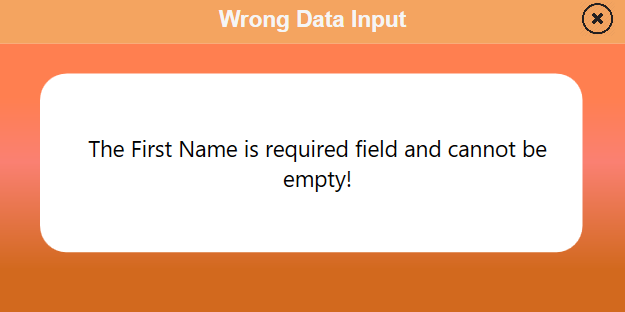
Последната опция, която има докторът е да регистрира пациент. След като клитне на таба Register Patient се отваря форма за регистация на пациент. Полетата в нея са валидирани и ако се опита да запази запис, който да има празно поле или с невалидни данни ще се отвори прозорец за съобщения, който ще покаже на докторът каква е грешката и какви са ограниченията за съответното поле. Когато се натисне бутонът Register се извиква POST HTTP заявка, която създава нов потребител като пациент в базата данни. При успешно създаване на нов пациент се визуализира съобщение с неговите потребителко име и парола, които се генерират автоматично от системата и докторът трябва да предостави на пациентът, за да има достъп до системата. При неуспешно създаване на потребител се визуализира отново прозорец със съответната грешка и при затварянето му програмта продължава норманото си функциониране.

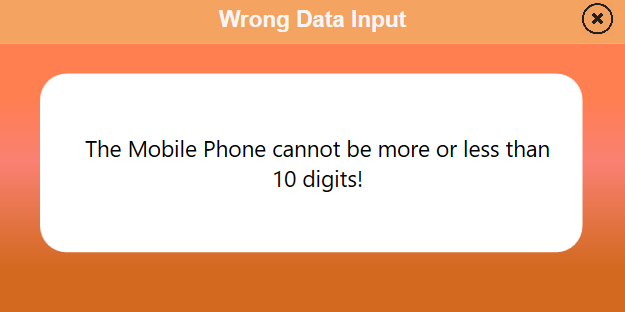


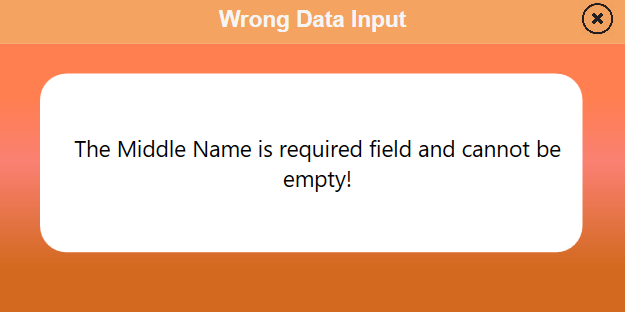








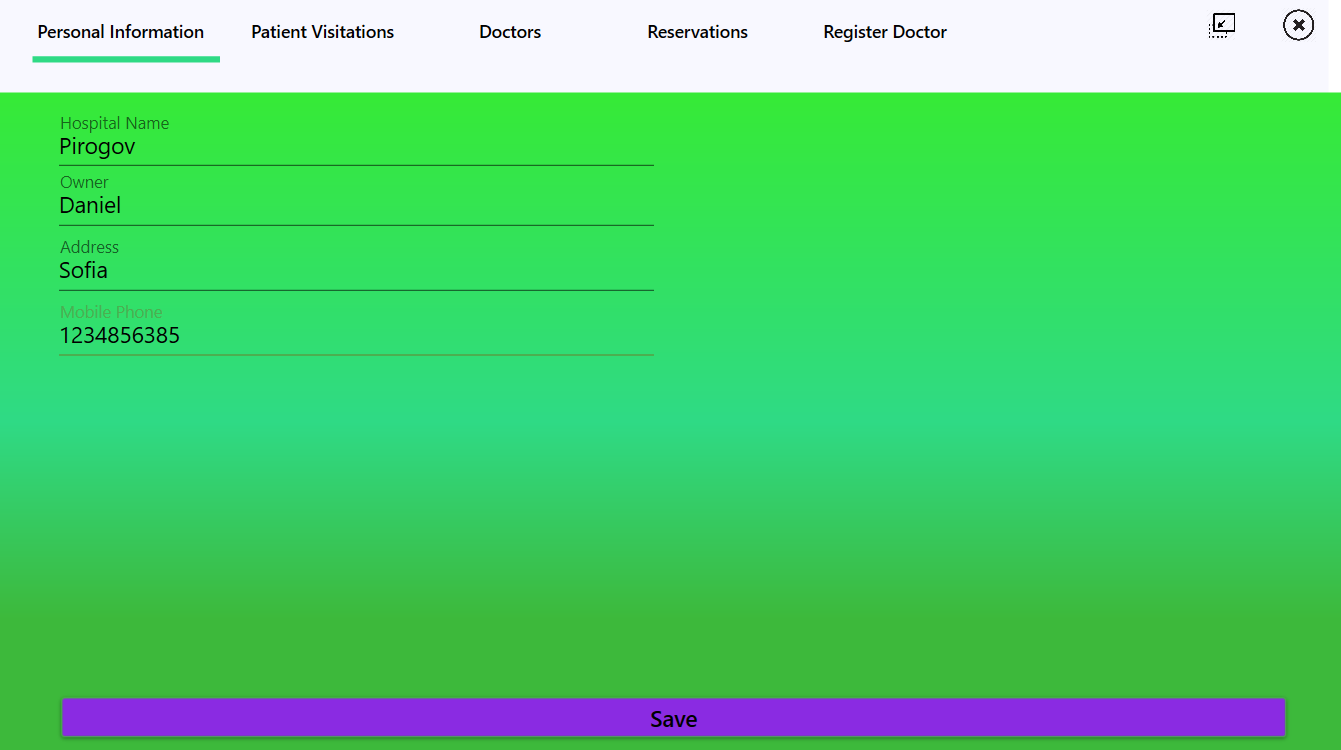


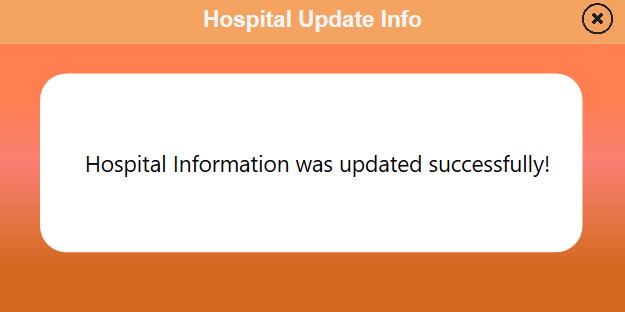
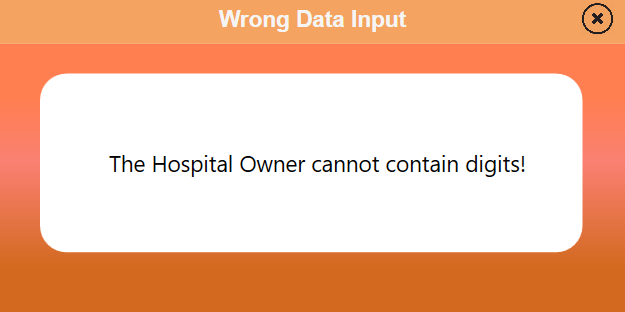


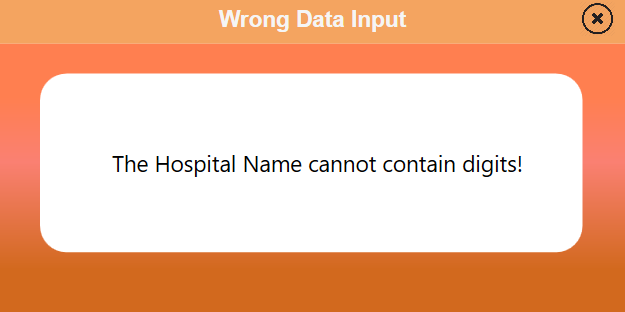
**Hospital Form**

Тази форма се отваря, когато потребител с роле Hospital се впише успешно в системата. Болницата като потребител може да види следната информация – информация за самата болница, посещенията, които са направени в тази болница, докторите, които работят в тази болница, резервациите , които са направени в болницата при съответният лекар и същото така има опция да се създаде лекар.

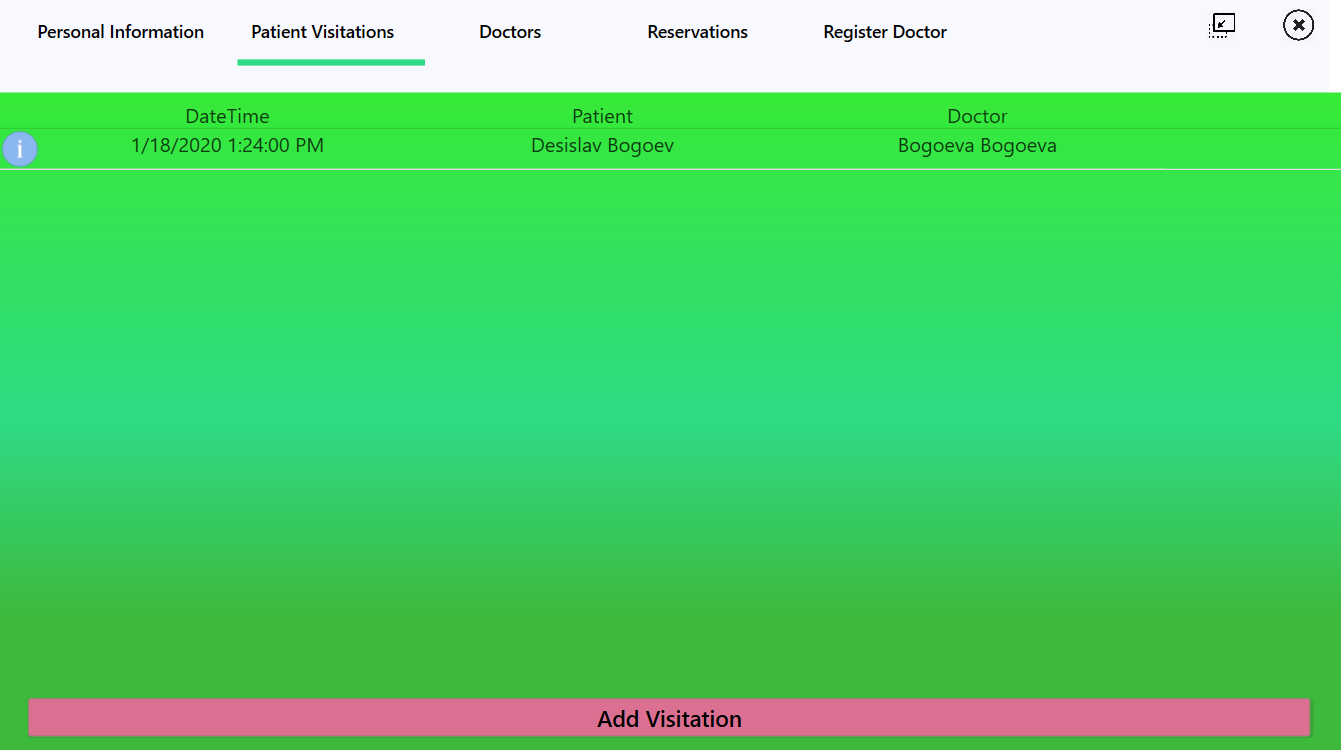
В първият таб се визуализират личните данни на вече логналната се болница в системата. Те се вземат чрез GET HTTP заявка, която връща информацията за болницата. Тя има възможност да коригира своите данни. След като е готова с корекцията на данните и се натисне бутонът Save се прави PUT HTTP заявка, която модифицира записът в базата данни. Има направена валидация за всяко от едно от полета, които може да коригира болницата. Ако е са въведени неправилни данни или някое от полетата е празно при опит за модификация се визувализира форма за съобщение , която дава индикация каква е грешката и какво трябва да се направи. При успешна модификация на записа се изпивсва съобщение, което информира потребителят, че промяната на данните е успешна. Това отново се постига, че формата , която е създадена специало за съобщения.

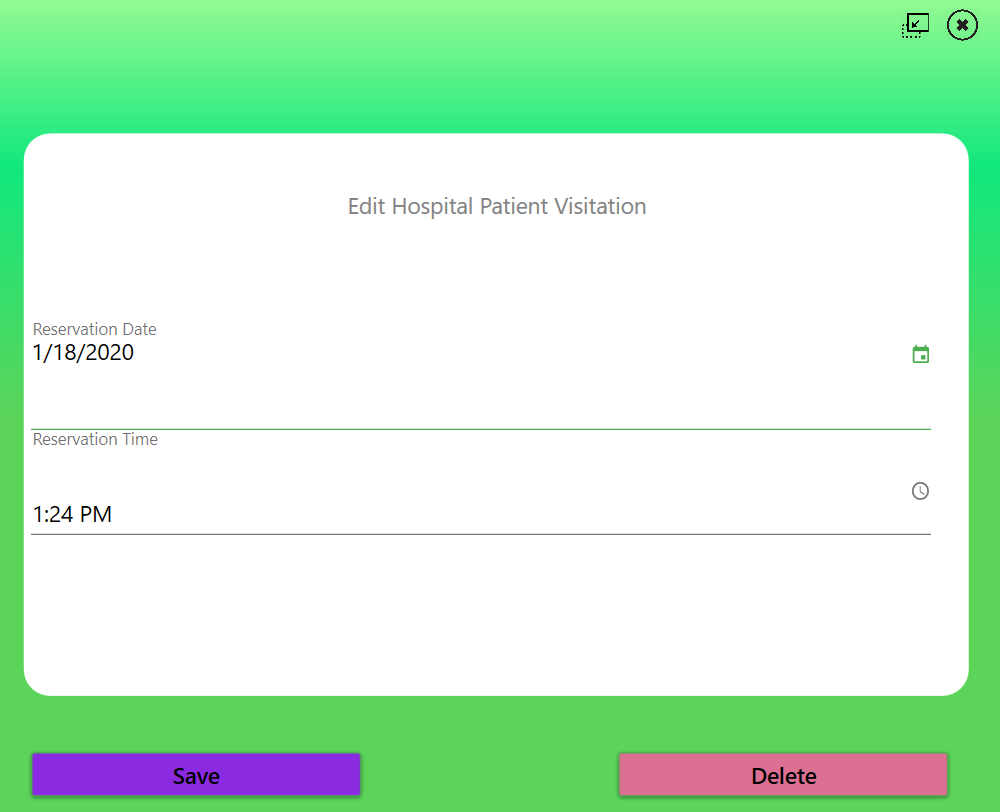






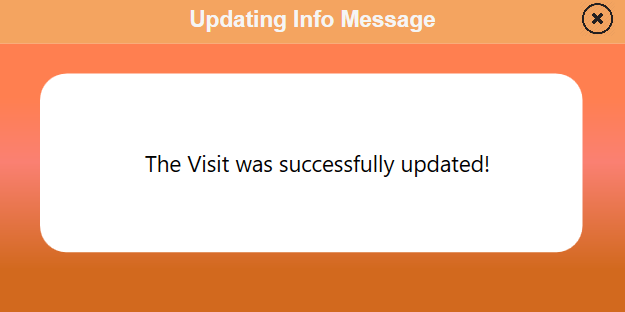
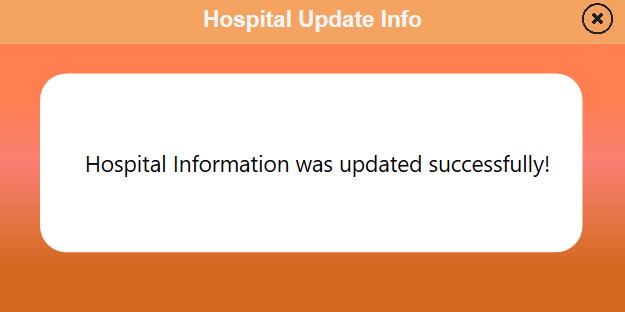
Във вторият таб се визуализира информацията за всички посещения при вписалaта се в системата болница. Информацията за тези записи се визуализират като се прави GET HTTP заявка, за да вземем необходимата информация. Болницата има опция да види по-детайлно всяко посещение като кликне върху info бутонът. По този начин се отваря Edit формата, която позволява на болницата да направи промяна върху съответният запис и кликне бутонът save. При кликането на този бутон се прави PUT заявка, която модифицира съответният запис. Ако е успешно се извиква формата за съобщения и информира потребителят, че записът е модифициран успешно, в противен случай се изписва определена грешка. Има направена валидация за всяко едно от полетата, които могат да се коригират. Също така болницата има възможност да изтрие определено посещение, като кликне бутонът Delete. При натискането на този бутон се визуализира форма за съобщение за потвръждение дали наистина потребителят иска да изтрие този запис. При натискането на Yes се прави DELETE HTTP заявка, която изтрива записът, а при натискането на No се затваря съобщението и се остава във формата за корекции. Таблицата със всички записи може да се филтрира във възходящ и низходящ ред спрямо всяка колона. Болницата може да види по-детайлна информация за пациентът, на когото е бил направен прегледът и за докторът, който го е прегледал. При създаването на посещение – избирането на доктор или пациет става чрез dopdown поле, което съдържа всички регистрирани пациенти в базата данни, а за докторът се визуализират всички доктори, в които работят в тази болница.



