

**Курсова работа Nº 1**

**Тема:** Медицинско заведение

**Изготвил:** Стивън Ивов Иванов

**Факултетен номер:** 21621523

**Специалност:** СИТ

**Дисциплина:** Технология на софтуерното производство - проект

**Курс:** 3

**Група:** 1а

[Въведение 4](#_Toc165933858)

[Цел на документа 4](#_Toc165933859)

[Обхват на проекта 4](#_Toc165933860)

[Технологии и инструменти 4](#_Toc165933861)

[Реализация 5](#_Toc165933862)

[Модул за Аутентификация (Login Form) 5](#_Toc165933863)

[Въведение 5](#_Toc165933864)

[Функционалности 5](#_Toc165933865)

[Влизане в профил (Login) 5](#_Toc165933866)

[Описание: 5](#_Toc165933867)

[Примерен код: 5](#_Toc165933868)

[Начален Екран (Home Form) 6](#_Toc165933869)

[Въведение: 6](#_Toc165933870)

[Лекарски Панел (Doctor Panel Form) 6](#_Toc165933871)

[Въведение 6](#_Toc165933872)

[Функционалности 6](#_Toc165933873)

[Създаване на пациент (Create Patient) 6](#_Toc165933874)

[Описание: 6](#_Toc165933875)

[Примерен код: 6](#_Toc165933876)

[Търсене на пациент (Search Patient) 7](#_Toc165933877)

[Описание: 7](#_Toc165933878)

[Примерен код: 7](#_Toc165933879)

[Създаване на заболявания (Create Illness) 9](#_Toc165933880)

[Описание: 9](#_Toc165933881)

[Примерен код: 9](#_Toc165933882)

[Намиране на всички предписания на даден пациент (Show all prescriptions of patient): 10](#_Toc165933883)

[Описание: 10](#_Toc165933884)

[Примерен код: 10](#_Toc165933885)

[Административен Панел (Admin Panel Form) 10](#_Toc165933886)

[Въведение 10](#_Toc165933887)

[Функционалности 11](#_Toc165933888)

[Създраване на доктор (Create Doctor) 11](#_Toc165933889)

[Описание: 11](#_Toc165933890)

[Примерен код: 11](#_Toc165933891)

[Създаване на медицинска специалност (Create Doctor Specialty) 12](#_Toc165933892)

[Описание: 12](#_Toc165933893)

[Примерен код: 12](#_Toc165933894)

[Изтриване на медицинска специалност (Delete Doctor Specialty) 13](#_Toc165933895)

[Описание: 13](#_Toc165933896)

[Примерен код: 13](#_Toc165933897)

[Редактиране на доктор (Edit doctor) 14](#_Toc165933898)

[Описание: 14](#_Toc165933899)

[Примерен код: 14](#_Toc165933900)

[Търсене на доктор (Search Doctor) 15](#_Toc165933901)

[Описание: 15](#_Toc165933902)

[Примерен код: 15](#_Toc165933903)

[Ръководство за потребителя. 17](#_Toc165933904)

[Въведение 17](#_Toc165933905)

[Основни модули и тяхната употреба 18](#_Toc165933906)

[Модул за аутентификация (Login Form) 18](#_Toc165933907)

[Начален екран (Home Form) 18](#_Toc165933908)

[Лекарски панел (Doctor Panel Form) 18](#_Toc165933909)

[Административен панел (Admin Panel Form) 18](#_Toc165933910)

[Тестване на приложението 18](#_Toc165933911)

[Методи на Тестване 18](#_Toc165933912)

[Резултати от Тестването 19](#_Toc165933913)

[Изводи 19](#_Toc165933914)

[Възможности за Развитие 19](#_Toc165933915)

[Използвана Литература 19](#_Toc165933916)

[Общодостъпни Ресурси и Техническа Документация 19](#_Toc165933917)

Медицинско заведение

# Въведение

## Цел на документа

Целта на този проект е разработването на интегрирана система за управление на медицинска информация, предназначена за медицински заведения, използвайки C# и Windows Forms. Системата цели да оптимизира процеса на обработка на медицински данни и да подобри комуникацията между медицински специалисти и отдели. Изборът на C# е определен от високата му производителност и интеграцията с други Microsoft технологии, което улеснява разработката на надеждни и мащабируеми приложения. Като резултат от проекта беше създадена функционална платформа, която демонстрира значителни подобрения в скоростта на обработка на данни и управлението на информацията в сравнение с предишни системи. Развитието предоставя стабилна основа за бъдещи усъвършенствания и потенциално разширение на функционалността.

