ДИПЛОМЕН ПРОЕКТ ЗА ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ

по професия код 481030 „ Приложен програмист“ специалност код 4810301 Приложно програмиране“

Тема: „ Самоподдържаща се зелена оранжерия “

Автор:

Жаклин Олегова Йорданова , клас XII В Ръководител:

Красимир Ватев

Бургас

СЪДЪРЖАНИЕ

1. [Увод 3](#_bookmark0)
2. [Цели и обхват на софтуерното приложение 4](#_bookmark1)
3. [Анализ на решението 5](#_bookmark2)
   1. [Потребителски изисквания и работен процес 5](#_bookmark3)
   2. [Примерен потреб ителски интерфейс 7](#_bookmark4)
   3. [Диаграми на анализа 9](#_bookmark5)
   4. [Модел на съдържанието / данните 11](#_bookmark6)

Error! Bookmark not defined.

1. [Дизайн 13](#_bookmark7) 
   1. [Реализация на архитектурата на приложение то 17](#_bookmark8)
   2. [Описание на слоевете, предназначението им, библиотеки и](#_bookmark9) [методи включени в съответния слой 18](#_bookmark9)
   3. [Организация и код на заявките към база от данни 27](#_bookmark10)
   4. [Наличие на потребителски интерфейс (конзолен, графичен,](#_bookmark11) [уеб) 28](#_bookmark11)
2. [Ефективност и бързодействие на решението 31](#_bookmark12)
3. [Тестване 32](#_bookmark13)
4. [Заключение и възможно бъдещо развитие 35](#_bookmark14)
5. [Използвани литературни източници и Уеб сайтове 36](#_bookmark15)
6. [Критерии и показатели за оценяване 37](#_bookmark16)

# Увод

Умната оранжерия осигурява почти перфектни условия за създаване и отглеждане на реколта, по всяко време от годината. Тя бива независима от метеорологичните условия навън и бива по-плодородна от тази, която се отглежда навън. За да се случи това, е нужно да се осигурят благоприятни условия за растеж – светлина, благоприятна температура , добра вентилация, осигурено напояване. Технологията, която се свързва с централното ядро, запазва данните на текущите условия и на тяхна основа взима решение относно предстоящите процеси на отглеждане. Тези условия могат да бъдат поддържани и без човешка намеса.

Технологията изцяло контролира цялата обработка на градините. Сензорите за слънчева светлина, спомагат включването на осветление с предварително зададен цветен спектър, в случай на липса на влажност на почвата, сензорите предизвикат напояване. Под технологичния контрол оранжериите ще предоставят богата и висококачествена реколта, която ще улесни трудноподвижните и хората, които нямат физическата възможност да се грижат всекидневно за добива.

Модел - Изглед - Контролери ( MVC) e модел на проектиране, кой то е особено полезен за внедряване на потребителски интерфейси по такъв начин, чe да разделя софтуера на отделни задачи, като тези, предназначени за обработка и съхранение на данни ( модел), бизнес логика, маршрутизиране на потребителски команди и т . н . , като по този начин прави повече от кода ви е наличен за тестване на единици. ” С него кодът става по - подреден и по - оптимизиран. Така много по-бързо и по-лесно, колегите могат да се ориентират в написаното или потребителя, при нужда от разясняване и демонстрация.

# Цели и обхват на софтуерното приложение

* + Оптимизираното на селскостопанското производство в оранжерии, допринася за забавяне на изменението на климата чрез спестяване на вода и енергия.
  + Този проект обхваща най-вече трудноподвижните по-възрастни хора и семействата, които работят в големите градове, и нямат достатъчно време, за да гледат оранжерия.
  + Предоставя модел на интелигентна оранжерия, която помага на трудномобилни и работещите всекидневно в големия град да извършват работата в оранжериите автоматично, без да се изисква постоянно наблюдение. Оранжерията, представляваща затворена структура, предпазва растенията от екстремни метеорологични условия, а именно: вятър, градушка, ултравиолетови лъчения и атаки на насекоми и вредители.
  + Напояването на селскостопанското поле се извършва с автоматично капково напояване, което работи според съответно зададения праг на влажност на почвата, така че да се подава оптимално количество вода върху растенията. Въз основа на данни от здравната карта на почвата, се вкарват подходящо количество азот, фосфор, калий и други минерали. На растенията се осигурява и необходимата светлина с дължина на вълната през нощта с помощта на растящи светлини. Температурата и влажността на въздуха се контролират от сензори за влажност и температура и за контрол.
  + Този проект описва дизайна на система за наблюдение и контрол на оранжерии, базирана на IOT, използваща Arduino. Първият проблем, който се преодолява в системата, е, че не винаги е необходимо човек да присъства в оранжерията. Растенията в оранжерията се отглеждат в контролирана среда. Температурните разлики могат да навредят на растенията.
  + Понякога фермерите не могат да предскажат какви действия трябва да бъдат предприети, за да контролират околната среда и могат да вземат грешни решения, причинявайки по този начин повече вреда на растенията в оранжерията. Проектът му позволява да взима правилни решения, като предоставя на фермера точна информация за състоянието на сензорите чрез уеб сървъра на IOT. Така тази система помага на фермера да контролира оранжерията от отдалечени места.