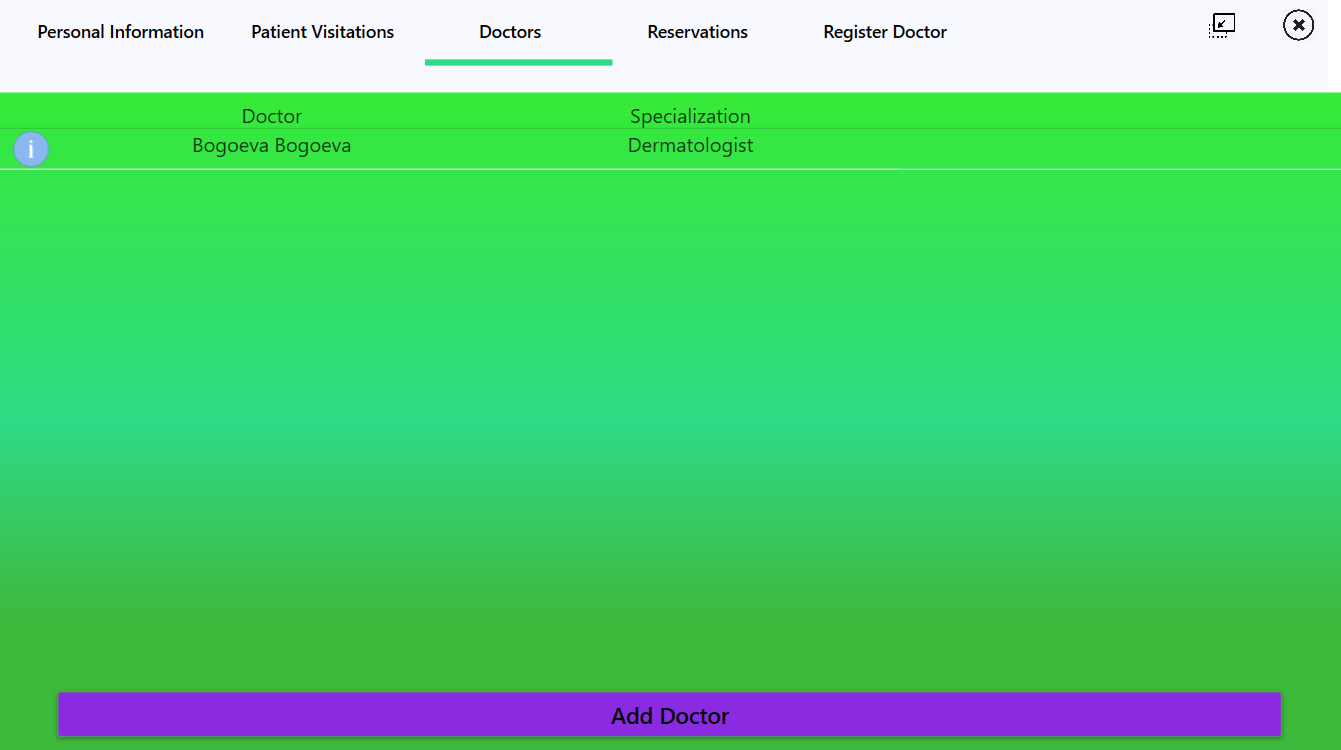


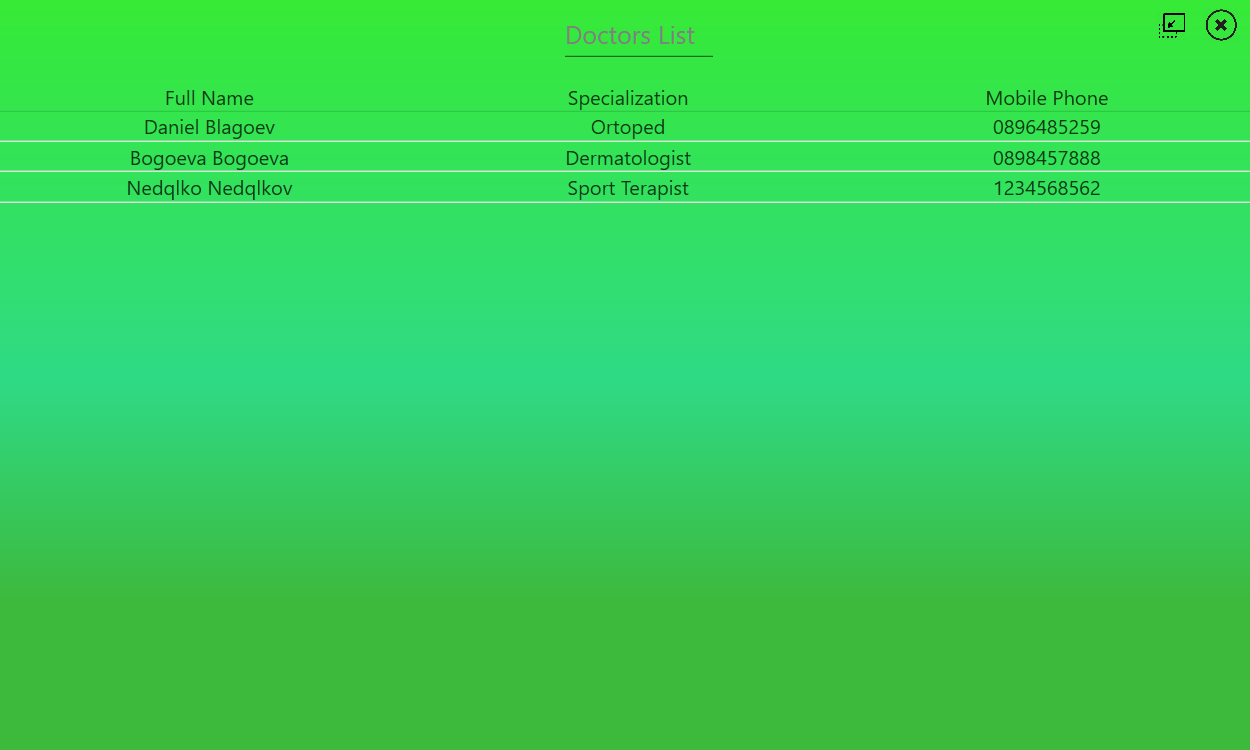


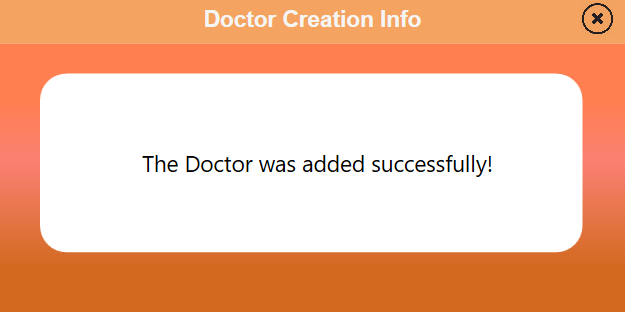




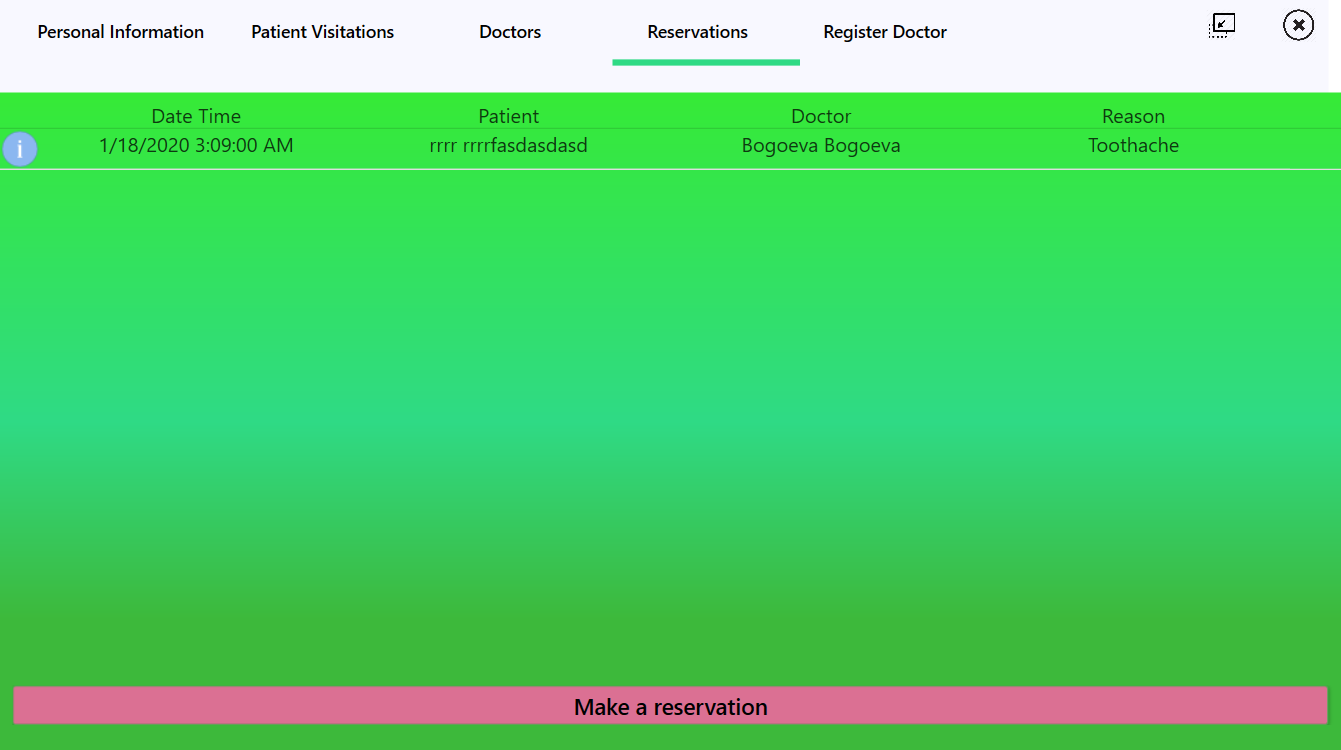
В третия таб са визуализирани всички доктори, които работят в тази болница. Като се кликне върху името на доктора се отваря инфо форма, която предоставя по-детайлна информация за самият доктор – име, презиме, фамилия, адрес, години, телефонен номер. Има опция докторът да добави нов доктор, който ще работи към тази болница. Това се случва с кликането на бутона Add Doctor се отваря нова форма, която предоставя списък със всички доктори, налични в системата. Болницата може да избере, кой от тях да добави към списъка с доктори, които ще работят в болницата. Визуализацията на всички доктори става с GET HTTP заявака, а добавянето на доктор към дадена болница става чрез POST HTTP заявка, която записва в базата данни, че определеният доктор работи в тази болница.



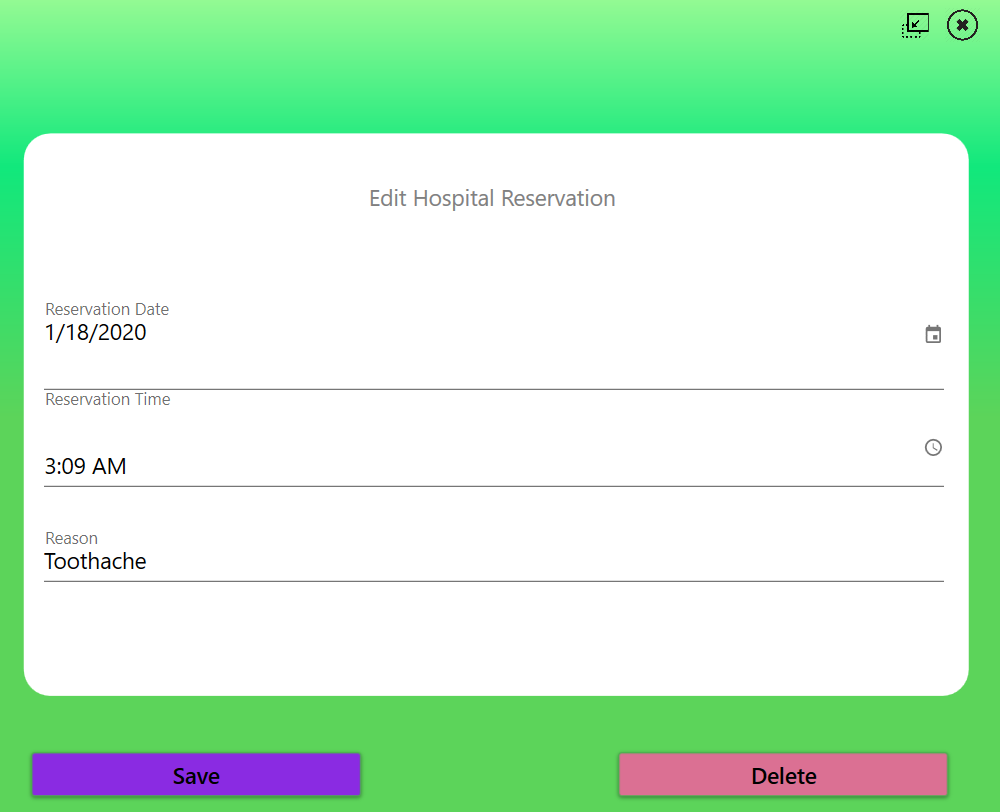


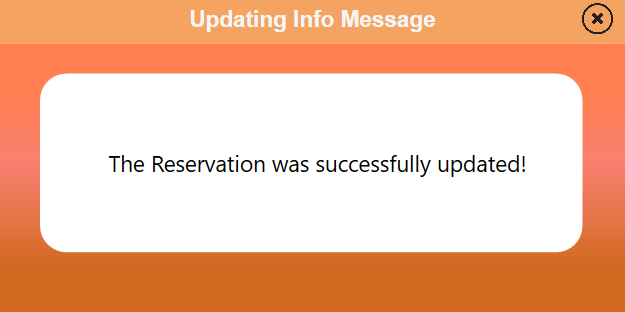


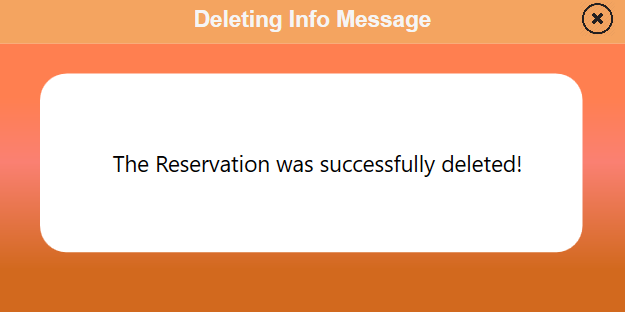
Друга опция, която предоставя болничната форма е визуализацията на списък с всички резервации, които са направени при работещият в болницата доктор със съответният пациент. Визуализацията на тези резервации се постига чрез извикване на GET HTTP заявка, която ще върне наличните записи от базата данни и след това ще бъдат показани на дисплея. Списъкът със резерваци визуализира - часът, датата, пациентът, докторът и причината. Болницата може да види по-детайлно информация за съответният запис като натиске инфо бутонът, който ще визуализира формата за корекции, на която има два бутона – един за запис и едни за изтриване. При едитването на резервация може да се модифицират часът, датата и причината, поради която се прави. При натискане на Save бутонът се прави POST HTTP заявка, която модифицира записа и извежда съобщение за успешно модифициране на записа в базата данни, когато е успешно и съобщение за грешка, когато не е . Болницата има възможност да изтрие определена резервация, като кликне бутонът Delete. При натискането на този бутон се визуализира форма за съобщение за потвръждение дали наистина потребителят иска да изтрие този запис. При натискането на Yes се прави DELETE HTTP заявка, която изтрива записът, а при натискането на No се затваря съобщението и се остава във формата за корекции. Таблицата със всички записи може да се филтрира във възходящ и низходящ ред спрямо всяка колона. Болницата може да види по-детайлна информация за пациентът, на когото е била направена резервацията и докторът, които ще го преглежда. Има Make a Reservation бутон, който отваря нова форма за резервация, в която има валидация за всяко едно от полетата, които ще се попъват. При натискането на Save се прави POST HTTP заявка, която създава нова резервация в базата данни и се визуализира прозорец, който информира потребителят, че резервацията е успешно направена, в противен случай се визуализира съобщение за грешка. При създаването на резервация– избирането на болница или пациет става чрез dopdown поле, което съдържа всички регистрирани пациенти в базата данни, а за докторът се визуализират всички доктори, които работят в тази болница.

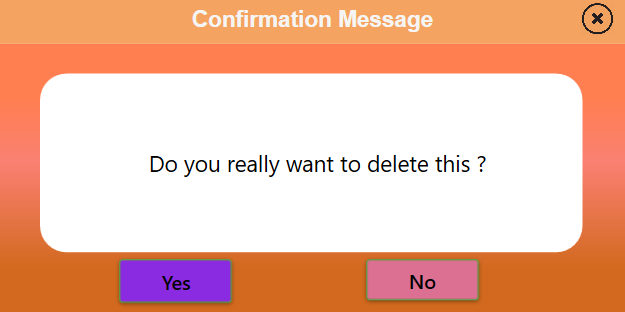
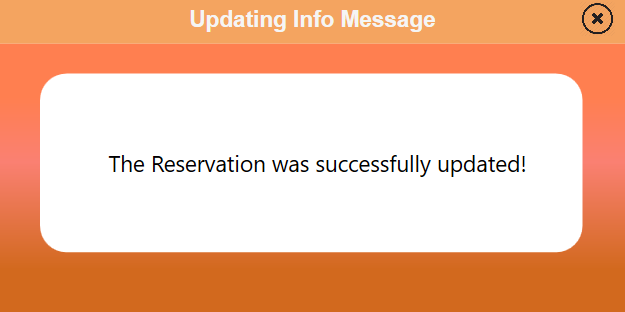
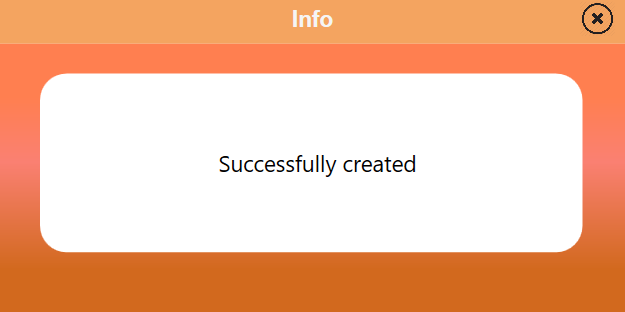
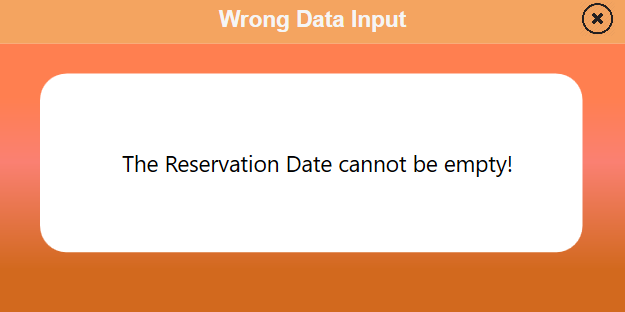
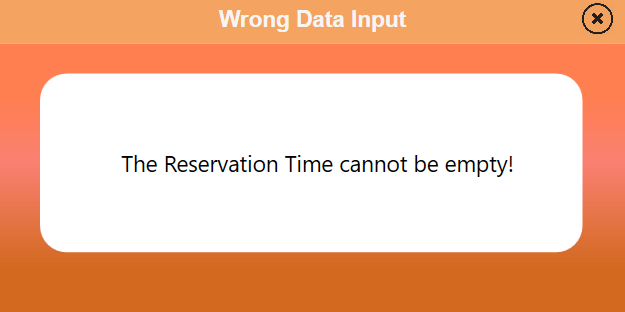
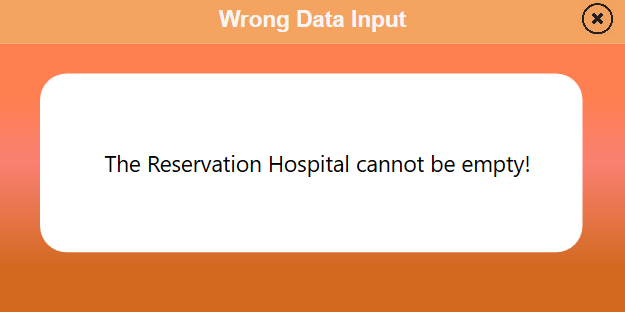


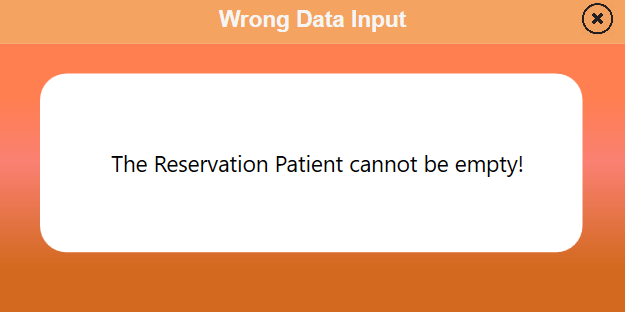
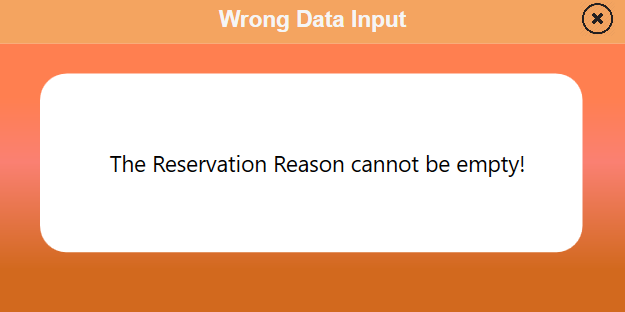