## Обхват на проекта

Обхватът на проекта включва разработката на базовите функционалности на системата, като вписване на пациенти, визуализация на медицински данни. Проектът не включва разработката на мобилно приложение или пряка интеграция с други външни системи за момента.

# Технологии и инструменти

Проектът използва C# и Windows Forms за създаването на медицинската информационна система. C# е избран поради мощните си функции за обработка на данни, отличната поддръжка от Microsoft и голямото общество от разработчици. Windows Forms пък предоставя богат набор от инструменти за създаване на графични потребителски интерфейси, което е от съществено значение за удобството на крайните потребители.

Архитектурата на системата е модулна, като включва отделни компоненти за управление на пациентски данни, взаимодействие със сървъри за данни и потребителски интерфейс. Тези модули са свързани чрез ясно дефинирани интерфейси, което улеснява мащабирането и поддръжката на системата.

Изборът на тези технологии е обусловен от нуждата за висока надеждност, лесна поддръжка и гъвкавост при разширенията на функционалностите в бъдеще.

# Реализация

Системата включва четири основни модула, предназначени да улеснят интеракцията на различни потребители с информационната база.

## Модул за Аутентификация (Login Form)

### Въведение

Поддържа идентификацията на потребителите чрез роли (администратор, лекар), осигурявайки разграничен достъп до функционалностите на системата.

### Функционалности

#### Влизане в профил (Login)

##### Описание:

Тази функция позволява на потребителят да влезе в системата.

##### Примерен код:

private void loginButton\_Click(object **sender**, *EventArgs* **e**)

{

var **\_email** = emailTextBox.*Text*;

var **\_password** = passwordTextBox.*Text*;

User **\_user** = LoginUser(**\_email**, **\_password**);

if (**\_user** != null)

{

*Form* **\_formToDisplay**;

switch (**\_user**.Role.Name.*ToLower*()) // за всеки случай ако ролите са в друг case

{

case "admin":

**\_formToDisplay** = new AdminPanelForm(**\_user**);

break;

case "doctor":

**\_formToDisplay** = new DoctorPanelForm(**\_user**);

break;

default:

throw new *ArgumentException*("Това приложение не подържа други роли.");

}

this.*Hide*(); // скрий сегашната форма

**\_formToDisplay**.*Closed* += (*s*, *args*) => this.*Close*(); // при затваряне на следваща форма затвори и тази (която е скрита вмомента)

**\_formToDisplay**.*Show*(); // покажи новата форма

}

else

{

errorLabel.*Visible* = true;

}

}

private User LoginUser(string **t\_email**, string **t\_password**)

{

// тук използваме SingleOrDefault за да намерим от таблицата Users ЕДИНСТВЕНИЯ

// user с такъв имейл и парола. Ако има повече от един user с такъв имейл

// ще получим грешка / но това няма как да стане тъй като в самата база от данни

// сме задали Email полето да е UNIQUE/ да няма еднакви имейли

// Ако не бъде намерен един user с такъв имейл и парола ще бъде върнано NULL

var **\_user** = m\_db.Users

.*Include*(*u* => *u*.Role) // зареди ми navigational property-то за ролите/ за да имам достъп до тях

.*SingleOrDefault*(*u* => *u*.Email == **t\_email** && *u*.Password == **t\_password**);

return **\_user**;

}

## Начален Екран (Home Form)

### Въведение:

* Предоставя информация за успешно входиран потребител и неговите правомощия в системата.
* Служи като централен навигационен хъб към специфичните модули, достъпни за съответната потребителска роля.

## Лекарски Панел (Doctor Panel Form)

### Въведение

* Създаване на пациент: Лекарите въвеждат основни данни за нови пациенти.
* Търсене на пациент: Функционалност, която позволява бързо и точно търсене на пациентски записи по множество критерии.
* Създаване на заболявания: Включва регистрация на новодиагностицирани заболявания в пациентската база данни.
* Намиране на всички предписания на даден пациент

### Функционалности

#### Създаване на пациент (Create Patient)

##### Описание:

Тази функция позволява на лекарите да регистрират нови пациенти в системата, като се въвеждат основни данни като име, презиме, фамилия и ЕГН.