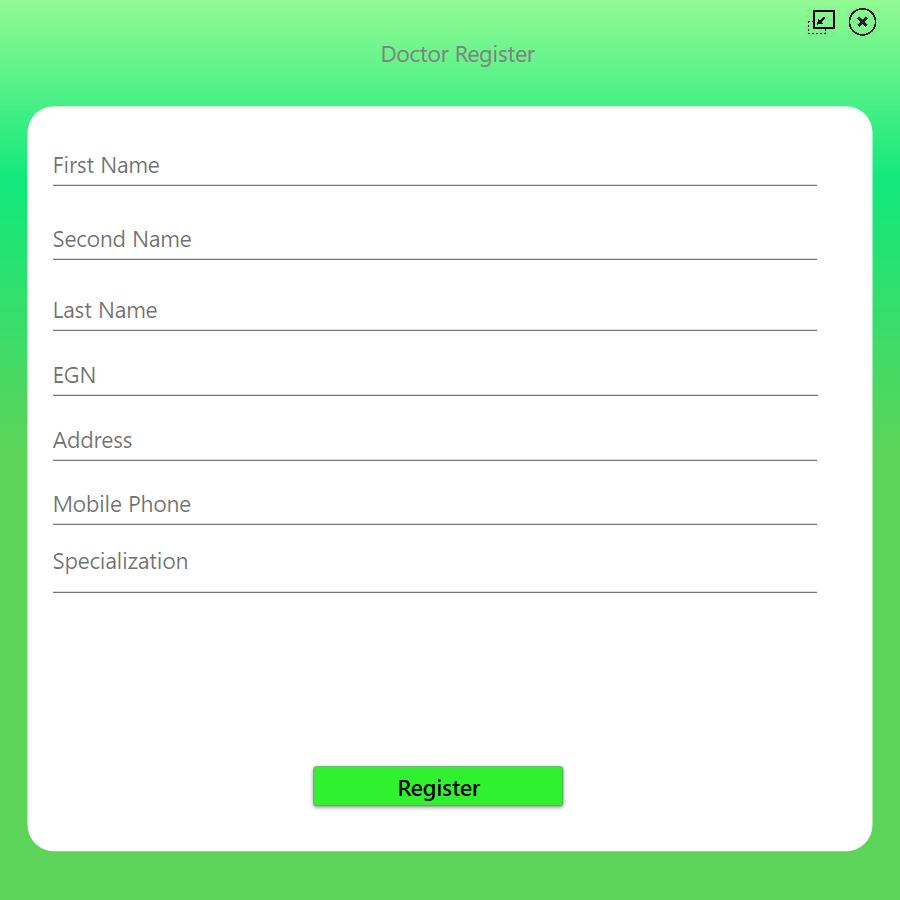






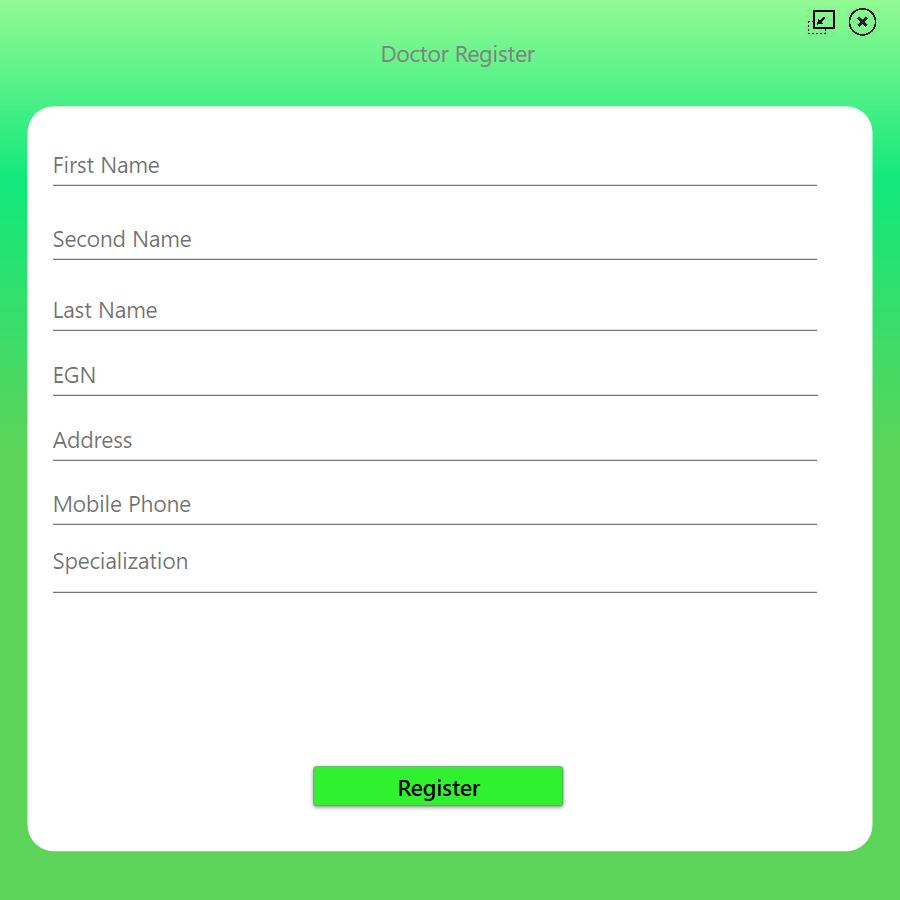


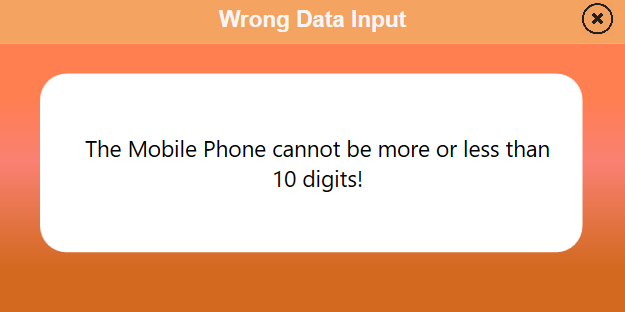
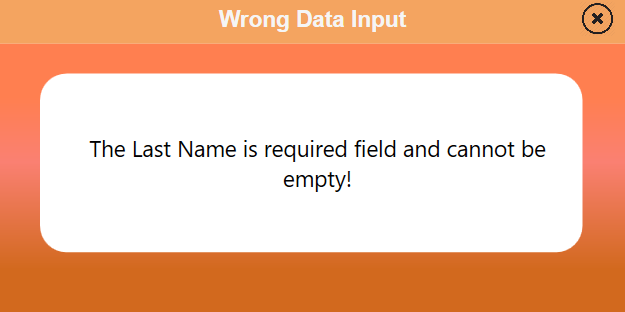
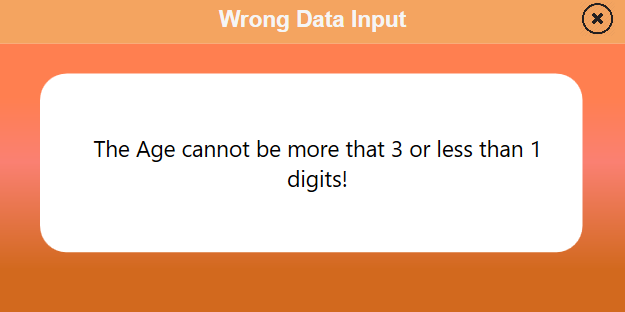
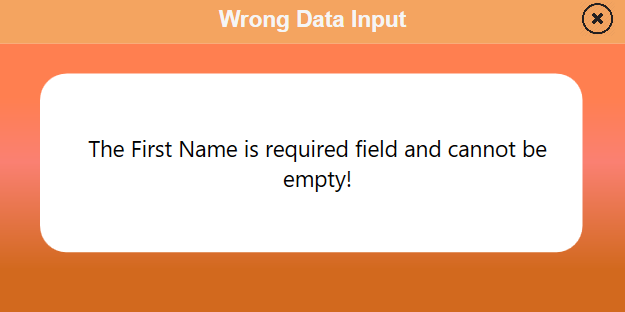
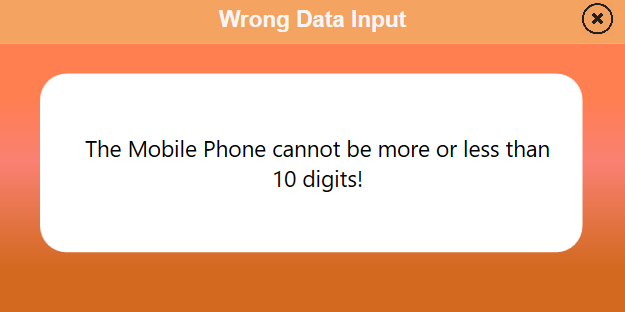
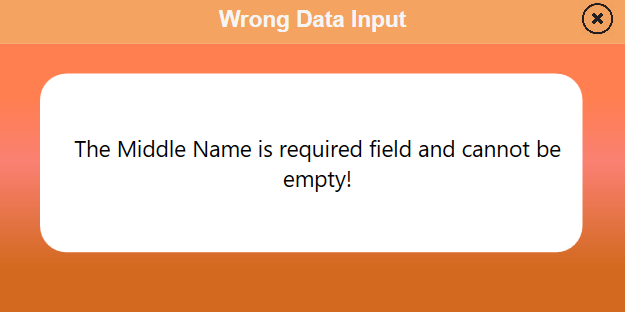
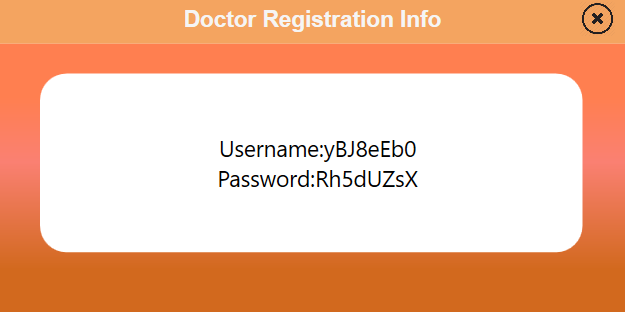
Последната опция, която има болницата е да регистрира доктор. След като клитне на таба Register Doctor се отваря форма за регистация на доктор. Полетата в нея са валидирани и ако се опита да запази запис, който да има празно поле или с невалидни данни ще се отвори прозорец за съобщения, който ще покаже на болницата каква е грешката и какви са ограниченията за съответното поле. Когато се натисне бутонът Register се извиква POST HTTP заявка, която създава нов потребител като доктор в базата данни. При успешно създаване на нов доктор се визуализира съобщение с неговите потребителко име и парола, които се генерират автоматично от системата и докторът трябва да предостави на доктора, за да има достъп до системата. При неуспешно създаване на потребител се визуализира отново прозорец със съответната грешка и при затварянето му програмта продължава норманото си функциониране.



**Register Doctor Form**

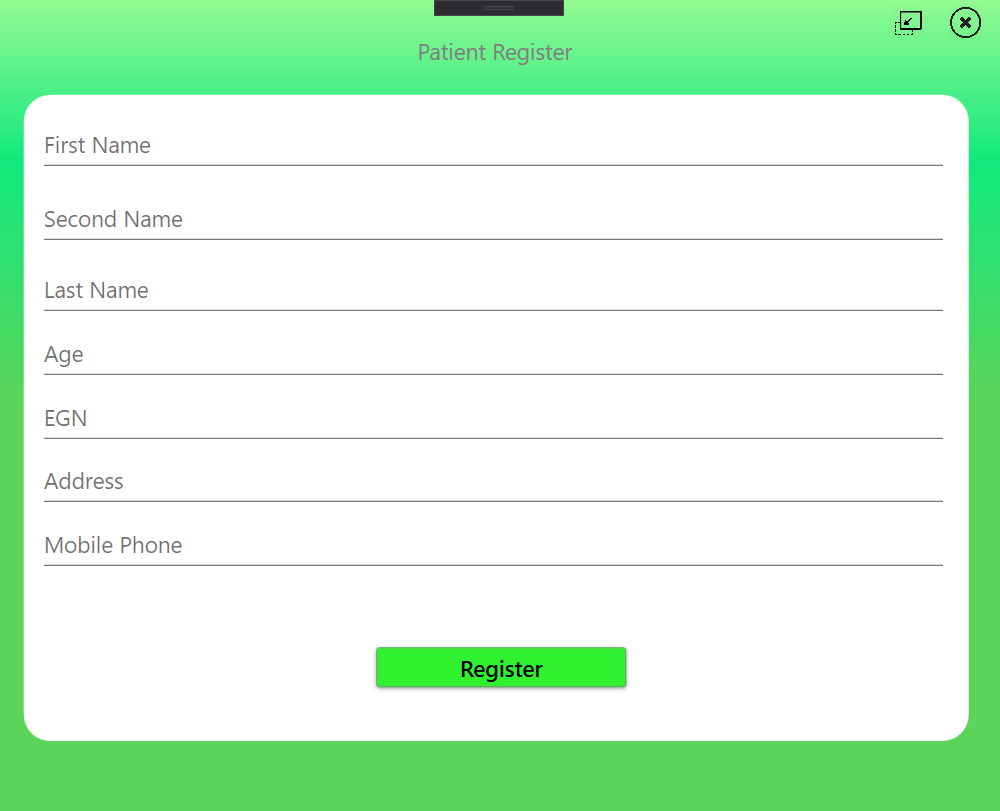
Тази форма се използва за регистриране на нов доктор като потребител в системата. Прави се POST HTTP заявка, която ако е успешна връща потребителкото име и парола на новосъздаденият потребител, а ако има греша се изписва на екрана извиквайки друга форма – Message Form, която е проектира специално за визуализация на разчен тип съобщения. Полетата, които трябва да бъдат попълнени са валидирани и при наличие на грешно въведени данни или празно поле се извежда съобщение на екрана. При докторът е нужно да се попълнят следните полета – име, презиме, фамилия, години, ЕГН, адрес, мобилен телефон и специализация.





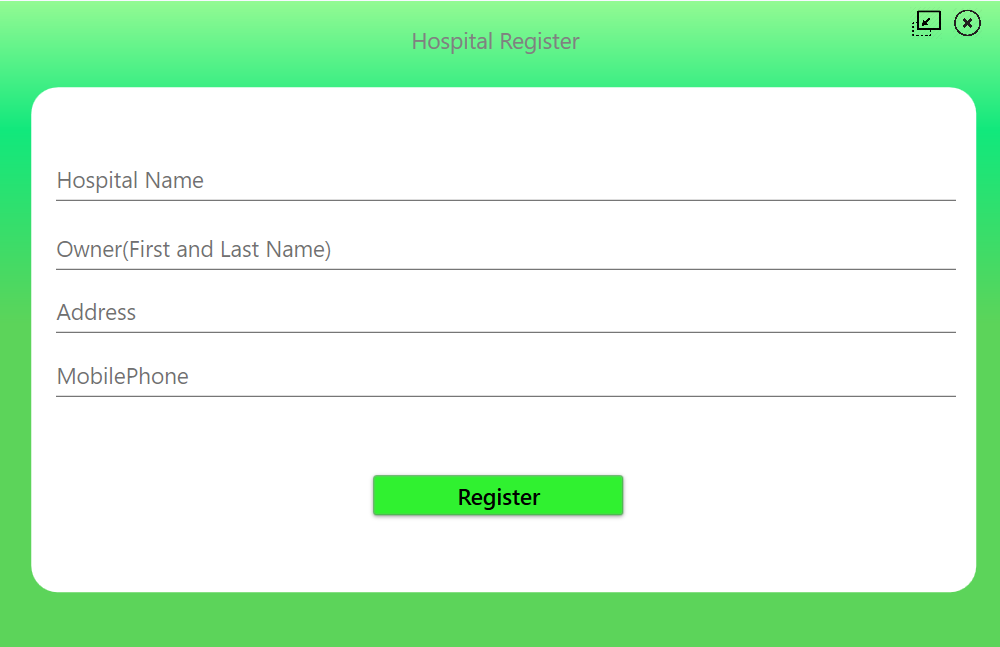
**Register Patient Form**

Тази форма се използва за регистриране на нов пациент като потребител в системата. Прави се POST HTTP заявка, която ако е успешна връща потребителкото име и парола на новосъздаденият потребител, а ако има греша се изписва на екрана извиквайки друга форма – Message Form, която е проектира специално за визуализация на разчен тип съобщения. Полетата, които трябва да бъдат попълнени са валидирани и при наличие на грешно въведени данни или празно поле се извежда съобщение на екрана. При пациента е нужно да се попълнят следните полета – име, презиме, фамилия, години, ЕГН, адрес, мобилен телефон.



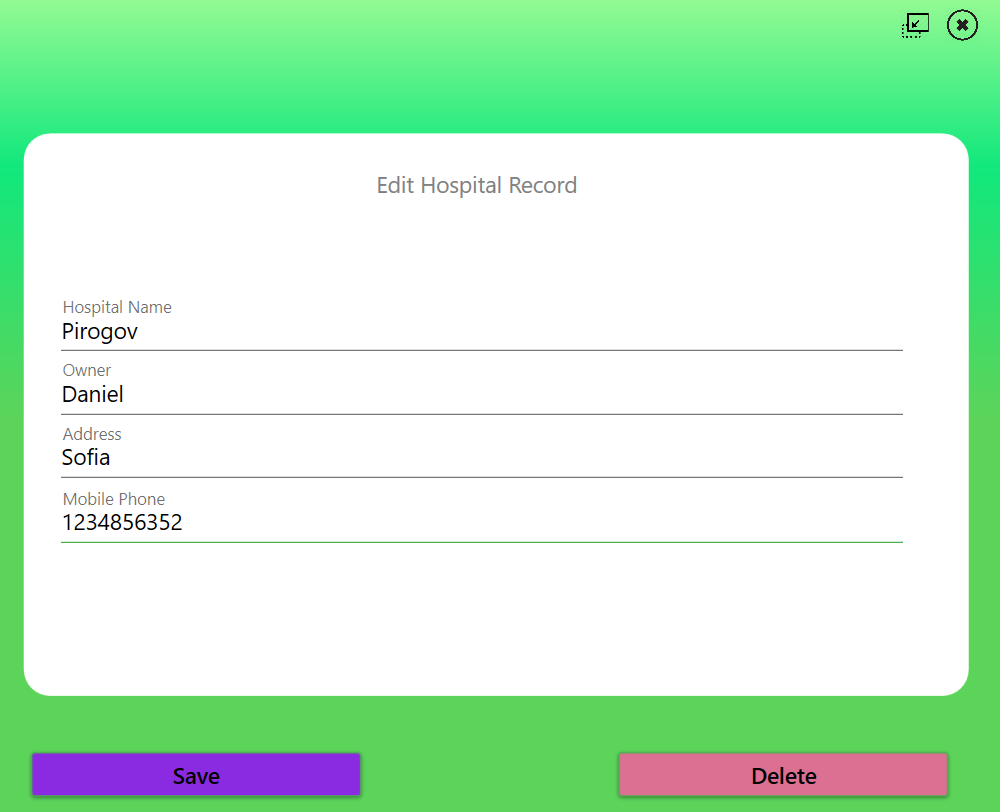
**Register Hospital Form**

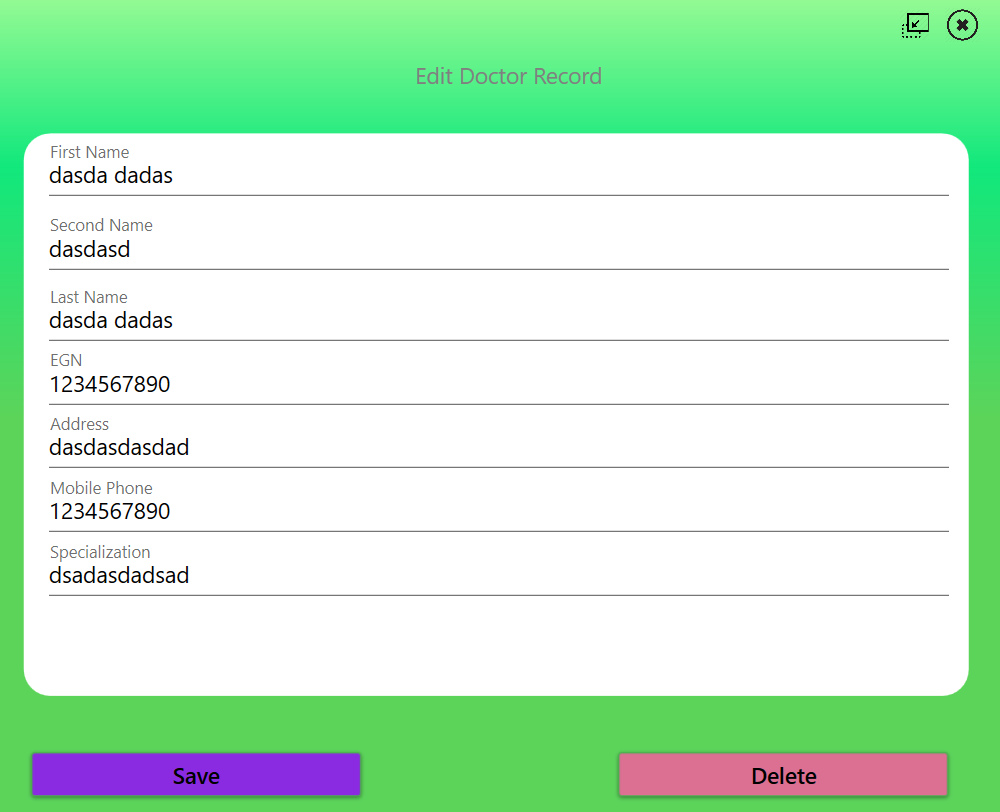
Тази форма се използва за регистриране на нов болница като потребител в системата. Прави се POST HTTP заявка, която ако е успешна връща потребителкото име и парола на новосъздаденият потребител, а ако има греша се изписва на екрана извиквайки друга форма – Message Form, която е проектира специално за визуализация на разчен тип съобщения. Полетата, които трябва да бъдат попълнени са валидирани и при наличие на грешно въведени данни или празно поле се извежда съобщение на екрана. При болницата е нужно да се попълнят следните полета – име, собственик, адрес, мобилен телефон.