##### Примерен код:

private async void createPatientButton\_Click(object **sender**, *EventArgs* **e**)

{

if (!CheckIfAllInfoIsFilled())

{

*MessageBox*.*Show*("Моля попълни всички полета нужни за създаването на пациент!", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

}

else

{

// ако ЕГН-то не е точно 10 символа ИЛИ на ЕГН-то всичките му символи НЕ са числа

if (egnTextBox.*Text*.*Length* != 10 || !(egnTextBox.*Text*.*All*(char.*IsDigit*)))

{

*MessageBox*.*Show*("Невалидно ЕГН. Егнто трябва да е точно 10 символа дълго и да съдържа само цифри", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

// провери дали има пациент с регистрирано такова ЕГН

var **\_patientAlreadyExists** = m\_db.Patients.*FirstOrDefault*(*x* => *x*.EGN == egnTextBox.*Text*);

if (**\_patientAlreadyExists** != null)

{

*MessageBox*.*Show*("Вече има регистриран пациент с такова ЕГН.", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

// ако пациента НЕ съществува ще продължи надолу

// създай пациент

var **\_patient** = new Patient()

{

FirstName = firstNameTextBox.*Text*,

MiddleName = middleNameTextBox.*Text*,

LastName = lastNameTextBox.*Text*,

EGN = egnTextBox.*Text*,

};

// ако има избрано заболяване за съответния пациент, запамети го

if (medicalConditionListBox.*SelectedIndex* != -1)

{

var **\_medicalCondition** = m\_db.MedicalConditions.*Single*(*x* => *x*.Name == medicalConditionListBox.*SelectedItem*.*ToString*());

**\_patient**.MedicalConditionId = **\_medicalCondition**.Id;

}

await m\_db.Patients.*AddAsync*(**\_patient**);

await m\_db.*SaveChangesAsync*();

foreach (var **\_control** in m\_createPatientControls)

{

**\_control**.*Text* = "";

}

medicalConditionListBox.*SelectedIndex* = -1;

*MessageBox*.*Show*("Вие успешно създадохте нов пациент. Вече можете да му изписвате рецепти.", "Успешно създаден пациент.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

}

}

#### Търсене на пациент (Search Patient)

##### Описание:

ази функционалност позволява на лекарите да търсят пациенти по множество критерии, включително име и ЕГН, за бърз достъп до медицински записи.

##### Примерен код:

private void searchPatientButton\_Click(object **sender**, *EventArgs* **e**)

{

if (string.*IsNullOrWhiteSpace*(searchTextBox.*Text*))

{

*MessageBox*.*Show*("Моля попълнете полето за търсене на пациент", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

if (searchCriteriaListBox.*SelectedIndex* == -1)

{

*MessageBox*.*Show*("Моля изберете критерий по който да се извърши търсенето на пациент.", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

var **\_foundPatientsList** = new *List*<Patient>();

var **\_searchCriteria** = searchCriteriaListBox.*SelectedItem*.*ToString*();

var **\_searchTerm** = searchTextBox.*Text*;

switch (**\_searchCriteria**)

{

case "ЕГН":

{

var **patient** = m\_db.Patients.*SingleOrDefault*(p => p.EGN == **\_searchTerm**);

if (**patient** != null)

{

**\_foundPatientsList**.*Add*(**patient**);

}

break;

}

case "Първо име":

{

var **patients** = m\_db.Patients.*Where*(p => p.FirstName == **\_searchTerm**).*ToList*();

foreach (var **patient** in **patients**)

{

**\_foundPatientsList**.*Add*(**patient**);

}

break;

}

case "Презиме":

{

var **patients** = m\_db.Patients.*Where*(p => p.MiddleName == **\_searchTerm**).*ToList*();

foreach (var **patient** in **patients**)

{

**\_foundPatientsList**.*Add*(**patient**);

}

break;

}

case "Фамилия":

{

var **patients** = m\_db.Patients.*Where*(p => p.LastName == **\_searchTerm**).*ToList*();

foreach (var **patient** in **patients**)

{

**\_foundPatientsList**.*Add*(**patient**);

}

break;

}

case "Заболяване":

{

var **medicalCondition** = m\_db.MedicalConditions.*SingleOrDefault*(*x* => *x*.Name == **\_searchTerm**.*ToUpper*());

if (**medicalCondition** == null)