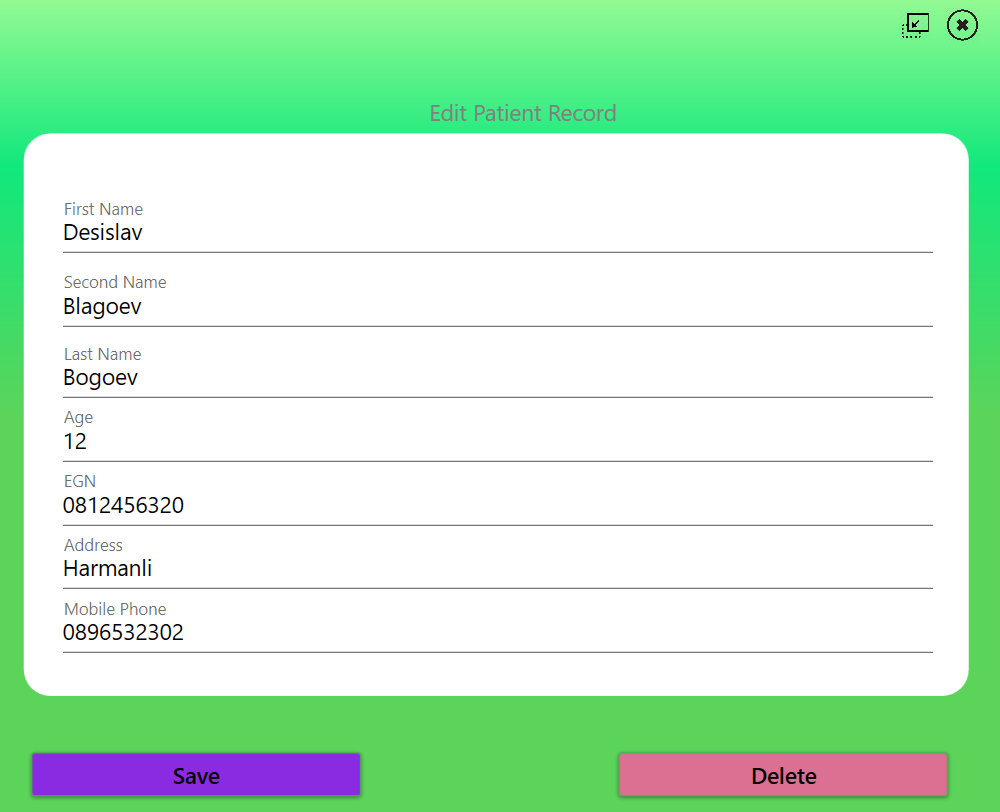


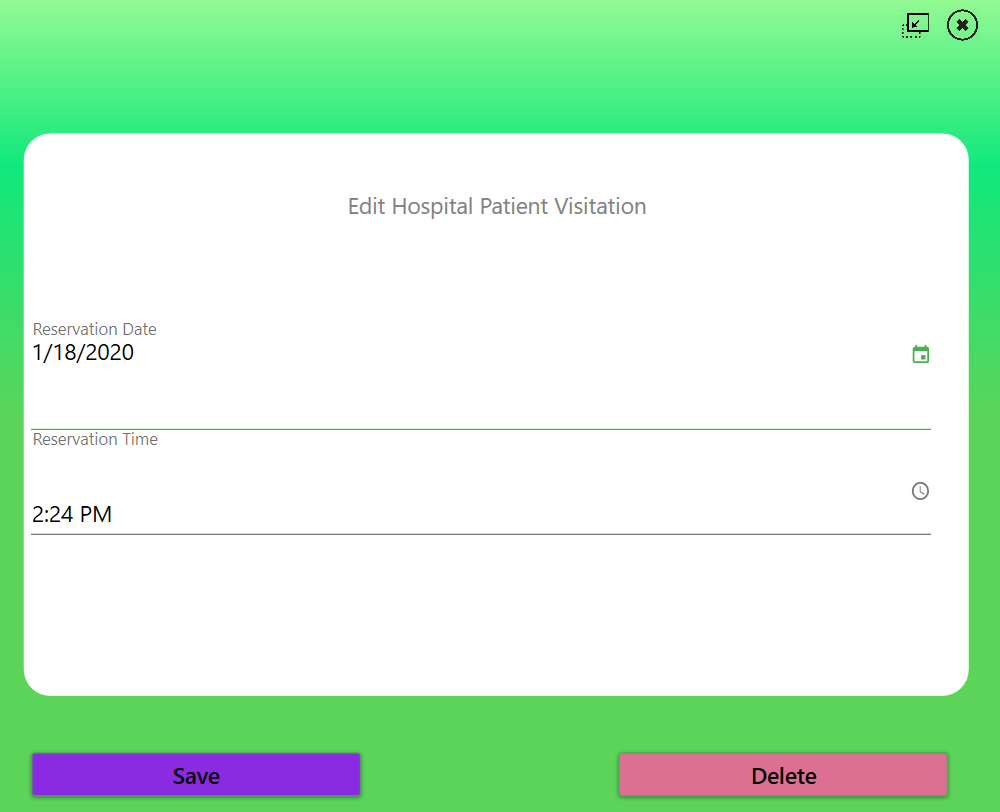
**Edit Form**

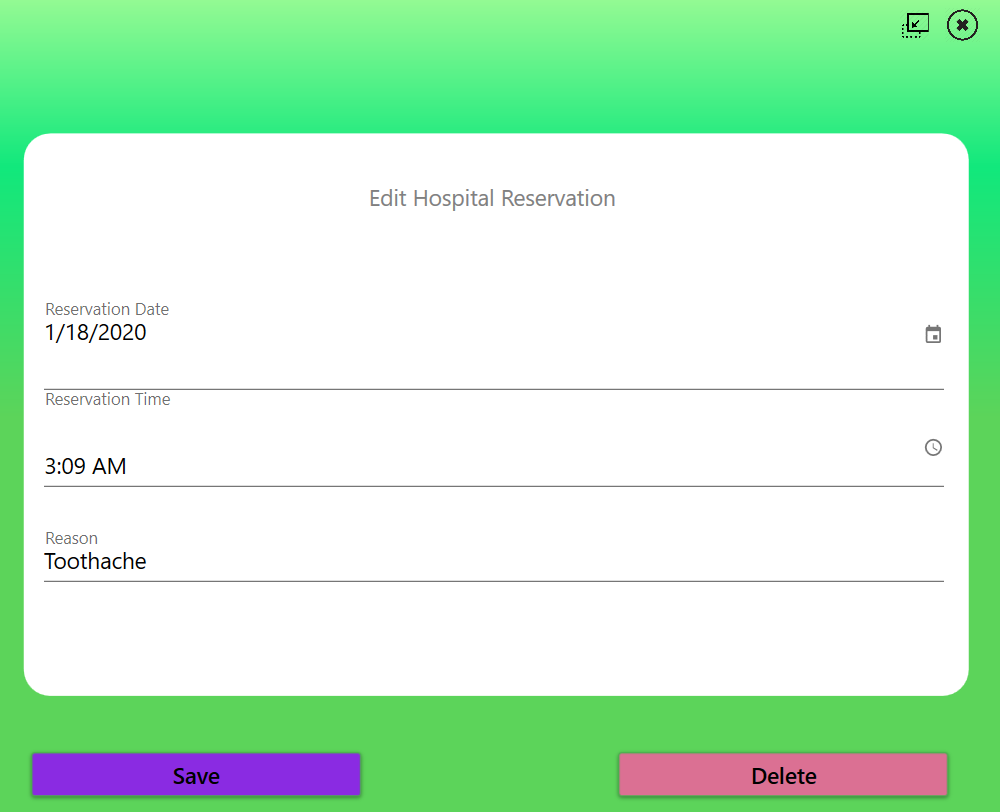
Тази форма предоставя различна информация, в зависимост от коя форма родителска форма е била извикана. В зависимост от коя форма е била извикана се визуализират различни дании, които могат да бъдат модифицирани или дори изтрити. Тази форма позволява да бъдат коригирани резервации съответно от доктор или болница, посещния на различни пациенти в зависимост дали е извикана от доктор или болница форма. Също така тази форма визуализира и позволява на админа да модифицира лични данни за които и да е пациент, доктор или болница или дори да го изтрие от системата. Модификацията на записите, които се визуализират в тази форма се правят чрез PUT HTTP заявка към определен endpoint, която прави промяната в базата от данни. Изтриването на които и да е от горе споменатите записи става с DELETE HTTP заявка към определен endpoint. При успешно модифициране или изтриване на даден запис се изписва съобщение, което информира потребителят, че операцията е успешно изпълнена, в противен случай се визуализира съобщение за грешка.









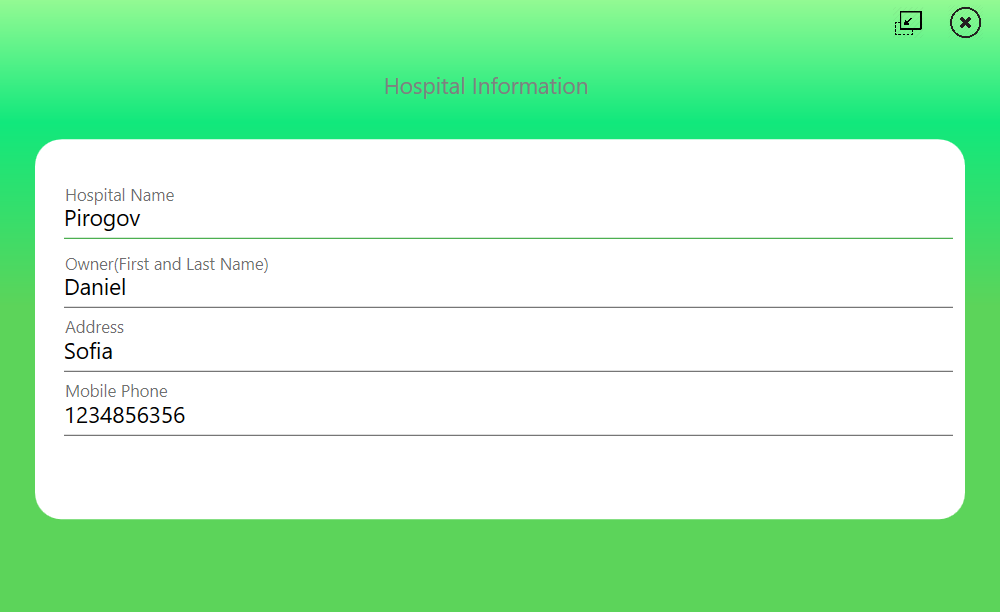


**Info Form**

Тази форма се използва за визуализация на различен тип информация, която е защитена от промяна. Всички полета, които са визуализирани са ReadOnly. Използва се визуализация на данни пациент, доктор, болница. Подава се съответно обект от тип Person – Доктор или Пациент или Болница, който съдържа определената информция за определеният потребител. В тази форма са представени следните данни за пациента/доктора – име, презиме, фамилия, години, адрес, телефон и специализация. За болницата – име, собственик, адрес и мобилен телефон. Използва се за по-дейтайлна информация на вече логналият се в системата потребител.



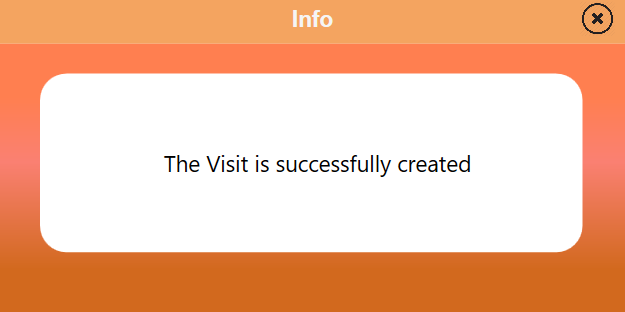
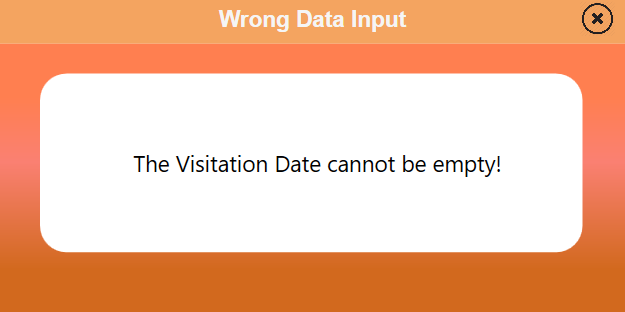
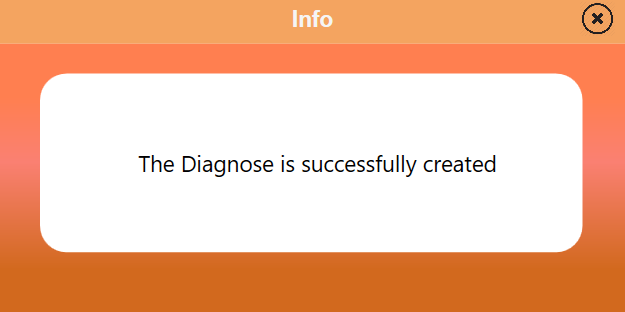


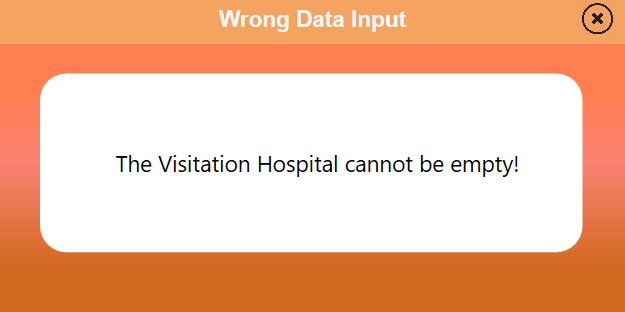


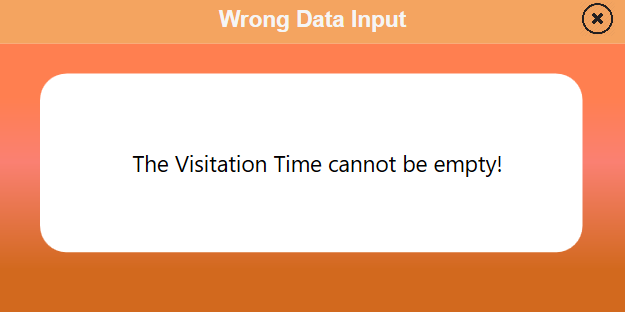
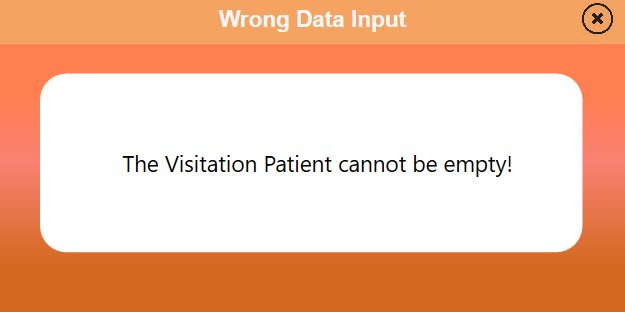
**Visitation Register**

Тази форма се използва създаваме не посещение на определен пациент при определен доктор, които работи в определена болница. Тук може да се избере съответно дата, час, пациент – всички налични пациенти в базата от данни - dropdown поле, доктор – работещите доктори в определена болница(ако формата е извикан от болничната форма) – dropdown поле, болница – всички налични болници, в които работи даден доктор(извикана от доктор формата) – dropdown поле и диагнозата, която му е била поставена. Вземането на нужните списъци с пациенти, доктори, болници се вземат чрез правенето на GET HTTP заявки, които се филтрират в зависимост от това от коя форма се извиква субформата. Когато потребителят се опита да добави ново посещение и ако има невалиден формат на данните или празно поле се визуализира се извежда съобщение за грешка. При кликането на бутона Save се прави POST HTTP заявка, която добавя новият запис в базата данни ако записът е успешен се визуализира съобщение, което информира потребителят, че операцията е успешна, в противен случай съобщение за грешка.





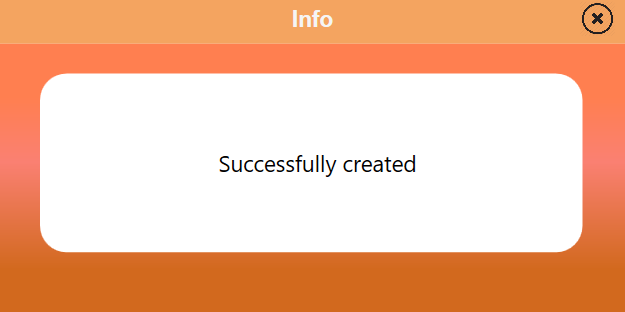
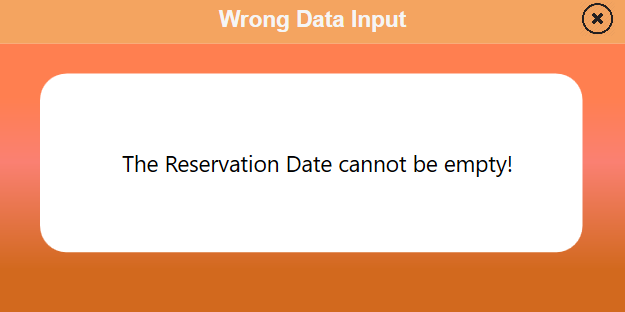
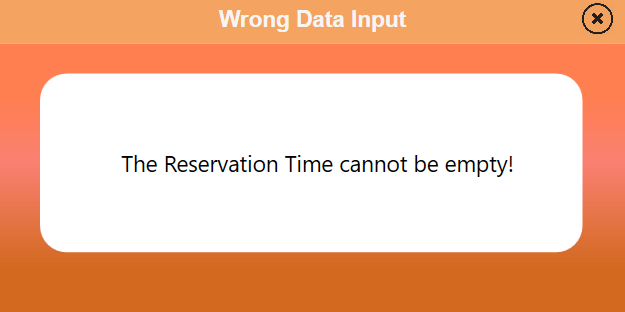
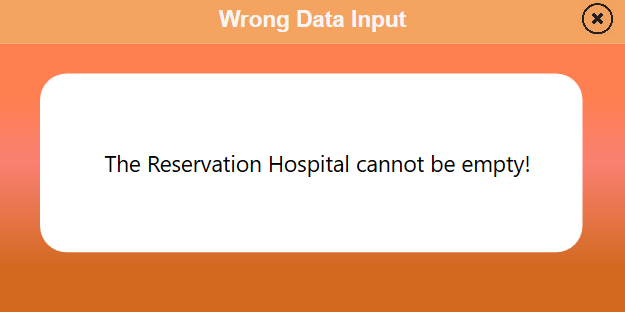
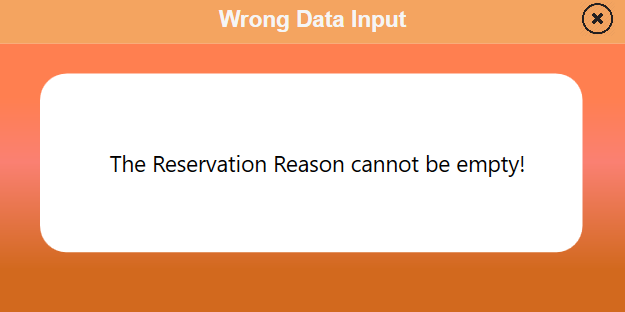
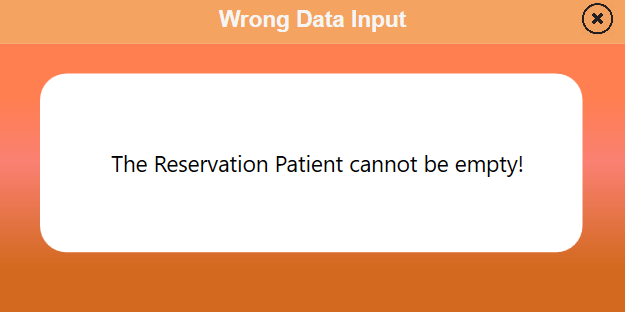




**Reservation Form**

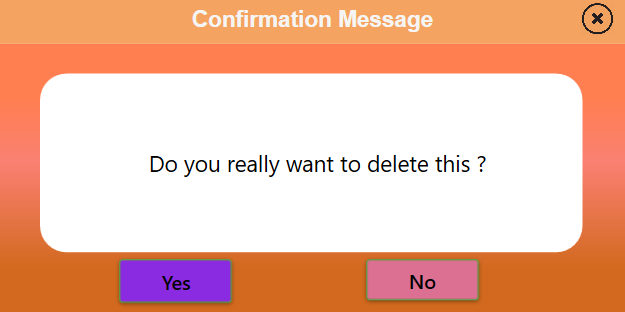
Тази форма се използва за създаванее нa резервация на определен пациент при определен доктор, които работи в определена болница. Тук може да се избере съответно дата, час, пациент – всички налични пациенти в базата от данни - dropdown поле, доктор – работещите доктори в определена болница(ако формата е извикан от болничната форма) – dropdown поле, болница – всички налични болници, в които работи даден доктор(извикана от доктор формата) – dropdown поле и причина , която му е била поставена. Вземането на нужните списъци с пациенти, доктори, болници се вземат чрез правенето на GET HTTP заявки, които се филтрират в зависимост от това от коя форма се извиква субформата. Когато потребителят се опита да добави нова резервация и ако има невалиден формат на данните или празно поле се визуализира се извежда съобщение за грешка. При кликането на бутона Save се прави POST HTTP заявка, която добавя новият запис в базата данни ако записът е успешен се визуализира съобщение, което информира потребителят, че операцията е успешна, в противен случай съобщение за грешка.

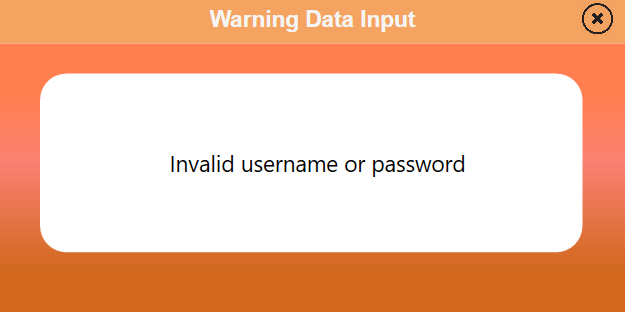


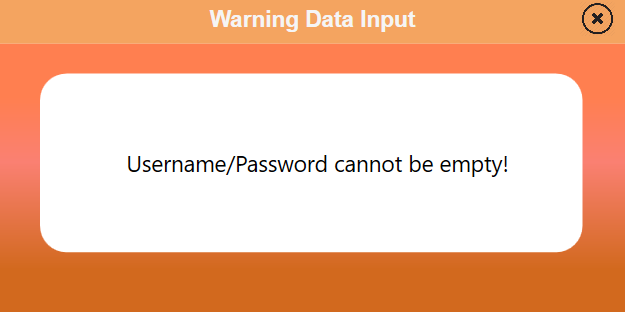


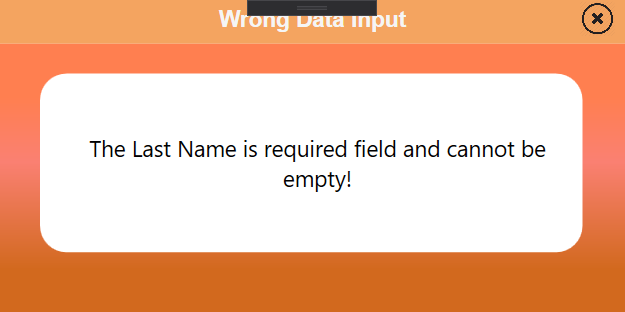
**Message Form**

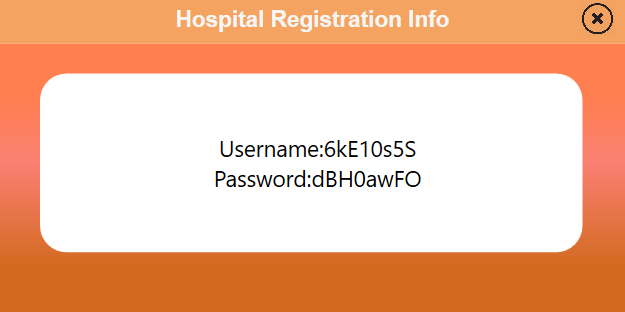
Tази форма е констрирана да бъде извиквана в разчини форми с цел да бъде визуализирано същение за грешка или съобщение, което информира потребителят, че дадена операция е успешна. Тази форма може да се използва и за визуализация на съобщения за потвръждение. Може да му се задават име на съобщението, текст, който да бъде визуализиран и също така какъв тип да бъде с потвръждение или без.