{

*MessageBox*.*Show*("Таково заболяване не съществува.", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

var **patients** = m\_db.Patients.*Include*(p=>p.MedicalCondition).*Where*(p => p.MedicalCondition.Name == **\_searchTerm**.*ToUpper*()).*ToList*();

foreach (var **patient** in **patients**)

{

**\_foundPatientsList**.*Add*(**patient**);

}

break;

}

default:

return;

}

if (**\_foundPatientsList**.*Count*() == 0)

{

foundPatientsListBox.*Items*.*Clear*();

*MessageBox*.*Show*("Няма намерени пациенти по този критерий.", "Не бяха намерени пациенти.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

return;

}

else

{

foundPatientsListBox.*Items*.*Clear*();

foreach (var **\_patient** in **\_foundPatientsList**)

{

foundPatientsListBox.*Items*.*Add*(**\_patient**.EGN);

}

*MessageBox*.*Show*("Бяха намерени " + **\_foundPatientsList**.*Count*() + " пациента.", "Има намерени пациенти.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

return;

}

}

#### Създаване на заболявания (Create Illness)

##### Описание:

Лекарите могат да добавят информация за новодиагностицирани заболявания към профилите на пациентите.

##### Примерен код:

private async void createMedicalConditionButton\_Click(object **sender**, *EventArgs* **e**)

{

if (string.*IsNullOrWhiteSpace*(medicalConditionNameTextBox.*Text*))

{

*MessageBox*.*Show*("Въведи име за новото заболяване!", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

}

else

{

// винаги запаметяваме заболяванията с главни букви -> .ToUpperInvariant()

var **\_medicalCondition** = new MedicalCondition()

{

Name = medicalConditionNameTextBox.*Text*.*ToUpperInvariant*()

};

if (m\_db.MedicalConditions.*FirstOrDefault*(*x* => *x*.Name == **\_medicalCondition**.Name) != null)

{

*MessageBox*.*Show*("Вече съществува такова заболяване!", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

}

else

{

await m\_db.MedicalConditions.*AddAsync*(**\_medicalCondition**);

await m\_db.*SaveChangesAsync*();

LoadMedicalConditionListBoxData();

*MessageBox*.*Show*("Вие успешно създадохте ново заболяване. Вече можете да създавате пациенти, които го притежават!", "Успешно създадено ново заболяване.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

}

}

}

#### Намиране на всички предписания на даден пациент (Show all prescriptions of patient):

##### Описание:

Намиране на всички предписания на даден пациент.

##### Примерен код:

public ShowPrescriptionsOfPatient(ApplicationDbContext **t\_db**, Patient **t\_patient**) :this()

{

this.m\_db = **t\_db**;

this.egnTextBox.*Text* = **t\_patient**.EGN;

var **\_allPatientPrescriptions** = **t\_db**.Prescriptions.*Where*(p => p.PatientId == **t\_patient**.Id).*ToList*();

if (**\_allPatientPrescriptions**.*Count* == 0)

{

allPatientPrescriptionsListBox.*Items*.*Add*("няма");

prescriptionTextBox.*Text* = "няма";

}

else

{

foreach (var **\_prescription** in **\_allPatientPrescriptions**)

{

allPatientPrescriptionsListBox.*Items*.*Add*(**\_prescription**.Id);

}

allPatientPrescriptionsListBox.*SelectedIndexChanged* += allPatientPrescriptionsListBox\_SelectedIndexChanged;

}

}

## Административен Панел (Admin Panel Form)

### Въведение

* Създаване на лекар: Администраторите могат да назначават нови лекари.
* Търсене на лекари.
* Специализации на лекари: Администраторите добавят или модифицират специализациите на лекарите.

### Функционалности

#### Създраване на доктор (Create Doctor)

##### Описание:

Тази функция позволява на администраторите да регистрират нови доктори в системата, като се въвеждат основни данни като име, презиме, фамилия, ЕГН и медицинската специалност на доктора.