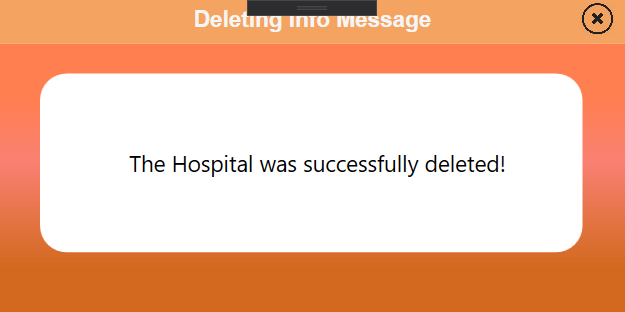












Класовете използвани в Client частта са следните:

1. **Person** – абстрактен клас, който следните property-та – Guid, Name, MiddleName, FamilyName, Age, Egn, Address, MobilePhone, FullName (Връща Name и FullName). Всичките пропъртита са от тип string, само годините са от тип int.
2. **Doctor** – наследява класът Person и има допълнително пропърти – Specilization от тип string;
3. **Patient** – наследява класът Person;
4. **Hospital** - има следните пропъртита - Guid, Name, Owner, Address, MobilePhone – всички те са от тип string.
5. **User** – има следните пропъртита – Guid, UserRole, AuthenticationCredentials, Username, Password – всички от тях са от тип тип string
6. **NewCreatedUser** – наследява User класа, като използва само Usename и Password пропъртитата.
7. **DiagnoseList** – този клас има следните пропъртита – Guid, DiagnosisDescription, които са от тип стринг, DiagnoseDateTime от тип DateTime и Doctor от тип Doctor.
8. **Reservation** – този клас има следните пропъртита - Guid, Reason, които са от тип стринг, ReservationDateTime от тип DateTime, Patient от тип Patient, Doctor от тип Doctor, Hospital от тип Hospital.
9. **Visit** – Guid от тип стринг, VisitTime от тип DateTime, Patient от тип Patient, Doctor от тип Doctor, Hospital от тип Hospital

Гореспоменатите класове се използват за съхранение и на получената JSON информация и мапването им от ViewModel - > View -то за съответната визуализирана форма на екрана.

1. **MessageBox** – този клас е статичен, клас, който клас има два статични метода за показване на съобщения за грешки и информационни съобщения.

* public static Action<string> ShowMessageBoxError(string MessageTitle, string MessageBody) – Извиква MessageForm- формат, за да визуализира съобщението за грешка на екрана. Връща Action – делегат, който върши нужната работа. Предимно се използва при връщането на грешки от HTTP завките. Ако няма никаква върната грешка от заявката се сетва default-на грешка, въведена от самият програмист.
* public static void ShowMessageBoxInfo(string MessageTitle, string MessageBody) - Извиква MessageForm- формат, за да визуализира информационно съобщение на екрана.

Пример:

ReservationList = HttpRequest.Of($"https://localhost:44348/api/reservations", RequestType.GET)

.BasicAuthentication(BasicAuthentication)

**.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))**

.Execute()

.DeserializeBody<ObservableCollection<Reservation>>();

**MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Middle Name is required field and cannot be empty!");**

1. JObjectBuilder – е клас, който използва Builder pattern за реализацията на JSON обект.

* public JObjectBuilder With(Action<dynamic> action) – в тази функция се подава анонимна функция(action), в която ще се дефинират пропъртитата, които ще съдържа JSON Обекта. Връща this, за да може бъде използвана много кратно тази функция
* public JObject Build() – тази функция връща JSON обекта след като вече сме приключили с добавянето на пропъртита.

Пример:

**POSTDoctorRequest(User.AuthenticationCredentials, new JObjectBuilder()**

**.With(doctor => doctor.Name = this.Name.Text)**

**.With(doctor => doctor.MiddleName = this.MiddleName.Text)**

**.With(doctor => doctor.FamilyName = this.FamilyName.Text)**

**.With(doctor => doctor.Egn = this.Egn.Text)**

**.With(doctor => doctor.MobilePhone = this.MobilePhone.Text)**

**.With(doctor => doctor.Address = this.Address.Text)**

**.With(doctor => doctor.Specialization = this.Specialization.Text)**

**.Build());**

1. **HttpRequest** – този клас се използва за създаването на различни HTTP заявки. В този клас отново е използван builder pattern (chain метод).

* public HttpRequest(string endpoint, RequestType requestType) – контрукторът може да приема endpoint-a(адреса, на който ще се намират определените ресурси) и типът на заявката която е от тип enum.

public enum RequestType

{

GET, POST, PUT, DELETE

}

* public static HttpRequest Of(string endpoint, RequestType requestType) – статичен метод, който позволява да се създаде инстанция на HttpRequest class-a, чрез извикването на конструктура му и връща отново HttpRequest.
* public HttpRequest ContentType(string contentType) – този метод позволя да се зададе content type-a на заявката.
* public HttpRequest Header(string header, string value) – задава се header-a и стойността му.
* public HttpRequest BasicAuthentication(string authentication) – задава се Basic Authentication, ако има такъв разбира се.
* public HttpRequest BodyData<T>(T value) – в този метод се се записва body-то, което ще съдържа заявката в зависимост от това дали е POST или PUT или DELETE. Value е Generic, за да може да се подават обекти от всякакъв тип.
* public HttpResponse Execute() – метод, който извършва заявката и при успешено изпълнение и върнат отговор се връща body, което съдържа необходимата информация (JSON) и статуса на заявката.Връща обект от тип HttpResponse, от който може да се вземат body-to и статуса на request-a. Ако всичко е успешно ще данните от body-то ще бъдат десериализирани и ще се продължи нормалното изпълнение на програмата, но противен случай ще бъде изписано съответното съобщение за грешка и стусът ще е неуспешен.
* private static bool IsSuccessful(HttpWebResponse response) – проверява дали статуса върнат от заявката е 200(ОК), 201(Created), 203(Accepted) или 204(No Content) и връща true ако един от тези статуси е върнат, в противен случай връща false.

1. **HttpResponse –** e клас, който отговаря за обработването на отговорът, който е върнал сървърт на клиента. Има статично пропърти, което връща HttpResponse с боди, което null, и статус Unsuccessful когато заявката не е успешна.

Пример:

public static HttpResponse Unsuccessful = new HttpResponse(null, HttpResponseStatus.Unsuccessful);

Този клас също така съдържа HttpResponseStatus пропърти, което се използва за задаването на статуса на заявката.

Пример:

private HttpResponseStatus HttpResponseStatus { get; set; }

В конструкторора на класа се сетват съдържанието(Body) върнато от заявката и нейният статус.

Пример:

-public HttpResponse(string body, HttpResponseStatus httpResponseStatus)

* public R DeserializeBody<R>() – се използва за десериализиране на полученто body от направената заявка. Той е Generic, за да може да се десериализират всякакви тип обекти. Връща зададеният му тип обект.
* public bool IsSuccessful() – този метод се използва за проверка на статуса дали е успешен или не. Ако е успешен връща true иначе false.

1. public static class WebRequestExtensions – extension class, който се използва за предотвратяване на избиване на грешки, когато се връща Http response без никакво body.

* public static WebResponse GetResponseWithoutException(this WebRequest request) – позволя програмата да продължи нормалният си процес, когато request-a не съдържа body. Връща Response-а, който е върнала заявката.

**Admin Form**

Тази форма съдържа няколко опции – Регистрация на пациент, доктор и болница. Същевременно с това може се визуализират списъци със всички пациенти, доктори и болници в зависимост на кой таб е кликнал потребителят. Информацията за различните табо-ве се контролира от partial класа, който е свързан с xaml file-a.

За визуализацията на табовете е използван следният xaml код:

<StackPanel Orientation="Horizontal" VerticalAlignment="Top" Margin="10,0,114.2,0">

<Button Uid="0" Width="145" Content="Register Patient" Height="50" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="Black" Click="Button\_Click"/>

<Button Uid="1" Width="145" Content="Register Doctor" Height="50" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="Black" Click="Button\_Click"/>

<Button Uid="2" Width="145" Content="Register Hospital" Height="50" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="Black" Click="Button\_Click"/>

</StackPanel>

При кликането на някой от табовете извиква функцията Button\_Click ,където се идентифицира, Uid-то на таба и в зависимост от това се визуализират определени DataGrid-ве и бутони съответно за регистрация на доктор,пациент и болница. В тази форма има съответно 3 ралични DataGrid-a, който представят различни списъци данни. Също така всеки DataGrid e има <DataGrid.CellStyle>, <DataGrid.ColumnHeaderStyle>, <DataGrid.RowStyle>, който правят таблицата transparent, нагласят се дължината,широчината на клетките, цвета а също така и разположението на текста в тях. Има опция при която при кликането върху името на доктор, пациент или болница се отваря нов инфо прозорец, който позволява да се види по-детайлна информация пациента, болницата или доктора. Този feature става доспъпен чрез въвеждането на

<EventSetter Event="MouseDoubleClick" Handler="DataGridRow\_MouseDoubleClick"> в <DataGrid.RowStyle>

Чрез съответна логика в xaml.cs файлът се индетифицира, коя форма трябва да бъде извикана дали за Info формата или Edit Form-та

private void DataGridRow\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

var isDoctorsListVisible = DoctorsList.Visibility == Visibility.Visible;

var isHospitalsListVisible = HospitalsList.Visibility == Visibility.Visible;

var index = isHospitalsListVisible ? HospitalsList.SelectedIndex : (isDoctorsListVisible ? DoctorsList.SelectedIndex : PatientsList.SelectedIndex);

var cell = isHospitalsListVisible ? HospitalsList.CurrentCell.Column.DisplayIndex : (isDoctorsListVisible ? DoctorsList.CurrentCell.Column.DisplayIndex : PatientsList.CurrentCell.Column.DisplayIndex);

InfoForm infoForm = null;

EditForm editForm = null;

if (cell == 0)

{

if (isHospitalsListVisible)

{

editForm = new EditForm(User, (object)Hospitals[index]);

}

else if (isDoctorsListVisible)

editForm = new EditForm(User, (object)Doctors[index]);

else

editForm = new EditForm(User, (object)Patients[index]);

}

else if (cell == 2)

{

if (isHospitalsListVisible)

infoForm = new InfoForm((object)Hospitals[index]);

else if (isDoctorsListVisible)

infoForm = new InfoForm((object)Doctors[index]);

else

infoForm = new InfoForm((object)Patients[index]);

}

else if (cell == 5)

{

if (isDoctorsListVisible)

infoForm = new InfoForm((object)Doctors[index]);

}

if (infoForm != null)

infoForm.ShowDialog();

if (editForm != null)

{

editForm.Closed += EditForm\_Closed;

editForm.ShowDialog();

}

}

Има info бутон, който позволява на админа да отвори някой запис от списъка и да направи промени, след като промяната е направена успешно се визуализира съобщение на екрана и след затварянето му вече в списъка може да се види направената промяна. Това става чрез следната функция

private void EditForm\_Closed(object sender, EventArgs e)

{

Patients = GETAllPatientsRequest(User.AuthenticationCredentials);

PatientsList.ItemsSource = Patients;

Doctors = GETAllDoctorsRequest(User.AuthenticationCredentials);

DoctorsList.ItemsSource = Doctors;

Hospitals = GETAllHospitalsRequest(User.AuthenticationCredentials);

HospitalsList.ItemsSource = Hospitals;

}

Чрез следните функции се контролира визуализацията на данните в различните табове:

private void Button\_Click(object sender, RoutedEventArgs e)

{

int currentIndex = int.Parse(((Button)e.Source).Uid);

if (currentIndex == 0)

{

GridCursor.Margin = new Thickness(20 + (145 \* currentIndex), 0, 0, 0);

}

else

{

GridCursor.Margin = new Thickness(25 + (145 \* currentIndex), 0, 0, 0);

}

DisplayCurrentTab(currentIndex);

}

public void CollapsePreviousTab(int index)

{

switch (index)

{

case 0:

PatientsList.Visibility = Visibility.Collapsed;

RegisterPatientBtn.Visibility = Visibility.Collapsed;

break;

case 1:

DoctorsList.Visibility = Visibility.Collapsed;

RegisterDoctorBtn.Visibility = Visibility.Collapsed;

break;

case 2:

HospitalsList.Visibility = Visibility.Collapsed;

RegisterHospitalBtn.Visibility = Visibility.Collapsed;

break;

}

}

public void DisplayCurrentTab(int index)

{

switch (index)

{

case 0:

CollapsePreviousTab(1);

CollapsePreviousTab(2);

Patients = GETAllPatientsRequest(User.AuthenticationCredentials);

PatientsList.ItemsSource = Patients;

PatientsList.Visibility = Visibility.Visible;

RegisterPatientBtn.Visibility = Visibility.Visible;

break;

case 1:

CollapsePreviousTab(0);

CollapsePreviousTab(2);

Doctors = GETAllDoctorsRequest(User.AuthenticationCredentials);

DoctorsList.ItemsSource = Doctors;

DoctorsList.Visibility = Visibility.Visible;

RegisterDoctorBtn.Visibility = Visibility.Visible;

break;

case 2:

CollapsePreviousTab(0);

CollapsePreviousTab(1);

Hospitals = GETAllHospitalsRequest(User.AuthenticationCredentials);

HospitalsList.ItemsSource = Hospitals;

HospitalsList.Visibility = Visibility.Visible;

RegisterHospitalBtn.Visibility = Visibility.Visible;

break;

}

}

Следната функция позволява да се местим прозореца по екрана.

private void Window\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (e.LeftButton == MouseButtonState.Pressed)

{

DragMove();

}

}

Тази функция се извиква при кликането на бутона Х приложението се затваря

private void ExitBtn\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

XAML връзка:

<Button x:Name="ExitBtn" Background="Transparent" BorderBrush="Transparent" HorizontalAlignment="Left" Margin="1009,0,-5.8,0" VerticalAlignment="Top" RenderTransformOrigin="-0.613,0.156" Width="60" Height="40" Click="ExitBtn\_Click">

<ContentControl>

<Image Source="../Assets/xButton.png" Height="28" RenderTransformOrigin="0.5,0.5" Width="55" />

</ContentControl>

</Button>

При кликането на бутона за минимизиране се ивиква следната функция:

private void MinimizeBtn\_Click(object sender, RoutedEventArgs e)

{

if (this.WindowState == WindowState.Normal)

{

this.WindowState = WindowState.Minimized;

}

else if (this.WindowState == WindowState.Minimized)

{

this.WindowState = WindowState.Normal;

}

}

XAML връзка:

<Button x:Name="minimizeBtn" Background="Transparent" BorderBrush="Transparent" HorizontalAlignment="Left" Margin="949,0,0,0" VerticalAlignment="Top" RenderTransformOrigin="-0.613,0.156" Width="55" Height="40" Click="MinimizeBtn\_Click" >

<ContentControl>

<Image Source="../Assets/minimize.png" Height="25" RenderTransformOrigin="0.5,0.5" Width="33" />

</ContentControl>

</Button>

При регистацията на нов доктор, пациент или болница се извикват следните методи при кликането съответно на бутоните – Register Patient, Register Doctor, Register Hospital

public void Register\_Patient\_Click(object sender, RoutedEventArgs e)

{

PatientRegister patientRegisterForm = new PatientRegister(User);

patientRegisterForm.Closed += PatientRegisterForm\_Closed;

patientRegisterForm.ShowDialog();

}

private void PatientRegisterForm\_Closed(object sender, EventArgs e)

{

Patients = GETAllPatientsRequest(User.AuthenticationCredentials);

PatientsList.ItemsSource = Patients;

}

XAML връзка:

<Button x:Name="RegisterPatientBtn" Visibility="Visible" Content="Register Patient" FontSize="18" Foreground="Black" Background="PaleVioletRed" HorizontalAlignment="Left" Margin="20,173.2,0,10" Grid.Row="2" Width="1030" Height="Auto" Click="Register\_Patient\_Click"/>

public void Register\_Doctor\_Click(object sender, RoutedEventArgs e)

{

DoctorRegister doctorRegisterForm = new DoctorRegister(User);

doctorRegisterForm.Closed += DoctorRegisterForm\_Closed;

doctorRegisterForm.ShowDialog();

}

private void DoctorRegisterForm\_Closed(object sender, EventArgs e)

{

Doctors = GETAllDoctorsRequest(User.AuthenticationCredentials);

DoctorsList.ItemsSource = Doctors;

}

XAML връзка:

<Button x:Name="RegisterDoctorBtn" Visibility="Collapsed" Content="Register Doctor" FontSize="18" Foreground="Black" Background="PaleVioletRed" HorizontalAlignment="Left" Margin="20,173.2,0,10" Grid.Row="2" Width="1030" Height="Auto" Click="Register\_Doctor\_Click"/>

public void Register\_Hospital\_Click(object sender, RoutedEventArgs e)

{

HospitalRegister hospitalRegisterForm = new HospitalRegister(User);

hospitalRegisterForm.Closed += HospitalRegisterForm\_Closed;

hospitalRegisterForm.ShowDialog();

}

private void HospitalRegisterForm\_Closed(object sender, EventArgs e)

{

Hospitals = GETAllHospitalsRequest(User.AuthenticationCredentials);

HospitalsList.ItemsSource = Hospitals;

}

XAML връзка:

<Button x:Name="RegisterHospitalBtn" Visibility="Collapsed" Content="Register Hospital" FontSize="18" Foreground="Black" Background="PaleVioletRed" HorizontalAlignment="Left" Margin="20,173.2,0,10" Grid.Row="2" Width="1030" Height="Auto" Click="Register\_Hospital\_Click"/>

За вземането и визуализацията на необходите данни се извикват съответните методи, които правят GET заявки. Данните се съхраняват в обекти от тип ОbservableCollection и се подават на ItemSource-a на всеки DataGrid чрез Binding property на <DataGridTextColumn> се свързват и визуализират в листа от записи .

Пример:

private ObservableCollection<Doctor> GETAllDoctorsRequest(string BasicAuthentication)

{

return HttpRequest.Of("https://localhost:44348/api/doctors", RequestType.GET)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute()

.DeserializeBody<ObservableCollection<Doctor>>();

}

private ObservableCollection<Patient> GETAllPatientsRequest(string BasicAuthentication)

{

return HttpRequest.Of("https://localhost:44348/api/patients", RequestType.GET)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute()

.DeserializeBody<ObservableCollection<Patient>>();

}

private ObservableCollection<Hospital> GETAllHospitalsRequest(string BasicAuthentication)

{

return HttpRequest.Of("https://localhost:44348/api/hospitals", RequestType.GET)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute()

.DeserializeBody<ObservableCollection<Hospital>>();

}

Patients = GETAllPatientsRequest(User.AuthenticationCredentials);

PatientsList.ItemsSource = Patients;

Doctors = GETAllDoctorsRequest(User.AuthenticationCredentials);

DoctorsList.ItemsSource = Doctors;

Hospitals = GETAllHospitalsRequest(User.AuthenticationCredentials);

HospitalsList.ItemsSource = Hospitals;

<DataGridTemplateColumn Header="" Width="SizeToCells" IsReadOnly="True" MinWidth="50" >

<DataGridTemplateColumn.CellTemplate>

<DataTemplate >

<Image Source="{Binding Source=../Assets/infoIcon.png}" Width="30" Height="30"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Header="Doctor" FontSize="14" Binding="{Binding Path=FullName}" MinWidth="300" >

<DataGridTextColumn.ElementStyle>

<Style TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Header="Mobile Phone" FontSize="14" Binding="{Binding Path=MobilePhone}" MinWidth="200" >

<DataGridTextColumn.ElementStyle>

<Style TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Header="Specialization" FontSize="14" Binding="{Binding Path=Specialization}" MinWidth="300" >

<DataGridTextColumn.ElementStyle>

<Style TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

Аналогично е и за другите два таба в Аdmin формата.

**Doctor Form**

Tази форма е аналогична на Admin формата. Една от разликите в тази форма е, че докторът може да промени данните за себе си, ако са били сгрешени при създаването на профила му. Докторът има възможност да създадва, редактира и изтрива посещения на пациенти. Също така може да създадва, редактира и изтрива резервации за определен пациент. Другата опция, която има доктора е създаването она пациент.

За създадаването на ново посещение се клика върху бутона Аdd Visitation – този бутон отваря нова форма за създаване на ново посещение. След успешно добавя не се актуализира листа с посещения.

private void VisitBtn\_Click(object sender, RoutedEventArgs e)

{

var VisitationForm = new VisitationRegisterForm(User, "DoctorPSForm");

VisitationForm.Closed += VisitationForm\_Closed;

VisitationForm.ShowDialog();

}

private void VisitationForm\_Closed(object sender, EventArgs e)

{

this.PatientVisitation.ItemsSource = GETDoctorVisitsRequest(User.AuthenticationCredentials);

}

XAML връзка:

<Button x:Name="VisitBtn" Visibility="Collapsed" Content="Add Visitation" FontSize="18" Foreground="Black" Background="PaleVioletRed" HorizontalAlignment="Left" Margin="22,177.2,0,10" Width="1027" Grid.Row="2" Click="VisitBtn\_Click"/>

За създадаването на нова резервация се клика върху бутона Make a reservation – този бутон отваря нова форма за създаване на нова резервация . След успешно добавяне се актуализира листа с резервации.

private void ReserveBtn\_Click(object sender, RoutedEventArgs e)

{

var ReservationForm = new ReservationForm(User, "DoctorReservationForm");

ReservationForm.Closed += ReservationForm\_Closed;

ReservationForm.ShowDialog();

}

private void ReservationForm\_Closed(object sender, EventArgs e)

{

this.Reservations.ItemsSource = GETDoctorReservationsRequest(User.AuthenticationCredentials);

}

XAML връзка:

<Button x:Name="ReserveBtn" Visibility="Collapsed" Content="Make a reservation" FontSize="18" Foreground="Black" Background="PaleVioletRed" HorizontalAlignment="Left" Margin="10,177,0,10" Width="1053" Grid.Row="2" Click="ReserveBtn\_Click"/>

Аналогично както в Admin формата се вземат необходимите данни, които са визуализирани в листовете в различните табове, чрез GET http заявки.