##### Примерен код:

private async void createDoctorButton\_Click(object **sender**, *EventArgs* **e**)

{

if (!CheckIfAllInfoIsFilled())

{

*MessageBox*.*Show*("Моля попълни всички полета нужни за създаването на доктор!", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

}

else

{

var **\_userAlreadyExists** = m\_db.Users.*FirstOrDefault*(*x* => *x*.Email == emailTextBox.*Text*);

if (**\_userAlreadyExists** != null)

{

*MessageBox*.*Show*("Вече има регистриран User с такъв имейл.", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

// ако ролята doctor не съществува ще хвърли грешка,

// което е напълно коректно, тъй като предстоящата логика

// е за доктори

var **\_doctorRole** = m\_db.Roles.*Single*(*x* => *x*.Name.*ToLower*() == "doctor");

var **\_user** = new User()

{

Email = emailTextBox.*Text*,

Password = passwordTextBox.*Text*,

RoleId = **\_doctorRole**.Id

};

var **\_savedUser** = await m\_db.*AddAsync*(**\_user**);

await m\_db.*SaveChangesAsync*();

var **\_currentSelectedSpeciality** = specialityListBox.*SelectedItem*.*ToString*();

// тук отново метода .Single() би хвърлил грешка ако не бъде намерена докторска специалност

// или има повече от 1 докторска специалност с такова име

var **\_doctorSpeciality** = m\_db.DoctorSpecialities.*Single*(*x* => *x*.Name == **\_currentSelectedSpeciality**);

var **\_doctor** = new Doctor()

{

FirstName = firstNameTextBox.*Text*,

MiddleName = middleNameTextBox.*Text*,

LastName = lastNameTextBox.*Text*,

UserId = **\_savedUser**.*Entity*.Id,

DoctorSpecialityId = **\_doctorSpeciality**.Id

};

await m\_db.Doctors.*AddAsync*(**\_doctor**);

await m\_db.*SaveChangesAsync*();

foreach (var **\_control** in m\_createDoctorControls)

{

**\_control**.*Text* = "";

}

specialityListBox.*SelectedIndex* = -1;

*MessageBox*.*Show*("Вие успешно създадохте нов докторски акаунт. Този доктор вече може да влиза с въведените от вас имейл и парола.", "Успешно създаден докторски акаунт", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

}

}

#### Създаване на медицинска специалност (Create Doctor Specialty)

##### Описание:

Тази функция позволява на администраторите да регистрират нови докторски специалности в системата, като се въвежда името на специалността.

##### Примерен код:

private async void createSpecialityButton\_Click(object **sender**, *EventArgs* **e**)

{

if (string.*IsNullOrWhiteSpace*(specialityNameTextBox.*Text*))

{

*MessageBox*.*Show*("Въведи име за новата специалност!", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

}

else

{

var **\_speciality** = new DoctorSpeciality()

{

Name = specialityNameTextBox.*Text*.*ToUpperInvariant*()

};

if (m\_db.DoctorSpecialities.*FirstOrDefault*(*x* => *x*.Name == **\_speciality**.Name) != null)

{

*MessageBox*.*Show*("Вече съществува такава докторска специалност!", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

}

else

{

await m\_db.DoctorSpecialities.*AddAsync*(**\_speciality**);

await m\_db.*SaveChangesAsync*();

LoadSpecialityListBoxData();

*MessageBox*.*Show*("Вие успешно създадохте нова докторска специалност. Вече можете да създавате доктори, които я притежават!", "Успешно създадена специалност за доктори.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

}

}

}

#### Изтриване на медицинска специалност (Delete Doctor Specialty)

##### Описание:

Тази функция позволява на администраторите да изтрива докторски специалности в системата, като се избира специалността от лист от специалностти.

##### Примерен код:

private async void deleteSpecialityButton\_Click(object **sender**, *EventArgs* **e**)

{

if (specialityListBox.*SelectedIndex* == -1)

{

*MessageBox*.*Show*("Моля изберете специалност, която желаете да изтриете.", "Изберете специалност за изтриване.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

}

else

{

var **\_result** = *MessageBox*.*Show*("Ако изтриете тази докторска специалност и има доктори, които я имат, и те ще бъдат изтрити заедно с нея! Искате ли да продължите въпреки това?", "Внимание!", *MessageBoxButtons*.*YesNo*, *MessageBoxIcon*.*Warning*);

if (**\_result** == *DialogResult*.*Yes*)

{

var **\_selectedSpeciality** = m\_db.DoctorSpecialities.*Single*(*x* => *x*.Name == specialityListBox.*SelectedItem*.*ToString*());

// намери ми всички доктори с избраната специалност

var **\_allDoctorsWithSpeciality** = m\_db.Doctors

.*Include*(*d* => *d*.DoctorSpeciality)

.*Where*(*d* => *d*.DoctorSpeciality.Name == **\_selectedSpeciality**.Name)

.*ToList*();

// изтрии всички User акаунти към които сочат докторите

// заедно с тях ще се изтрие и информацията на докторите

// автоматично тъй като таблиците са ни настроени

// с CASCADE опция за изтриване

// (ако изтрием ред на таблица,а той бива използван като foreign key

// в друга таблица, то тогава там където се ползва ще бъде изтрита информацията също и накрая той )

foreach (var **\_doctor** in **\_allDoctorsWithSpeciality**)

{

var **\_doctorUserAccount** = m\_db.Users.*Single*(*u* => *u*.Id == **\_doctor**.UserId);

m\_db.Users.*Remove*(**\_doctorUserAccount**);

}

// ако нямахме горния код отгоре, а само тука този отдолу

// по CASCADE начин щяха да бъдат изтрити всички доктори с тази специалност,

// но проблема е че USER акаунтите на докторите щяха да останат в базата

// което би довело до бъгове, тъй като USERITE макар да нямат доктор на който

// да съответства (тъй като той бива изтрит) , ролята все още им е Doctor

// а ние НЕ искаме това, именно затова трием самите User акаунти

// -> от там автоматично ни се трият докторите, които сочат с foreign key към тях

// и сега вече като сме се отървали от докторите, можем да изтрием и самата докторска специалност

m\_db.DoctorSpecialities.*Remove*(**\_selectedSpeciality**);

await m\_db.*SaveChangesAsync*();

LoadSpecialityListBoxData();

}

}

}

#### Редактиране на доктор (Edit doctor)

##### Описание:

Тази функция позволява на администраторите да редактира докторите в системата.

##### Примерен код:

private async void editDoctorButton\_Click(object **sender**, *EventArgs* **e**)

{

if (!CheckIfAllInfoIsFilled())

{

*MessageBox*.*Show*("Моля попълни всички полета нужни за редактиране на доктор!", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

}

else

{

var **\_dbUser** = m\_db.Users.*Single*(*u* => *u*.Id == m\_userInfoToEdit.Id);

var **\_dbDoctor** = m\_db.Doctors.*Single*(*d* => *d*.Id == m\_userInfoToEdit.Id);

var **\_currentSelectedSpeciality** = specialityListBox.*SelectedItem*.*ToString*();

var **\_doctorSpeciality** = m\_db.DoctorSpecialities.*Single*(*x* => *x*.Name == **\_currentSelectedSpeciality**);

// редактирай User профила на доктора

// провери дали вече има друг user с такъв имейл

var **\_dbUserWithNewEmail** = m\_db.Users.*SingleOrDefault*(*u* => *u*.Email == emailTextBox.*Text*);

// ако не е NULL значи има такъв user с такъв имейл

if (**\_dbUserWithNewEmail** != null)

{

// ако Id-то на Userа който едитваме НЕ Е равно на Id-то на usera с новия имейл

// то тогава значи че това са два различни Usera

// тоест вече има регистриран User с този имейл

// и съответно не можем да променим този който редактираме на него.

if (**\_dbUser**.Id != **\_dbUserWithNewEmail**.Id)

{

*MessageBox*.*Show*("Вече има такъв регистриран потребител с този имейл. Моля пробвайте друг.", "Грешка", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

}

**\_dbUser**.Email = emailTextBox.*Text*;

**\_dbUser**.Password = passwordTextBox.*Text*;

await m\_db.*SaveChangesAsync*();

// редактирай самата информация на доктора

**\_dbDoctor**.FirstName = firstNameTextBox.*Text*;

**\_dbDoctor**.MiddleName = middleNameTextBox.*Text*;

**\_dbDoctor**.LastName = lastNameTextBox.*Text*;

**\_dbDoctor**.DoctorSpecialityId = **\_doctorSpeciality**.Id;

await m\_db.*SaveChangesAsync*();

*MessageBox*.*Show*("Вие успешно редактирахте този докторски акаунт.", "Успешно редактиран докторски акаунт", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

}

}

#### Търсене на доктор (Search Doctor)

##### Описание:

Тази функция позволява на администраторите да търси доктор в системата по дадени критерии.