Коригирането на личните данни в базата данни става с PUT заявка:

private void PUTDoctorInfoRequest(string param, string BasicAuthentication)

{

var response = HttpRequest.Of($"https://localhost:44348/api/doctors/{param}", RequestType.PUT)

.BasicAuthentication(BasicAuthentication)

.BodyData(Doctor)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", "Updating of the Doctor Info cannot be done!"))

.Execute();

if (response != null)

{

MessageBox.ShowMessageBoxInfo("Doctor Update Info", "Doctor Information was updated successfully!");

}

}

Всички полета за лична информация на доктора са валидирани и при невалидно въведени данни се визуализира съобщение за грешка. Същото така при промяна на някое от полетата има property Doctor, в което се актуализират нововъдедените вече валидирани данни и при натискането на бутона Save се извиква **PUTDoctorInfoRequest** метода.

Пример:

private bool ValidateDoctorEGN()

{

if (Egn.Text.ToString().Length != 10)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Doctor's EGN should be excatly 10 digits!");

return false;

}

else if (!Egn.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Doctor's EGN should contains only digts!");

return false;

}

return true;

}

private bool ValidateDoctorFirstName()

{

if (Name.Text.Length == 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The First Name is required field and cannot be empty!");

return false;

}

else if (Name.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The First Name cannot contain digits!");

return false;

}

return true;

}

private bool ValidateDoctorMiddleName()

{

if (MiddleName.Text.Length == 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Middle Name is required field and cannot be empty!");

return false;

}

else if (MiddleName.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Middle Name cannot contain digits!");

return false;

}

return true;

}

private bool ValidateDoctorLastName()

{

if (FamilyName.Text.Length == 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Last Name is required field and cannot be empty!");

return false;

}

else if (FamilyName.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Last Name cannot contain digits!");

return false;

}

return true;

}

private bool ValidateDoctorMobilePhone()

{

if (MobilePhone.Text.Length != 10)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Mobile Phone cannot be more or less than 10 digits!");

return false;

}

else if (!MobilePhone.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Mobile Phone can contain only digits!");

return false;

}

return true;

}

private bool ValidateDoctorAddress()

{

if (Address.Text.Length <= 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Address cannot be empty!");

return false;

}

return true;

}

private bool ValidateDoctorSpecialization()

{

if (Specialization.Text.Length <= 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Specialization cannot be more or less than 10 digits!");

return false;

}

return true;

}

private void SaveBtn\_Click(object sender, RoutedEventArgs e)

{

if (!ValidateDoctorFirstName())

return;

else if (!ValidateDoctorMiddleName())

return;

else if (!ValidateDoctorLastName())

return;

else if (!ValidateDoctorEGN())

return;

else if (!ValidateDoctorMobilePhone())

return;

else if (!ValidateDoctorAddress())

return;

else

{

PUTDoctorInfoRequest(User.Guid, User.AuthenticationCredentials);

Doctor = GETDoctorInfoRequest(User.Guid, User.AuthenticationCredentials);

}

}

private void Name\_TextChanged(object sender, TextChangedEventArgs e)

{

Doctor.Name = this.Name.Text;

}

private void MiddleName\_TextChanged(object sender, TextChangedEventArgs e)

{

Doctor.MiddleName = this.MiddleName.Text;

}

private void FamilyName\_TextChanged(object sender, TextChangedEventArgs e)

{

Doctor.FamilyName = this.FamilyName.Text;

}

private void Egn\_TextChanged(object sender, TextChangedEventArgs e)

{

Doctor.Egn = this.Egn.Text;

}

private void Address\_TextChanged(object sender, TextChangedEventArgs e)

{

Doctor.Address = this.Address.Text;

}

private void MobilePhone\_TextChanged(object sender, TextChangedEventArgs e)

{

Doctor.MobilePhone = this.MobilePhone.Text;

}

private void Specialization\_TextChanged(object sender, TextChangedEventArgs e)

{

Doctor.Specialization = this.Specialization.Text;

}

private void Name\_LostFocus(object sender, RoutedEventArgs e)

{

ValidateDoctorFirstName();

}

private void MiddleName\_LostFocus(object sender, RoutedEventArgs e)

{

ValidateDoctorMiddleName();

}

private void FamilyName\_LostFocus(object sender, RoutedEventArgs e)

{

ValidateDoctorLastName();

}

private void Egn\_LostFocus(object sender, RoutedEventArgs e)

{

ValidateDoctorEGN();

}

private void Address\_LostFocus(object sender, RoutedEventArgs e)

{

ValidateDoctorAddress();

}

private void MobilePhone\_LostFocus(object sender, RoutedEventArgs e)

{

ValidateDoctorMobilePhone();

}

private void Specialization\_LostFocus(object sender, RoutedEventArgs e)

{

ValidateDoctorSpecialization();

}

}

Patient Form

Тази форма е аналогична на формата за доктор. В тази форма потребителят с роля пациент може да коригира само своите лични данни и при кликане на save бутона да се актуализират в базата от данни. Тук също има валидация за всяко едно от полетата за лична информация на пациента и при успешно предмината валидация, промените се се актуализират в пропърти Patient, което държи текущите промени(лични данни) за пациента. Кодът е аналогичен както при формата за доктор :

private void saveBtn\_Click(object sender, RoutedEventArgs e)

{

if (!ValidatePatientFirstName())

return;

else if (!ValidatePatientMiddleName())

return;

else if (!ValidatePatientLastName())

return;

else if (!ValidatePatientEGN())

return;

else if (!ValidatePatientAge())

return;

else if (!ValidatePatientMobilePhone())

return;

else if (!ValidatePatientAddress())

return;

else

{

PUTPatientInfoRequest(User.Guid, User.AuthenticationCredentials);

patient = GETPatientInfoRequest(User.Guid, User.AuthenticationCredentials);

}

}

XAML връзка:

<Button x:Name="saveBtn" Content="Save" FontSize="18" Foreground="Black" Background="BlueViolet" HorizontalAlignment="Left" Margin="39,477,-477,10" Width="980" Click="saveBtn\_Click"/>

private bool ValidatePatientEGN()

{

if (Egn.Text.ToString().Length != 10)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Doctor's EGN should be excatly 10 digits!");

return false;

}

else if (!Egn.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Doctor's EGN should contains only digts!");

return false;

}

return true;

}

private bool ValidatePatientAge()

{

if (Age.Text.Length > 3 || Age.Text.Length < 1)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Age cannot be more that 3 or less than 1 digits!");

return false;

}

else if (!Age.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Age can contain only digits!");

return false;

}

return true;

}

private bool ValidatePatientFirstName()

{

if (Name.Text.Length == 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The First Name is required field and cannot be empty!");

return false;

}

else if (Name.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The First Name cannot contain digits!");

return false;

}

return true;

}

private bool ValidatePatientMiddleName()

{

if (MiddleName.Text.Length == 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Middle Name is required field and cannot be empty!");

return false;

}

else if (MiddleName.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Middle Name cannot contain digits!");

return false;

}

return true;

}

private bool ValidatePatientLastName()

{

if (FamilyName.Text.Length == 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Last Name is required field and cannot be empty!");

return false;

}

else if (FamilyName.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Last Name cannot contain digits!");

return false;

}

return true;

}

private bool ValidatePatientMobilePhone()

{

if (MobilePhone.Text.Length != 10)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Mobile Phone cannot be more or less than 10 digits!");

return false;

}

else if (!MobilePhone.Text.ToString().All(c => char.IsDigit(c)))

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Mobile Phone can contain only digits!");

return false;

}

return true;

}

private bool ValidatePatientAddress()

{

if (Address.Text.Length <= 0)

{

MessageBox.ShowMessageBoxInfo("Wrong Data Input", "The Address cannot be empty!");

return false;

}

return true;

}

private void EGN\_LostFocus(object sender, RoutedEventArgs e)

{

ValidatePatientEGN();

}

private void FirstName\_LostFocus(object sender, RoutedEventArgs e)

{

ValidatePatientFirstName();

}

private void MiddleName\_LostFocus(object sender, RoutedEventArgs e)

{

ValidatePatientMiddleName();

}

private void LastName\_LostFocus(object sender, RoutedEventArgs e)

{

ValidatePatientLastName();

}

private void Age\_LostFocus(object sender, RoutedEventArgs e)

{

ValidatePatientAge();

}

private void MobilePhone\_LostFocus(object sender, RoutedEventArgs e)

{

ValidatePatientMobilePhone();

}

private void Name\_TextChanged(object sender, TextChangedEventArgs e)

{

patient.Name = this.Name.Text;

}

private void MiddleName\_TextChanged(object sender, TextChangedEventArgs e)

{

patient.MiddleName = this.MiddleName.Text;

}

private void LastName\_TextChanged(object sender, TextChangedEventArgs e)

{

patient.FamilyName = this.FamilyName.Text;

}

private void Egn\_TextChanged(object sender, TextChangedEventArgs e)

{

patient.Egn = this.Egn.Text;

}

private void Age\_TextChanged(object sender, TextChangedEventArgs e)

{

patient.Age = Int32.Parse(this.Age.Text);

}

private void MobilePhone\_TextChanged(object sender, TextChangedEventArgs e)

{

patient.MobilePhone = this.MobilePhone.Text;

}

private void Address\_TextChanged(object sender, TextChangedEventArgs e)

{

patient.Address = this.Address.Text;

}

Пациентът мoже да вижда вече направените при съответният доктор и в съответната болница прегледи. Може да види кой доктор какви диагнози му е поставил и кога. Също така може да види вече направените резервации при определн доктор и определена болница. Има и таб, който показва всички доктори, който са регистрирани в системата. Когато потребителят кликне върху името на някой доктор или болница се отваря прозорец, който визуализира по-подробна информация съответно за болницата или доктора.

private void DataGridRow\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

var isVisitsVisible = Visits.Visibility == Visibility.Visible;

var isDiagnosesVisible = Diagnoses.Visibility == Visibility.Visible;

var isDoctorsVisible = Doctors.Visibility == Visibility.Visible;

var index = isDiagnosesVisible ? Diagnoses.SelectedIndex : (isDoctorsVisible ? Doctors.SelectedIndex : (isVisitsVisible ? Visits.SelectedIndex : Reservations.SelectedIndex));

var cell = isDiagnosesVisible ? Diagnoses.CurrentCell.Column.DisplayIndex : (isDoctorsVisible ? Doctors.CurrentCell.Column.DisplayIndex : (isVisitsVisible ? Visits.CurrentCell.Column.DisplayIndex : Reservations.CurrentCell.Column.DisplayIndex));

InfoForm infoForm = null;

if (cell == 0)

{

if (isDoctorsVisible)

infoForm = new InfoForm((object)DoctorList[index]);

}

else if (cell == 1)

{

if (isDiagnosesVisible)

infoForm = new InfoForm((object)DiagnosesList[index].Doctor);

else if (isVisitsVisible)

infoForm = new InfoForm((object)VisitList[index].Doctor);

else if (!isVisitsVisible && !isDiagnosesVisible && !isDoctorsVisible)

infoForm = new InfoForm((object)ReservationLlist[index].Doctor);

}

else if (cell == 2)

{

if (isVisitsVisible)

infoForm = new InfoForm((object)VisitList[index].Hospital);

else if (!isVisitsVisible && !isDiagnosesVisible && !isDoctorsVisible)

infoForm = new InfoForm((object)ReservationLlist[index].Hospital);

}

if (infoForm != null)

infoForm.ShowDialog();

}

Смяната на табовете и визуализацията на данните става аналогично както при Doctor Form.

**Hospital Form**

Формата за болницата е аналогична на формите за доктор. Тази форма позволява на потребителят с роля болница да вижда всички направени посещения на пациенти в нея и при кой доктор кога са били на посещение, може да коригира определените записи за посещения на определените пациенти. Може да вижда, кои доктори работя в нея.

Има опция за правенето на нова резервация при определен доктор. Това става аналогично както в доктор формата. Тук Болницата може да регистрира нов доктор в системата, като автоматично го добавя към нея на работа. Болницата може да премахва докторите, които работят в нея без да премахват техните акаути от системата.

Болницата също има възможност да коригира своята лична информация. Това отново става аналогично както доктор и пациент формата. Отново има направена валидация за всички полета, които могат да бъдат променяни от болницата.

Различното от другите две форми е, че има опция за добавяне на вече съществуващ в системата доктор към определената болница. При кликането на бутона Аdd Doctor се отваря форма, която визуализира лист със всички налични доктори в системата и болницата, може да си добави доктор, който ще работи в нея. След успешното добавяне листа се актуализира.

private ObservableCollection<Doctor> GETAllDoctorsRequest(string BasicAuthentication)

{

return HttpRequest.Of("https://localhost:44348/api/doctors", RequestType.GET)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute()

.DeserializeBody<ObservableCollection<Doctor>>();

}

private ObservableCollection<Doctor> GETDoctorsForHospital(string BaseAuthentication, string hospitalId)

{

List<Doctor> allDoctors = GETAllDoctorsRequest(BaseAuthentication).ToList();

List<String> doctorIds = GETHospitalDoctorsRequest(BaseAuthentication, hospitalId);

return new ObservableCollection<Doctor>(allDoctors.Where(doctor => doctorIds.Contains(doctor.Guid)).ToList());

}

private List<String> GETHospitalDoctorsRequest(string BasicAuthentication, string hospitalId)

{

return HttpRequest.Of($"https://localhost:44348/api/hospitals/{hospitalId}/doctors", RequestType.GET)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute()

.DeserializeBody<List<String>>();

}

private void AddDoctorBtn\_Click(object sender, RoutedEventArgs e)

{

DoctorsList DoctorsList = new DoctorsList(User, DoctorList);

DoctorsList.ShowDialog();

this.Doctors.ItemsSource = GETDoctorsForHospital(User.AuthenticationCredentials, User.Guid);

}

XAML връзка:

<Button x:Name="AddDoctorBtn" Content="Add Doctor" FontSize="18" Visibility="Collapsed" Foreground="Black" Background="BlueViolet" HorizontalAlignment="Left" Margin="46,172.2,0,10" Width="980" Grid.Row="2" Height="Auto" Click="AddDoctorBtn\_Click"/>

**PatiеntRegister, DoctorRegister, HospitalRegister** формите са индентични. Полета, който се попълват са от тип TextBox , пакетирани в StackPanel, който разположен в Grid.

При тях всяко от полетата е валидирано и при въвеждане на невалидни данни се изписва съобщение, което информира потребителя какъв трябва да бъде направеният от него input. И при трите форми затварянето, минимизирането и преместването на прозореца става по аналогичен начин както при админ формата.

Doctor Register Form (XAML код) :

<Grid Width="750" Height="623" VerticalAlignment="Bottom" Margin="10,0,-40,-101">

<TextBlock Text="Doctor Register " HorizontalAlignment="Center" VerticalAlignment="Top" Margin="294,-168,327,0" Foreground="Gray" FontSize="18" Height="28"/>

<StackPanel VerticalAlignment="Center" Margin="32,-104,54.4,270" Height="457" Grid.ColumnSpan="2">

<TextBox x:Name="Name" materialDesign:HintAssist.Hint="First Name" Height="55" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,52,0" LostFocus="FirstName\_LostFocus"/>

<TextBox x:Name="MiddleName" materialDesign:HintAssist.Hint="Second Name" Height="59" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,52,0" LostFocus="SecondName\_LostFocus"/>

<TextBox x:Name="FamilyName" materialDesign:HintAssist.Hint="Last Name" Height="57" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,52,0" LostFocus="LastName\_LostFocus"/>

<TextBox x:Name="Egn" materialDesign:HintAssist.Hint="EGN" Height="52" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" HorizontalAlignment="Left" Width="612" LostFocus="EGN\_LostFocus"/>

<TextBox x:Name="Address" materialDesign:HintAssist.Hint="Address" Height="52" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,52,0"/>

<TextBox x:Name="MobilePhone" materialDesign:HintAssist.Hint="Mobile Phone" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,52,0" LostFocus="MobilePhone\_LostFocus"/>

<TextBox x:Name="Specialization" materialDesign:HintAssist.Hint="Specialization" TextWrapping="Wrap" TextAlignment="Left" VerticalContentAlignment="Top" AcceptsReturn="True" HorizontalScrollBarVisibility="Disabled" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontSize="18" VerticalScrollBarVisibility="Auto" Margin="0,0,52,0" Height="54" />

</StackPanel>

</Grid>

Hospital Register Form (XAML код) :

<Grid Width="750" Height="623" VerticalAlignment="Bottom" Margin="25,0,25,10">

<TextBlock Text="Hospital Register " HorizontalAlignment="Center" VerticalAlignment="Top" Margin="304,135,296,0" Foreground="Gray" FontSize="18" Height="30" Width="150"/>

<StackPanel VerticalAlignment="Center" Margin="20,221,0,122" Height="280">

<TextBox x:Name="Name" materialDesign:HintAssist.Hint="Hospital Name" Height="55" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" LostFocus="HospitalName\_LostFocus"/>

<TextBox x:Name="Owner" materialDesign:HintAssist.Hint="Owner(First and Last Name)" Height="55" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" LostFocus="HospitalOwner\_LostFocus"/>

<TextBox x:Name="Address" materialDesign:HintAssist.Hint="Address" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<TextBox x:Name="MobilePhone" materialDesign:HintAssist.Hint="MobilePhone" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

</StackPanel>

</Grid>

Когато се натисне Register button-a се прави POST заявка която създава нов запис в базата данни.

private void POSTHospitalRequest(string BasicAuthentication, JObject Hospital)

{

var createdUser = HttpRequest.Of("https://localhost:44348/api/hospitals", RequestType.POST)

.BasicAuthentication(BasicAuthentication)

.BodyData(Hospital)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute()

.DeserializeBody<NewCreatedUser>();

if (createdUser != null)

{

MessageBox.ShowMessageBoxInfo("Hospital Registration Info", $"Username:{createdUser.Username}\nPassword:{createdUser.Password}");

}

}

private void Register\_Click(object sender, RoutedEventArgs e)

{

if (!ValidateHospitalName())

return;

else if (!ValidateHospitalOwner())

return;

else if (!ValidateHospitalMobilePhone())

return;

else if (!ValidateHospitalAddress())

return;

else

{

POSTHospitalRequest(User.AuthenticationCredentials, new JObjectBuilder().With(hospital => hospital.Name = this.Name.Text)

.With(hospital => hospital.Owner = this.Owner.Text)

.With(hospital => hospital.MobilePhone = this.MobilePhone.Text)

.With(hospital => hospital.Address = this.Address.Text)

.Build());

this.Close();

}

XAML връзка:

<Button x:Name="RegisterBtn" VerticalAlignment="Bottom" Margin="299,0,301,105" HorizontalAlignment="Center" Width="200" Content="Register" Background="#FF30F130" FontSize="18" Foreground="Black" RenderTransformOrigin="0.504,-1.387" Click="Register\_Click" ></Button>

private void POSTDoctorRequest(string BasicAuthentication, JObject Doctor)

{

var createdUser = HttpRequest.Of("https://localhost:44348/api/doctors", RequestType.POST)

.BasicAuthentication(BasicAuthentication)

.BodyData(Doctor)

.OnError(MessageBox.ShowMessageBoxError("Wrong Data Input", ""))

.Execute()

.DeserializeBody<NewCreatedUser>();

if (createdUser != null)

{

MessageBox.ShowMessageBoxInfo("Doctor Registration Info", $"Username:{createdUser.Username}\nPassword:{createdUser.Password}");

}

}

private void RegisterBtn\_Click(object sender, RoutedEventArgs e)

{

if (!ValidateDoctorFirstName())

return;

else if (!ValidateDoctorMiddleName())

return;

else if (!ValidateDoctorLastName())

return;

else if (!ValidateDoctorPhone())

return;

else if (!ValidateDoctorEGN())

return;

else if (!ValidateDoctorAddress())

return;

else if (!ValidateDoctorSpecialization())

return;

else

{

POSTDoctorRequest(User.AuthenticationCredentials, new JObjectBuilder().With(doctor => doctor.Name = this.Name.Text)

.With(doctor => doctor.MiddleName = this.MiddleName.Text)

.With(doctor => doctor.FamilyName = this.FamilyName.Text)

.With(doctor => doctor.Egn = this.Egn.Text)

.With(doctor => doctor.MobilePhone = this.MobilePhone.Text)

.With(doctor => doctor.Address = this.Address.Text)

.With(doctor => doctor.Specialization = this.Specialization.Text)

.Build());

this.Close();

}

XAML връзка:

<Button x:Name="RegisterBtn" VerticalAlignment="Bottom" Margin="251,0,269,75" HorizontalAlignment="Center" Width="200" Content="Register" Background="#FF30F130" FontSize="18" Foreground="Black" Click="RegisterBtn\_Click" ></Button>

**DoctorsList**

Тази форма е проектирана да показва списък със всички доктори. Визуализацията става чрез DataGrid, a списъкът с доктори се подава като параметър при извикване на формата.

Когато се кликне върху определен доктор той се прави POST заявка, която добавя доктора към определена болница. Това се определя от потребителя, който също се подава като параметър при извикването на тази форма. Дизайнът на прозорецът е същият както при админ формата.

XAML форма :

<TextBox Text="Doctors List" Foreground="Gray" FontSize="20" VerticalAlignment="Top" HorizontalAlignment="Center" Margin="452,10,430,0" Width="118"></TextBox>

<DataGrid x:Name="Doctors" AutoGenerateColumns="False" Visibility="Visible" Height="533" HorizontalAlignment="Center" VerticalAlignment="Top" Width="1000" Background="Transparent" BorderBrush="Transparent" Margin="0,67,0,0"

ScrollViewer.CanContentScroll="True" ScrollViewer.VerticalScrollBarVisibility="Auto" ScrollViewer.HorizontalScrollBarVisibility="Hidden" IsReadOnly="True" materialDesign:ScrollBarAssist.ThumbHeight="300" RenderTransformOrigin="0.462,-0.162">

<DataGrid.ColumnHeaderStyle>

<Style TargetType="DataGridColumnHeader" >

<Setter Property="FontSize" Value="16"/>

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderBrush" Value="Transparent"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

</Style>

</DataGrid.ColumnHeaderStyle>

<DataGrid.RowStyle>

<Style TargetType="DataGridRow">

<Setter Property="FontSize" Value="16"/>

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderBrush" Value="Transparent"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

<EventSetter Event="MouseDoubleClick" Handler="DataGridRow\_MouseDoubleClick"/>

</Style>

</DataGrid.RowStyle>

<DataGrid.CellStyle>

<Style TargetType="DataGridCell">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderBrush" Value="Transparent"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

</Style>

</DataGrid.CellStyle>

<DataGrid.Columns >

<DataGridTextColumn Header="Full Name" FontSize="16" Binding="{Binding Path=FullName}" MinWidth="335" >

<DataGridTextColumn.ElementStyle>

<Style TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Header="Specialization" FontSize="16" Binding="{Binding Path=Specialization}" MinWidth="335" >

<DataGridTextColumn.ElementStyle>

<Style TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Header="Mobile Phone" FontSize="16" Binding="{Binding Path=MobilePhone}" MinWidth="335" >

<DataGridTextColumn.ElementStyle>

<Style TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

Връзката с xaml.cs класа при правенето на HTTP заявката:

private void POSTDoctorInfoRequest(string param, string BasicAuthentication, Doctor Doctor)

{

var httpResponse = HttpRequest.Of($"https://localhost:44348/api/hospitals/{param}/doctors", RequestType.POST)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", "Adding of Doctor cannot be done!"))

.BodyData(Doctor.Guid)

.Execute();

if (httpResponse.IsSuccessful())

{

MessageBox.ShowMessageBoxInfo("Doctor Creation Info", "The Doctor was added successfully!");

}

}

private void DataGridRow\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

POSTDoctorInfoRequest(User.Guid, User.AuthenticationCredentials, HospitalDoctors[Doctors.SelectedIndex]);

this.Close();

}

**Edit Form**

Тази форма е проектирана да визуализира разлини инофрмация в зависимост от коя форма е извикана. Позволява да се коригират или изтриват записи. При извикването и се подава потребителят, който я извиква и също така обекът, който трябва да се променя. Проверява се от какъв тип е обекта и кой е потребителят, който иска да направи промяната. На базата на тези две ограничения се визуализират раличните Grid-ве и StackPanel-ли, който държата информацията за определеният запис, който трябва да бъде показан на екрана. Мапването на данните отново става чрез използването на пропъртито Binding в TextBox атрибута.

Пример:

<Grid x:Name="PatientReservationsForm" Height="340" VerticalAlignment="Bottom" Margin="25,0,25,8" Visibility="Collapsed">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="196\*"/>

<ColumnDefinition Width="179\*"/>

</Grid.ColumnDefinitions>

<TextBlock x:Name="PatientReservationTitle" Text="Edit Reservation Record" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="262,-144,256,0" Foreground="Gray" FontSize="18" Height="28" Grid.ColumnSpan="2" Width="232"></TextBlock>

<StackPanel VerticalAlignment="Center" Margin="10,-32,10,124" Height="248" Grid.ColumnSpan="2">

<DatePicker x:Name="PatientReservationsDate" Text="{Binding Path=PatientReservationProfile.Date}" materialDesign:HintAssist.Hint="Reservation Date" Height="79" Style="{StaticResource MaterialDesignFloatingHintDatePicker}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<materialDesign:TimePicker x:Name="PatientReservationsTime" Style="{StaticResource MaterialDesignFloatingHintTimePicker}" materialDesign:HintAssist.Hint="Reservation Time" Height="79" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

</StackPanel>

</Grid>

<Grid x:Name="HospitalReservationForm" Height="340" VerticalAlignment="Bottom" Margin="25,0,25,8" Visibility="Collapsed">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="196\*"/>

<ColumnDefinition Width="179\*"/>

</Grid.ColumnDefinitions>

<TextBlock x:Name="HospitalReservationTitle" Text="Edit Hospital Reservation" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="270,-142,278,0" Foreground="Gray" FontSize="18" Height="28" Grid.ColumnSpan="2" Width="202"></TextBlock>

<StackPanel VerticalAlignment="Center" Margin="10,-48,10,140" Height="248" Grid.ColumnSpan="2">

<DatePicker x:Name="HospitalReservationDate" Text="{Binding Path=HospitalReservationProfile.Date}" materialDesign:HintAssist.Hint="Reservation Date" Height="72" Style="{StaticResource MaterialDesignFloatingHintDatePicker}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<materialDesign:TimePicker x:Name="HospitalReservationTime" Style="{StaticResource MaterialDesignFloatingHintTimePicker}" materialDesign:HintAssist.Hint="Reservation Time" Height="78" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<TextBox x:Name="HospitalReservationReason" materialDesign:HintAssist.Hint="Reason" Text="{Binding Path=HospitalReservationProfile.Reason}" IsReadOnly="False" Height="76" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

</StackPanel>

</Grid>

<Grid x:Name="AdminRegPatientForm" Height="542" VerticalAlignment="Bottom" Margin="40,0,10,-29" Visibility="Collapsed" Grid.RowSpan="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="196\*"/>

<ColumnDefinition Width="179\*"/>

</Grid.ColumnDefinitions>

<TextBlock x:Name="AdminRegPatientRecord" Text="Edit Patient Record" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="303,-60,296,0" Foreground="Gray" FontSize="18" Height="28" Grid.ColumnSpan="2"></TextBlock>

<StackPanel VerticalAlignment="Center" Margin="10,10,34,132" Height="400" Grid.ColumnSpan="2">

<TextBox x:Name="AdminPatientName" Text="{Binding AdminPatientProfile.Name}" materialDesign:HintAssist.Hint="First Name" Height="55" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminPatientName\_TextChanged" />

<TextBox x:Name="AdminPatientMiddleName" Text="{Binding AdminPatientProfile.MiddleName}" materialDesign:HintAssist.Hint="Second Name" Height="59" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminPatientMiddleName\_TextChanged" />

<TextBox x:Name="AdminPatientFamilyName" Text="{Binding AdminPatientProfile.FamilyName}" materialDesign:HintAssist.Hint="Last Name" Height="57" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminPatientFamilyName\_TextChanged" />

<TextBox x:Name="AdminPatientAge" Text="{Binding AdminPatientProfile.Age}" materialDesign:HintAssist.Hint="Age" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminPatientAge\_TextChanged" />

<TextBox x:Name="AdminPatientEgn" Text="{Binding AdminPatientProfile.Egn}" materialDesign:HintAssist.Hint="EGN" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminPatientEgn\_TextChanged" />

<TextBox x:Name="AdminPatientAddress" Text="{Binding AdminPatientProfile.Address}" materialDesign:HintAssist.Hint="Address" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminPatientAddress\_TextChanged" />

<TextBox x:Name="AdminPatientMobilePhone" Text="{Binding AdminPatientProfile.MobilePhone}" materialDesign:HintAssist.Hint="Mobile Phone" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminPatientMobilePhone\_TextChanged" />

</StackPanel>

<

/Grid>

<Grid x:Name="AdminRegDoctorForm" Height="393" VerticalAlignment="Bottom" Margin="19,0,31,105" Visibility="Collapsed">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="196\*"/>

<ColumnDefinition Width="179\*"/>

</Grid.ColumnDefinitions>

<TextBlock x:Name="AdminRegDoctorRecord" Text="Edit Doctor Record" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="313,-90,287,0" Foreground="Gray" FontSize="18" Height="28" Grid.ColumnSpan="2"></TextBlock>

<StackPanel VerticalAlignment="Center" Margin="20,-36,0,0" Height="429" Grid.ColumnSpan="2">

<TextBox x:Name="AdminDoctorName" Text="{Binding Path=AdminDoctorProfile.Name}" materialDesign:HintAssist.Hint="First Name" Height="55" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminDoctorName\_TextChanged" />

<TextBox x:Name="AdminDoctorMiddleName" Text="{Binding Path=AdminDoctorProfile.MiddleName}" materialDesign:HintAssist.Hint="Second Name" Height="59" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminDoctorMiddleName\_TextChanged" />

<TextBox x:Name="AdminDoctorFamilyName" Text="{Binding Path=AdminDoctorProfile.FamilyName}" materialDesign:HintAssist.Hint="Last Name" Height="57" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminDoctorFamilyName\_TextChanged" />

<TextBox x:Name="AdminDoctorEgn" Text="{Binding Path=AdminDoctorProfile.Egn}" materialDesign:HintAssist.Hint="EGN" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminDoctorEgn\_TextChanged" />

<TextBox x:Name="AdminDoctorAddress" Text="{Binding Path=AdminDoctorProfile.Address}" materialDesign:HintAssist.Hint="Address" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminDoctorAddress\_TextChanged" />

<TextBox x:Name="AdminDoctorMobilePhone" Text="{Binding Path=AdminDoctorProfile.MobilePhone}" materialDesign:HintAssist.Hint="Mobile Phone" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminDoctorMobilePhone\_TextChanged" />

<TextBox x:Name="AdminDoctorSpecialization" Text="{Binding Path=AdminDoctorProfile.Specialization}" materialDesign:HintAssist.Hint="Specialization" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0" TextChanged="AdminDoctorSpecialization\_TextChanged"/>

</StackPanel>

</Grid>

Всяко от полетата, които могат да се редактират има пропърти, което държи наличната информарция за опреденият обект и при опит за редакция първо се валидира от функции в xaml.cs класа. Това става посредством събития при промяна да на текста на полето.

Пример:

private void AdminPatientName\_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

if (EditedRecord is Patient && User.UserRole.Equals("Admin"))

((Patient)EditedRecord).Name = AdminPatientName.Text;

}

private void AdminPatientMiddleName\_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

if (EditedRecord is Patient && User.UserRole.Equals("Admin"))

((Patient)EditedRecord).MiddleName = AdminPatientMiddleName.Text;

}

private void AdminPatientFamilyName\_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

if (EditedRecord is Patient && User.UserRole.Equals("Admin"))

((Patient)EditedRecord).FamilyName = AdminPatientFamilyName.Text;

}

private void AdminPatientAge\_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

if (EditedRecord is Patient && User.UserRole.Equals("Admin"))

((Patient)EditedRecord).Age = Int32.Parse(AdminPatientAge.Text);

}

private void AdminPatientEgn\_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

if (EditedRecord is Patient && User.UserRole.Equals("Admin"))

((Patient)EditedRecord).Egn = AdminPatientEgn.Text;

}

private void AdminPatientAddress\_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

if (EditedRecord is Patient && User.UserRole.Equals("Admin"))

((Patient)EditedRecord).Address = AdminPatientAddress.Text;

}

След като са вече валидирани промените мога да се запишат като се кликне бутонът Save, който прави PUT заявка и актуализира данните за съответният запис в базата.

Пример:

private void SaveBtn\_Click(object sender, RoutedEventArgs e)

{

if (EditedRecord is Visit && User.UserRole.Equals("Doctor"))

{

PUTVisitsRequest(User.AuthenticationCredentials, ((Visit)EditedRecord).Guid, new JObjectBuilder().With(visit => visit.VisitTime = ((DoctorVisitDate.SelectedDate.Value).Date.Add(DoctorVisitTime.SelectedTime.Value.TimeOfDay))).Build());

this.Close();

}

else if (EditedRecord is Reservation && User.UserRole.Equals("Doctor"))

{

PUTReservationRequest(User.AuthenticationCredentials, ((Reservation)EditedRecord).Guid, new JObjectBuilder().With(reservation => reservation.ReservationTime = ((DoctorReservationDate.SelectedDate.Value).Date.Add(DoctorReservationTime.SelectedTime.Value.TimeOfDay))).With(reservation => reservation.Reason = DoctorReservationReason.Text).Build());

this.Close();

}

else if (EditedRecord is Patient && User.UserRole.Equals("Admin"))

{

PUTPatientRequest(User.AuthenticationCredentials, ((Patient)EditedRecord).Guid, new JObjectBuilder().With(patient => patient.Name = ((Patient)EditedRecord).Name)

.With(patient => patient.MiddleName = ((Patient)EditedRecord).MiddleName)

.With(patient => patient.FamilyName = ((Patient)EditedRecord).FamilyName)

.With(patient => patient.Egn = ((Patient)EditedRecord).Egn)

.With(patient => patient.Age = ((Patient)EditedRecord).Age)

.With(patient => patient.MobilePhone = ((Patient)EditedRecord).MobilePhone)

.With(patient => patient.Address = ((Patient)EditedRecord).Address).Build());

this.Close();

}

else if (EditedRecord is Doctor && User.UserRole.Equals("Admin"))

{

PUTDoctorRequest(User.AuthenticationCredentials, ((Doctor)EditedRecord).Guid, new JObjectBuilder().With(doctor => doctor.Name = ((Doctor)EditedRecord).Name)

.With(doctor => doctor.MiddleName = ((Doctor)EditedRecord).MiddleName)

.With(doctor => doctor.FamilyName = ((Doctor)EditedRecord).FamilyName)

.With(doctor => doctor.Egn = ((Doctor)EditedRecord).Egn)

.With(doctor => doctor.MobilePhone = ((Doctor)EditedRecord).MobilePhone)

.With(doctor => doctor.Address = ((Doctor)EditedRecord).Address)

.With(doctor => doctor.Specialization = ((Doctor)EditedRecord).Specialization)

.Build());

this.Close();

}

else if (EditedRecord is Hospital && User.UserRole.Equals("Admin"))

{

PUTHospitalRequest(User.AuthenticationCredentials, ((Hospital)EditedRecord).Guid, new JObjectBuilder().With(hospital => hospital.Name = ((Hospital)EditedRecord).Name)

.With(hospital => hospital.Owner = ((Hospital)EditedRecord).Owner)

.With(hospital => hospital.MobilePhone = ((Hospital)EditedRecord).MobilePhone)

.With(hospital => hospital.Address = ((Hospital)EditedRecord).Address).Build());

this.Close();

}

else if (EditedRecord is Visit && User.UserRole.Equals("Hospital"))

{

PUTVisitsRequest(User.AuthenticationCredentials, ((Visit)EditedRecord).Guid, new JObjectBuilder().With(visit => visit.VisitTime = ((HospitalPatientDate.SelectedDate.Value).Date.Add(HospitalPatientTime.SelectedTime.Value.TimeOfDay))).Build());

this.Close();

}

else if (EditedRecord is Reservation && User.UserRole.Equals("Hospital"))

{

PUTReservationRequest(User.AuthenticationCredentials, ((Reservation)EditedRecord).Guid, new JObjectBuilder().With(reservation => reservation.ReservationTime = ((HospitalReservationDate.SelectedDate.Value).Date.Add(HospitalReservationTime.SelectedTime.Value.TimeOfDay))).Build());

this.Close();

}

}

XAML връзка:

<Button x:Name="saveBtn" Visibility="Collapsed" Content="Save" FontSize="18" Foreground="Black" Background="BlueViolet" HorizontalAlignment="Left" Margin="25,602,0,13" Width="264" Height="Auto" Grid.RowSpan="2" Click="SaveBtn\_Click"/>

PUT заявки:

private void PUTVisitsRequest(string BasicAuthentication, string Guid, JObject Visit)

{

var httpResponse = HttpRequest.Of($"https://localhost:44348/api/visits/{Guid}", RequestType.PUT)

.BasicAuthentication(BasicAuthentication)

.BodyData(Visit)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageBox.ShowMessageBoxInfo("Updating Info Message", "The Visit was successfully updated!");

}

}

private void PUTReservationRequest(string BasicAuthentication, string Guid, JObject Reservation)

{

var httpResponse = HttpRequest.Of($"https://localhost:44348/api/reservations/{Guid}", RequestType.PUT)

.BasicAuthentication(BasicAuthentication)

.BodyData(Reservation)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageBox.ShowMessageBoxInfo("Updating Info Message", "The Reservation was successfully updated!");

}

}

private void PUTPatientRequest(string BasicAuthentication, string Guid, JObject Patient)

{

var httpResponse = HttpRequest.Of($"https://localhost:44348/api/patients/{Guid}", RequestType.PUT)

.BasicAuthentication(BasicAuthentication)

.BodyData(Patient)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageBox.ShowMessageBoxInfo("Updating Info Message", "The Patient was successfully updated!");

}

}

При кликането на delete бутона се прави DELETE заявка, която изтрива записа от базата данни. Ако са нужни каскадни изтривания от базата се правят.

Пример:

XAML връзка:

private void DeleteBtn\_Click(object sender, RoutedEventArgs e)

{

if (EditedRecord is Visit && User.UserRole.Equals("Doctor"))

{

MessageForm MessageForm = new MessageForm("Confirmation Message", "Do you really want to delete this ?", () =>

{

DELETEVisitRequest(User.AuthenticationCredentials, ((Visit)EditedRecord).Guid);

this.Close();

});

;

MessageForm.ShowDialog();

return;

}

else if (EditedRecord is Reservation && User.UserRole.Equals("Doctor"))

{

MessageForm MessageForm = new MessageForm("Confirmation Message", "Do you really want to delete this ?", () =>

{

DELETEReservationRequest(User.AuthenticationCredentials, ((Reservation)EditedRecord).Guid);

this.Close();

});

MessageForm.ShowDialog();

return;

}

else if (EditedRecord is Patient && User.UserRole.Equals("Admin"))

{

MessageForm MessageForm = new MessageForm("Confirmation Message", "Do you really want to delete this ?", () =>

{

DELETEPatientRequest(User.AuthenticationCredentials, ((Patient)EditedRecord).Guid);

this.Close();

});

MessageForm.ShowDialog();

return;

}

else if (EditedRecord is Doctor && User.UserRole.Equals("Admin"))

{

MessageForm MessageForm = new MessageForm("Confirmation Message", "Do you really want to delete this ?", () =>

{

DELETEDoctorRequest(User.AuthenticationCredentials, ((Doctor)EditedRecord).Guid);

this.Close();

});

MessageForm.ShowDialog();

return;

}

else if (EditedRecord is Hospital && User.UserRole.Equals("Admin"))

{

MessageForm MessageForm = new MessageForm("Confirmation Message", "Do you really want to delete this ?", () =>

{

DELETEHospitalRequest(User.AuthenticationCredentials, ((Hospital)EditedRecord).Guid);

this.Close();

});

MessageForm.ShowDialog();

return;

}

else if (EditedRecord is Visit && User.UserRole.Equals("Hospital"))

{

MessageForm MessageForm = new MessageForm("Confirmation Message", "Do you really want to delete this ?", () =>

{

DELETEVisitRequest(User.AuthenticationCredentials, ((Visit)EditedRecord).Guid);

this.Close();

});

MessageForm.ShowDialog();

return;

}

else if (EditedRecord is Reservation && User.UserRole.Equals("Hospital"))

{

MessageForm MessageForm = new MessageForm("Confirmation Message", "Do you really want to delete this ?", () =>

{

DELETEReservationRequest(User.AuthenticationCredentials, ((Reservation)EditedRecord).Guid);

this.Close();

});

MessageForm.ShowDialog();

return;

}

}

<Button x:Name="deleteBtn" Visibility="Collapsed" Content="Delete" FontSize="18" Foreground="Black" Background="PaleVioletRed" HorizontalAlignment="Right" Width="264" Margin="0,602,41,13" Click="DeleteBtn\_Click" Height="Auto" Grid.RowSpan="2"/>

DELETE заявки:

private void DELETEVisitRequest(string BasicAuthentication, string Guid)

{

var httpResponse = HttpRequest.Of($"https://localhost:44348/api/visits/{Guid}", RequestType.DELETE)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageBox.ShowMessageBoxInfo("Deleting Info Message", "The Visit was successfully deleted!");

}

}

private void DELETEReservationRequest(string BasicAuthentication, string Guid)

{

var httpResponse = HttpRequest.Of($"https://localhost:44348/api/reservations/{Guid}", RequestType.DELETE)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageBox.ShowMessageBoxInfo("Deleting Info Message", "The Reservation was successfully deleted!");

}

}

private void DELETEPatientRequest(string BasicAuthentication, string Guid)

{

var httpResponse = HttpRequest.Of($"https://localhost:44348/api/patients/{Guid}", RequestType.DELETE)

.BasicAuthentication(BasicAuthentication)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageBox.ShowMessageBoxInfo("Deleting Info Message", "The Patient was successfully deleted!");

}

}

**Info Form**

Тази форма се използа за визуализацията на по-подробна информация на пациента, доктора или болницата. Данните, за определеният обект се подават като параметър. Проверява се типът на обекта и в зависимост от това се решава коя форма да се визуализира тази за човек или тази за болница. Всички полета тук са ReadOnly. Това става чрез задаването на пропъртито на ТеxtBox-a IsReadOnly на true, което позволява полетата да не могат да бъдат променяни. Мапването на полетата става чрез Binding пропъртито към съответното поле.