##### Примерен код:

private void searchDoctorButton\_Click(object **sender**, *EventArgs* **e**)

{

if (string.*IsNullOrWhiteSpace*(searchTextBox.*Text*))

{

*MessageBox*.*Show*("Моля попълнете полето за търсене на доктор", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

if (searchCriteriaListBox.*SelectedIndex* == -1)

{

*MessageBox*.*Show*("Моля изберете критерий по който да се извърши търсенето на доктор.", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

var **\_foundDoctorsList** = new *List*<Doctor>();

var **\_searchCriteria** = searchCriteriaListBox.*SelectedItem*.*ToString*();

var **\_searchTerm** = searchTextBox.*Text*;

switch (**\_searchCriteria**)

{

case "Имейл":

{

// намери дали такъв User е регистриран с този имейл

// ползваме SingleOrDefault тъй като

// трябва да имаме само един User с такъв имейл

// ако не бъде намерен 1 такъв User, то ми върни NULL

// също така зареди ми Role полето на User-а с .Include() че да имам достъп до името на ролята на usera

var **\_user** = m\_db.Users.*Include*(*u* => *u*.Role).*SingleOrDefault*(*u* => *u*.Email == **\_searchTerm**);

if (**\_user** != null)

{

// ако ролята е доктор

if (**\_user**.Role.Name.*ToLower*() == "doctor")

{

// намери информацията за доктора, който отговаря на съответния user

// използваме Single(), тъй като връзката ни е едно към едно

// и не би трябвало да има друг доктор с Id на намерения user

// всеки един доктор отговаря на точно един user - не по малко/ не повече

var **doctor** = m\_db.Doctors.*Single*(*d* => *d*.UserId == **\_user**.Id);

**\_foundDoctorsList**.*Add*(**doctor**);

}

// ако ролята не му е доктор, не бива да го добавяме към списъка с намерени доктори

}

break;

}

case "Първо име":

{

// ползваме .Where за да намерим колекция от доктори, които притежават това име

var **\_doctors** = m\_db.Doctors.*Where*(*d* => *d*.FirstName == **\_searchTerm**).*ToList*();

if (**\_doctors** != null && **\_doctors**.*Count* != 0)

{

**\_foundDoctorsList**.*AddRange*(**\_doctors**);

}

break;

}

case "Презиме":

{

var **\_doctors** = m\_db.Doctors.*Where*(*d* => *d*.MiddleName == **\_searchTerm**).*ToList*();

if (**\_doctors** != null && **\_doctors**.*Count* != 0)

{

**\_foundDoctorsList**.*AddRange*(**\_doctors**);

}

break;

}

case "Фамилия":

{

var **\_doctors** = m\_db.Doctors.*Where*(*d* => *d*.LastName == **\_searchTerm**).*ToList*();

if (**\_doctors** != null && **\_doctors**.*Count* != 0)

{

**\_foundDoctorsList**.*AddRange*(**\_doctors**);

}

break;

}

case "Специалност":

{

// отново ползваме SingleOrDefault

// защото би трябвало да имаме не повече от 1 специалност с такова име в базата

// и защото ако не бъде намерена 1 специалност, то да ми върне NULL

var **\_doctorSpeciality** = m\_db.DoctorSpecialities.*SingleOrDefault*(*x* => *x*.Name == **\_searchTerm**.*ToUpper*());

if (**\_doctorSpeciality** == null)

{

*MessageBox*.*Show*("Такава докторска специалност не съществува.", "Грешка.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Error*);

return;

}

var **\_doctors** = m\_db.Doctors.*Include*(*d* => *d*.DoctorSpeciality).*Where*(*d* => *d*.DoctorSpeciality.Name == **\_searchTerm**.*ToUpper*()).*ToList*();

if (**\_doctors** != null && **\_doctors**.*Count* != 0)

{

**\_foundDoctorsList**.*AddRange*(**\_doctors**);

}

break;

}

default:

// тука даже не би трябвало да стигне програмата тъй като имаме

// само тези 5 неща в searchCriteriaListBox заредени

return;

}

if (**\_foundDoctorsList**.*Count*() == 0)

{

foundDoctorsListBox.*Items*.*Clear*();

*MessageBox*.*Show*("Няма намерени доктори по този критерий.", "Не бяха намерени доктори.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

return;

}

else

{

foundDoctorsListBox.*Items*.*Clear*();

foreach (var **\_doctor** in **\_foundDoctorsList**)