ХАМL код:

<Grid x:Name="PersonInfo" Width="750" Height="704" VerticalAlignment="Top" Margin="25,0,25,-116" Visibility="Collapsed">

<Rectangle Height="416" Fill="White" VerticalAlignment="Bottom" Margin="0,0,0,245" RadiusY="21.2" RadiusX="21.2"/>

<TextBlock x:Name="PersonInfoTitle" Text="" TextAlignment="Center" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="298,10,302,0" Foreground="Gray" FontSize="18" Height="28" Width="150"/>

<StackPanel VerticalAlignment="Center" Margin="10,43,10,243" Height="418">

<TextBox x:Name="FirstName" Text="{Binding Path=Person.Name}" IsReadOnly="True" materialDesign:HintAssist.Hint="First Name" Height="55" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="10,0,10.4,0"/>

<TextBox x:Name="SecondName" Text="{Binding Path=Person.MiddleName}" IsReadOnly="True" materialDesign:HintAssist.Hint="Second Name" Height="59" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="10,0,10.4,0"/>

<TextBox x:Name="LastName" Text="{Binding Path=Person.FamilyName}" IsReadOnly="True" materialDesign:HintAssist.Hint="Last Name" Height="57" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="10,0,10.4,0"/>

<TextBox x:Name="Age" Text="{Binding Path=Person.Age}" IsReadOnly="True" materialDesign:HintAssist.Hint="Age" Height="51" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="10,0,10.4,0"/>

<TextBox x:Name="Address" Text="{Binding Path=Person.Address}" IsReadOnly="True" materialDesign:HintAssist.Hint="Address" Height="52" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="10,0,10.4,0"/>

<TextBox x:Name="MobilePhone" IsReadOnly="True" Text="{Binding Path=Person.MobilePhone}" materialDesign:HintAssist.Hint="Mobile Phone" Height="52" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="10,0,10.4,0"/>

<TextBox x:Name="Specialization" materialDesign:HintAssist.Hint="Specialization" Text="{Binding Path=Person.Specialization}" TextWrapping="Wrap" TextAlignment="Left" VerticalContentAlignment="Top" AcceptsReturn="True" HorizontalScrollBarVisibility="Disabled" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontSize="18" VerticalScrollBarVisibility="Auto" Visibility="Collapsed" Margin="10,0,10.4,0" Height="54" />

</StackPanel>

</Grid>

<Grid x:Name="HospitalInfo" Height="588" VerticalAlignment="Bottom" Visibility="Collapsed">

<Rectangle Height="304" Fill="White" VerticalAlignment="Bottom" Margin="28,0,28,75" RadiusY="21.2" RadiusX="21.2"/>

<TextBlock x:Name="HospitalTittle" Text="Hospital Information " HorizontalAlignment="Center" VerticalAlignment="Top" Margin="307,153,326,0" Foreground="Gray" FontSize="18" Height="30" Width="167"/>

<StackPanel VerticalAlignment="Center" Margin="51,234,28,108" Height="246">

<TextBox materialDesign:HintAssist.Hint="Hospital Name" x:Name="HospitalName" Text="{Binding Path=Hospital.Name}" IsReadOnly="True" Height="55" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<TextBox materialDesign:HintAssist.Hint="Owner(First and Last Name)" x:Name="Owner" Text="{Binding Path=Hospital.Owner}" IsReadOnly="True" Height="55" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<TextBox materialDesign:HintAssist.Hint="Address" x:Name="HospitalAddress" Text="{Binding Path=Hospital.Address}" Height="51" IsReadOnly="True" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<TextBox x:Name="HospitalMobilePhone" materialDesign:HintAssist.Hint="Mobile Phone" Text="{Binding Path=Hospital.MobilePhone}" Height="51" IsReadOnly="True" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

</StackPanel>

</Grid>

Чрез следният код се избира кой Grid да бъде визуализиран. Всичките Grid-ве има пропърти Visibility="Collapsed" което позволява да не бъде изобщо зареден грида при стартиране на приложението.

private void showForm(object InfoObject)

{

if(InfoObject is Patient || InfoObject is Doctor)

{

if (InfoObject is Doctor)

{

this.Specialization.Visibility = Visibility.Visible;

this.Age.Visibility = Visibility.Collapsed;

}

this.showPersonForm(InfoObject);

} else if(InfoObject is Hospital)

{

this.showHospitalForm(InfoObject);

}

}

private void showPersonForm(object Person)

{

this.DataContext = this;

var personTitle = (Person is Patient ? "Patient" : (Person is Doctor ? "Doctor" : ""));

this.PersonInfoTitle.Text = $"{personTitle} Info";

this.Person = (Person)Person;

this.InfoGrid.Visibility = Visibility.Visible;

this.PersonInfo.Visibility = Visibility.Visible;

}

private void showHospitalForm(object Hospital)

{

this.DataContext = this;

this.Hospital = (Hospital) Hospital;

this.InfoGrid.Visibility = Visibility.Visible;

this.HospitalInfo.Visibility = Visibility.Visible;

}

**Message Form**

Тази форма е проектирана за визуализацията на съобщения от следните видове:

* Информационни/ Съобщения за грешка
* Съобщения с потвръждение

При извикването на тази форма и подадените в нея параметри може да се определи какъв е видът на съобщението. Ако има само подадено заглавие на съобщението или тяло на съобщението, то тогава е информационно или съобщение за грешка. Но ако има и параметър от тип Аction, то тогава следва, че ще бъде съобщение с потвръждение, защото ще трябва да се изпълни някакъво действие след потвръждението.

Заглавието и самото съобщение се свързват с TextBox-вете в XAML чрез използването на пропъртито Binding, като му се подават съответният път от където да вземе данните.

XAML код:

<Grid>

<TextBlock x:Name="MsgTitle" DockPanel.Dock="Top" Text="{Binding Path=MessageTitle}" HorizontalAlignment="Stretch" VerticalAlignment="Top" FontSize="18" Background="SandyBrown" TextAlignment="Center" Foreground="WhiteSmoke" FontFamily="Arial" FontWeight="DemiBold" Height="35" Padding="5"></TextBlock>

<Rectangle Height="215" VerticalAlignment="Top" Margin="0,35,0,-0.4" RenderTransformOrigin="0.5,0.5">

<Rectangle.Fill>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0.2">

<GradientStop Color="Coral" />

<GradientStop Color="Salmon" Offset="0.3"/>

<GradientStop Color="Chocolate" Offset="0.8"/>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<Rectangle Height="143" Fill="White" VerticalAlignment="Bottom" Width="auto" Margin="32,0,34,48.6" RadiusY="21.2" RadiusX="21.2"/>

<Button x:Name="ExitBtn" Background="Transparent" BorderBrush="Transparent" HorizontalAlignment="Left" Margin="448,-5,-8,0" VerticalAlignment="Top" RenderTransformOrigin="-0.613,0.156" Width="60" Height="40" Click="ExitBtn\_Click">

<ContentControl>

<Image Source="../Assets/xButton.png" Height="28" RenderTransformOrigin="0.5,0.5" Width="55" />

</ContentControl>

</Button>

<TextBlock x:Name="MsgContent" TextWrapping="Wrap" TextAlignment="Center" Text="{Binding Path=MessageContent}" VerticalAlignment="Center" FontSize="18" Margin="56,58,48,48.6"/>

<Button x:Name="ConfirmBtn" Visibility="Collapsed" Content="Yes" FontSize="16" Foreground="Black" Background="BlueViolet" HorizontalAlignment="Left" Margin="96,208,0,7.6" Width="90" Height="Auto" Click="ConfirmBtn\_Click"/>

<Button x:Name="CancelBtn" Visibility="Collapsed" Content="No" FontSize="16" Foreground="Black" Background="PaleVioletRed" HorizontalAlignment="Right" Width="90" Height="Auto" Margin="0,208,117,9.6" Click="CancelBtn\_Click"/>

</Grid>

Когато съобщението е с потвръждение се показват и бутоните Yes и No.

Xaml.cs код :

public MessageForm(string MessageTitle, string MessageContent)

{

InitializeComponent();

this.DataContext = this;

this.MessageTitle = MessageTitle;

this.MessageContent = MessageContent;

}

public MessageForm(string MessageTitle, string MessageContent, Action ActionOnConfirm)

{

InitializeComponent();

this.DataContext = this;

this.MessageTitle = MessageTitle;

this.MessageContent = MessageContent;

this.ActionOnConfirm = ActionOnConfirm;

ShowConfirmationMessageBox();

}

private void Window\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (e.LeftButton == MouseButtonState.Pressed)

{

DragMove();

}

}

private void ExitBtn\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void CancelBtn\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void ConfirmBtn\_Click(object sender, RoutedEventArgs e)

{

ActionOnConfirm();

this.Close();

}

private void ShowConfirmationMessageBox()

{

this.ConfirmBtn.Visibility = Visibility.Visible;

this.CancelBtn.Visibility = Visibility.Visible;

}

**Reservation Form**

Тази форма се използва за създаването на резервации. При извикването на тази форма се подават като параметри, потребителят и какъв тип е резервацията – направена от доктора или от болницата. Вземат се всички налични пациенти, доктори и болници и според зависи кой прави заявката (доктор или пациент), в падащи менюта се визуализират само болниците, в който работи доктора и докторите, които работя в дадената болница. Това става чрез правеното на GET заявки. Валидирани причината за резервацията и часът и датата . А чрез кликането на Save бутона се PОST заявка, която създадва новият запис в базата даннни.

Xaml.cs код:

private void POSTReservationRequest(string BasicAuthentication, JObject Reservation)

{

var httpResponse = HttpRequest.Of("https://localhost:44348/api/reservations", RequestType.POST)

.BasicAuthentication(BasicAuthentication)

.BodyData(Reservation)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageForm MessageForm = new MessageForm("Info", "Successfully created");

MessageForm.ShowDialog();

this.Close();

}

}

private void SaveBtn\_Click(object sender, RoutedEventArgs e)

{

if (!ValidateReservationDate())

return;

else if (!ValidateReservationTime())

return;

else if (!ValidateReservationPatient())

return;

else if (!ValidateReservationDoctor())

return;

else if (!ValidateReservationHospital())

return;

else if (!ValidateReason())

return;

else

{

switch (ReservationType)

{

case "DoctorReservationForm":

JObject doctorReservation = new JObjectBuilder()

.With(reservation => reservation.Patient = new JObjectBuilder().With(patient => patient.Guid = PatientList[DoctorReservationPatient.SelectedIndex].Guid).Build())

.With(reservation => reservation.Doctor = new JObjectBuilder().With(doctor => doctor.Guid = User.Guid).Build())

.With(reservation => reservation.Hospital = new JObjectBuilder().With(hospital => hospital.Guid = HospitalList[DoctorReservationHospital.SelectedIndex].Guid).Build())

.With(reservation => reservation.ReservationTime = (DoctorReservationDate.SelectedDate.Value).Date.Add(DoctorReservationTime.SelectedTime.Value.TimeOfDay))

.With(reservation => reservation.Reason = DoctorReason.Text)

.Build();

POSTReservationRequest(User.AuthenticationCredentials, doctorReservation);

return;

case "HospitalReservationForm":

JObject hospitalReservation = new JObjectBuilder()

.With(reservation => reservation.Patient = new JObjectBuilder()

.With(patient => patient.Guid = PatientList[HospitalReservationPatient.SelectedIndex].Guid)

.Build())

.With(reservation => reservation.Doctor = new JObjectBuilder().With(doctor => doctor.Guid = DoctorList[HospitalReservationDoctor.SelectedIndex].Guid)

.Build())

.With(reservation => reservation.Hospital = new JObjectBuilder()

.With(hospital => hospital.Guid = User.Guid)

.Build())

.With(reservation => reservation.ReservationTime = (HospitalReservationDate.SelectedDate.Value).Date.Add(HospitalReservationTime.SelectedTime.Value.TimeOfDay))

.With(reservation => reservation.Reason = HospitalReason.Text)

.Build();

POSTReservationRequest(User.AuthenticationCredentials, hospitalReservation);

return;

}

}

}

**Visitation Form**

Тази форма се използва за създаването на посещения. Тя е аналогична на формата за резервации. При извикването на тази форма се подават като параметри, потребителят и какъв тип е посещението – направена от доктора или от болницата. Вземат се всички налични пациенти, доктори и болници и според зависи кой прави заявката (доктор или пациент), в падащи менюта се визуализират само болниците, в който работи доктора и докторите, които работя в дадената болница. Това става чрез правеното на GET заявки. Валидирани са диагнозата, часът и датата . А чрез кликането на Save бутона се PОST заявка, която създадва новият запис в базата даннни.

ХАМЛ код:

<Grid Margin="0,0,0,-36.2">

<Rectangle Height="528" VerticalAlignment="Top" Margin="0,0,-3,0">

<Rectangle.Fill>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF93FA93"/>

<GradientStop Color="#FF11E87C" Offset="0.2"/>

<GradientStop Color="#FF5CD45A" Offset="0.5"/>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<Rectangle Height="320" Fill="White" VerticalAlignment="Bottom" Margin="10,0,10,107" RadiusY="21.2" RadiusX="21.2"/>

<Grid x:Name="DoctorPSForm" Height="542" VerticalAlignment="Bottom" Margin="40,-56,-3,-29.2" Visibility="Collapsed">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="196\*"/>

<ColumnDefinition Width="179\*"/>

</Grid.ColumnDefinitions>

<TextBlock x:Name="DoctoPSTitle" Text="Add Visitation" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="299,42,352.2,0" Foreground="Gray" FontSize="18" Height="28" Grid.ColumnSpan="2" RenderTransformOrigin="0.511,-0.286"></TextBlock>

<StackPanel VerticalAlignment="Center" Margin="0,114,44.2,168" Height="260" Grid.ColumnSpan="2">

<DatePicker x:Name="DoctorPSDate" materialDesign:HintAssist.Hint="Date" Height="51" Style="{StaticResource MaterialDesignFloatingHintDatePicker}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10.2,0"/>

<materialDesign:TimePicker x:Name="DoctorPSTime" Style="{StaticResource MaterialDesignFloatingHintTimePicker}" materialDesign:HintAssist.Hint="Time" Height="45" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<ComboBox x:Name="DoctorPSPatient" materialDesign:HintAssist.Hint="Patient" Height="55" Style="{StaticResource MaterialDesignFloatingHintComboBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10.4,0"/>

<ComboBox x:Name="DoctorPSHospital" materialDesign:HintAssist.Hint="Hospital" Height="51" Style="{StaticResource MaterialDesignFloatingHintComboBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<TextBox x:Name="DoctorPSDiagnose" materialDesign:HintAssist.Hint="Diagnose" Height="52" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10.2,0"/>

</StackPanel>

</Grid>

<Grid x:Name="HospitalPSForm" Height="542" VerticalAlignment="Bottom" Margin="40,-56,-3,-29.2" Visibility="Collapsed">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="196\*"/>

<ColumnDefinition Width="179\*"/>

</Grid.ColumnDefinitions>

<TextBlock x:Name="HospitalPSTitle" Text="Add Visitation" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="318,35,326.8,0" Foreground="Gray" FontSize="18" Height="28" Grid.ColumnSpan="2"></TextBlock>

<StackPanel VerticalAlignment="Center" Margin="0,114,44.2,168" Height="260" Grid.ColumnSpan="2">

<DatePicker x:Name="HospitalPSDate" materialDesign:HintAssist.Hint="Date" Height="51" Style="{StaticResource MaterialDesignFloatingHintDatePicker}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10.2,0"/>

<materialDesign:TimePicker x:Name="HospitalPSTime" Style="{StaticResource MaterialDesignFloatingHintTimePicker}" materialDesign:HintAssist.Hint="Time" Height="45" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<ComboBox x:Name="HospitalPSPatient" materialDesign:HintAssist.Hint="Patient" Height="55" Style="{StaticResource MaterialDesignFloatingHintComboBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10.4,0"/>

<ComboBox x:Name="HospitalPSDoctor" materialDesign:HintAssist.Hint="Doctor" Height="51" Style="{StaticResource MaterialDesignFloatingHintComboBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10,0"/>

<TextBox x:Name="HospitalPSDiagnose" materialDesign:HintAssist.Hint="Diagnose" Height="52" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontFamily="Champagne &amp; Limousines" FontSize="18" Margin="0,0,10.2,0"/>

</StackPanel>

</Grid>

<Button x:Name="ExitBtn" Background="Transparent" BorderBrush="Transparent" HorizontalAlignment="Left" Margin="740,0,0,0" VerticalAlignment="Top" RenderTransformOrigin="-0.613,0.156" Width="60" Height="40" Click="ExitBtn\_Click">

<ContentControl>

<Image Source="../Assets/xButton.png" Height="28" RenderTransformOrigin="0.5,0.5" Width="55" />

</ContentControl>

</Button>

<Button x:Name="minimizeBtn" Background="Transparent" BorderBrush="Transparent" HorizontalAlignment="Left" Margin="704,0,0,0" VerticalAlignment="Top" RenderTransformOrigin="-0.613,0.156" Width="55" Height="40" Click="MinimizeBtn\_Click" >

<ContentControl>

<Image Source="../Assets/minimize.png" Height="25" RenderTransformOrigin="0.5,0.5" Width="33" />

</ContentControl>

</Button>

<Button x:Name="saveBtn" Visibility="Collapsed" Content="Save" FontSize="18" Foreground="Black" Background="BlueViolet" HorizontalAlignment="Left" Margin="260,421,0,40" Width="264" Click="saveBtn\_Click"/>

</Grid>

XAML.cs код:

PUT заявки:

private void POSTVisitsRequest(string BasicAuthentication, JObject visit)

{

var httpResponse = HttpRequest.Of("https://localhost:44348/api/visits", RequestType.POST)

.BasicAuthentication(BasicAuthentication)

.BodyData(visit)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageForm MessageForm = new MessageForm("Info", "The Visit is successfully created");

MessageForm.ShowDialog();

this.Close();

}

}

private void POSTDiagnosisRequest(string BasicAuthentication, JObject diagnosis)

{

var httpResponse = HttpRequest.Of("https://localhost:44348/api/diagnoses", RequestType.POST)

.BasicAuthentication(BasicAuthentication)

.BodyData(diagnosis)

.OnError(MessageBox.ShowMessageBoxError("Warning Data Input", ""))

.Execute();

if (httpResponse.IsSuccessful())

{

MessageForm MessageForm = new MessageForm("Info", "The Diagnose is successfully created");

MessageForm.ShowDialog();

this.Close();

}

}

private void saveBtn\_Click(object sender, RoutedEventArgs e)

{

if (!ValidatePSDate())

return;

else if (!ValidatePSTime())

return;

else if (!ValidatePSPatient())

return;

else if (!ValidatePSDoctor())

return;

else if (!ValidatePSHospital())

return;

else if (!ValidateDiagnose())

return;

else

{

switch (VisitationType)

{

case "DoctorPSForm":

JObject doctorVisitation = new JObjectBuilder()

.With(visit => visit.Patient = new JObjectBuilder()

.With(patient => patient.Guid = Patients[DoctorPSPatient.SelectedIndex].Guid)

.Build())

.With(visit => visit.Doctor = new JObjectBuilder()

.With(doctor => doctor.Guid = User.Guid)

.Build())

.With(visit => visit.Hospital = new JObjectBuilder()

.With(hospital => hospital.Guid = Hospitals[DoctorPSHospital.SelectedIndex].Guid)

.Build())

.With(visit => visit.VisitTime = (DoctorPSDate.SelectedDate.Value).Date.Add(DoctorPSTime.SelectedTime.Value.TimeOfDay))

.Build();

POSTVisitsRequest(User.AuthenticationCredentials, doctorVisitation);

JObject doctorDiagnosis = new JObjectBuilder()

.With(diagnosis => diagnosis.Patient = new JObjectBuilder()

.With(patient => patient.Guid = Patients[DoctorPSPatient.SelectedIndex].Guid)

.Build())

.With(diagnosis => diagnosis.Doctor = new JObjectBuilder()

.With(doctor => doctor.Guid = doctor.Guild = User.Guid)

.Build())

.With(diagnosis => diagnosis.DiagnosisTime = (DoctorPSDate.SelectedDate.Value).Date.Add(DoctorPSTime.SelectedTime.Value.TimeOfDay))

.With(diagnosis => diagnosis.DiagnosisDescription = DoctorPSDiagnose.Text)

.Build();

POSTDiagnosisRequest(User.AuthenticationCredentials, doctorDiagnosis);

return;

case "HospitalPSForm":

JObject hospitalVisitation = new JObjectBuilder()

.With(visit => visit.Patient = new JObjectBuilder()

.With(patient => patient.Guid = Patients[HospitalPSPatient.SelectedIndex].Guid)

.Build())

.With(visit => visit.Doctor = new JObjectBuilder()

.With(doctor => doctor.Guid = Doctors[HospitalPSDoctor.SelectedIndex].Guid)

.Build())

.With(visit => visit.Hospital = new JObjectBuilder()

.With(hospital => hospital.Guid = User.Guid)

.Build())

.With(visit => visit.VisitTime = (HospitalPSDate.SelectedDate.Value).Date.Add(HospitalPSTime.SelectedTime.Value.TimeOfDay))

.Build();

POSTVisitsRequest(User.AuthenticationCredentials, hospitalVisitation);

JObject hospitalDiagnosis = new JObjectBuilder()

.With(diagnosis => diagnosis.Patient = new JObjectBuilder()

.With(patient => patient.Guid = Patients[HospitalPSPatient.SelectedIndex].Guid)

.Build())

.With(diagnosis => diagnosis.Doctor = new JObjectBuilder()

.With(doctor => doctor.Guid = Doctors[HospitalPSDoctor.SelectedIndex].Guid)

.Build())

.With(diagnosis => diagnosis.DiagnosisDescription = HospitalPSDiagnose.Text)

.With(diagnosis => diagnosis.DiagnosisTime = (HospitalPSDate.SelectedDate.Value).Date.Add(HospitalPSTime.SelectedTime.Value.TimeOfDay))

.Build();

POSTDiagnosisRequest(User.AuthenticationCredentials, hospitalDiagnosis);

return;

}

}

}

**Заключение**:

Нашият продукт е готов за употреба и среща нуждите на нашите потребители. Смятам, че достатъчно конкурентноспособен на база на вечете разработени функционалности.

Литература:

<https://stackoverflow.com/>

<http://materialdesigninxaml.net/>