{

// зареди ми имейла на User профила на намерения доктор

// ползваме Single() тъй като знаем че щом имаме намерени доктори

// всеки един от тях принадлежи само на един User профил

// нямаме един User профил, който да може да се използва от много доктори

// тъй като отново -> имаме връзка едно към едно

// един User съответства на един Doctor

// и е невъзможно един и същи User да отговаря на повече от един доктор

// заради дизайна на базата (foreign key-a към User-ите е UNIQUE, не може да се повтаря)

var **\_user** = m\_db.Users.*Single*(*x* => *x*.Id == **\_doctor**.UserId);

foundDoctorsListBox.*Items*.*Add*(**\_user**.Email);

}

*MessageBox*.*Show*("Бяха намерени " + **\_foundDoctorsList**.*Count*() + " доктора.", "Има намерени доктори.", *MessageBoxButtons*.*OK*, *MessageBoxIcon*.*Information*);

return;

}

}

# Ръководство за потребителя.

## Въведение

Това ръководство е предназначено за всички потребители на медицинската информационна система, разработена с C# и Windows Forms. Ръководството обхваща всички аспекти на работа с системата, включително начални стъпки за използване, детайлни инструкции за различните модули и съвети за решаване на общи проблеми.

## Основни модули и тяхната употреба

### Модул за аутентификация (Login Form)

* **Функции**: Вход в системата с регистрираните потребителски акаунти.
* **Стъпки за използване**: Въведете вашето потребителско име и парола в съответните полета и натиснете бутона за вход. В зависимост от вашата роля (администратор или лекар), вие ще бъдете пренасочени към съответния потребителски интерфейс.

### Начален екран (Home Form)

* **Функции**: Предоставя информация за успешно входиран потребител и неговите права.
* **Навигация**: Използвайте менюто за бърз достъп до различните модули, налични за вашата роля.

### Лекарски панел (Doctor Panel Form)

* **Създаване на пациент**: Въведете необходимите данни за пациента в предоставените полета и натиснете 'Запази'.
* **Търсене на пациент**: Въведете критерии за търсене и използвайте бутона за търсене, за да намерите пациентски записи.
* **Регистрация на заболявания**: След като изберете пациент, можете да добавите информация за диагностицирани заболявания.

### Административен панел (Admin Panel Form)

* **Създаване на лекар**: Попълнете данните за новия лекар и натиснете 'Добави'.
* **Управление на специалности**: Можете да добавите, редактирате или изтриете медицински специалности.
* **Търсене на лекари**: Въведете критерии и търсете лекари в системата.

# Тестване на приложението

## Методи на Тестване

За да се гарантира надеждността и ефективността на медицинската информационна система, бяха приложени следните методи на тестване:

* **Функционално тестване**: Проверка дали всяка функция на системата работи както е очаквано. Това включва тестване на вход и изход във всички форми и модули.
* **Интеграционно тестване**: Тестване на взаимодействието между различните модули на системата, за да се уверим, че те коректно обменят данни и работят хармонично заедно.

## Резултати от Тестването

Всички основни функционалности работят според изискванията.

## Изводи

Системата демонстрира висока функционалност и потенциал за употреба в реална среда.

## Възможности за Развитие

* **Оптимизация на производителността**: Имплементиране на кеширане и по-ефективни заявки към базата данни.
* **Подобряване на употребяемостта**: Редизайн на интерфейса с фокус върху интуитивност и минимализъм.
* **Разширяване на функционалностите**: Въвеждане на нови модули, като например мобилно приложение за достъп на пациентите до техните медицински данни.
* **Интеграция с други системи**: Създаване на API за взаимодействие с други медицински и административни системи.

# Използвана Литература

## Общодостъпни Ресурси и Техническа Документация

За разработката на медицинската информационна система основно се разчиташе на следните видове ресурси:

* Официална Документация на Microsoft за C# и .NET Framework: Използвана за дълбоко разбиране на езика за програмиране C# и платформата .NET, които са основа на системата.
  + Microsoft C# Documentation
  + Microsoft .NET Documentation
* Документация на Windows Forms: Съдържа информация за използването на Windows Forms за създаване на графични потребителски интерфейси.
  + Windows Forms Documentation
* Stack Overflow и други форуми за програмисти: Често използвани за намиране на решения на специфични проблеми или за получаване на съвети относно най-добри практики.
  + Stack Overflow