# Анализ на решението

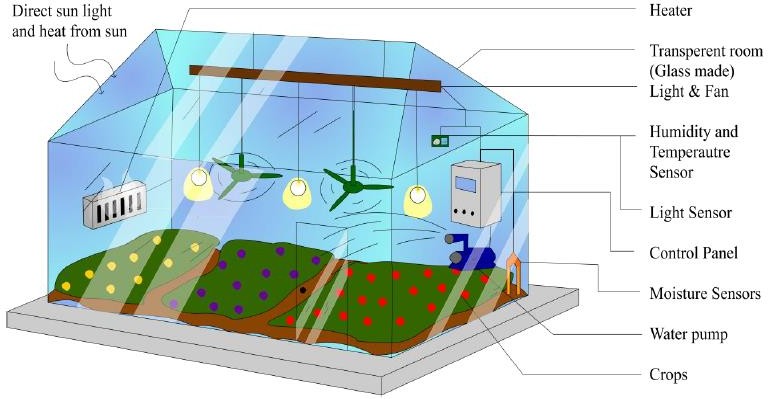
## 1 Потребителски изисквания и работен процес

Оборудвани с модерни сензорни и комуникационни технологии, интелигентните оранжерии автоматично улавят и доставят информация 24/7 за околната среда и реколтата. Събраните данни се подават в „ Интернет на нещата“(IoT) платформа и след това се обработват в базата данни и се подават към потребителя.

Потребителят влиза в приложението и първото, което трябва да направи е да се регистрира в регистрационната форма, откъдето данните въведени в нея се пренасочват и запазват в базата данни. При вече направена регистрация, потребителят е свободен да напише своето име и парола във формата за вход и да натисне бутона за влизане. При правилно въведени данни, пред потребителя се добавя прозорец, който е свързан със създаването на градина. При положение, че вече имате създадена градина, ще имате право да изберете нея. Ако все пак сега създавате градина, след натискането на бутона, се отваря прозорец, в който трябва да се въведат основни данни за градината. След натискането на бутона „ Изпрати“ (Submit), се отваря вече крайният прозорец, в който можете да види те информацията за вашата градина и данните, изпратени от Ардуиното, с които вие можете да

следите състоянието на оранжерията и начина, по който можете да я подобрите в нейното отглеждане.

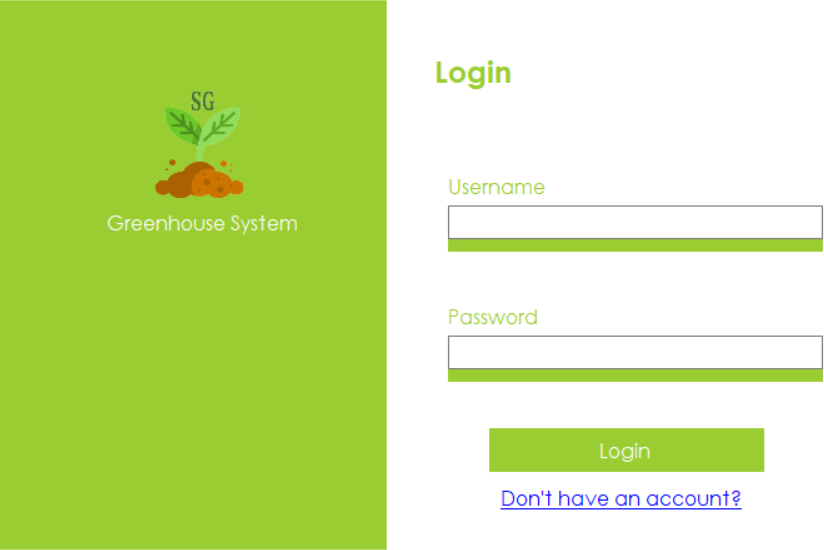
Данните се актуализират чрез сензорите на Ардуиното. Оттам те се обработват и се пращат към потребителското приложение, в което се изобразяват под формата на таблица. От там потребителят вижда състоянието на своята градина и как може да регулира условията.



*Ф и г ур а 1*

На фигура 1 е изобразена схема за проекта „ Самоподдържаща се оранжерия“, в която се показва цялостната идея на проекта и неговата функционалност. Изобразена е легенда, с която се показват различните видове сензори на Ардуиното, водна помпа, нагревател и пр.

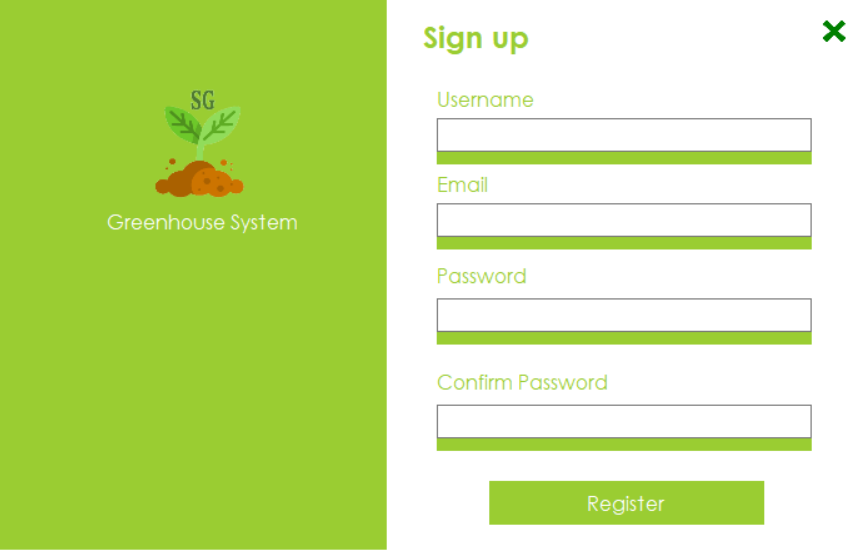
## 2 Примерен потребителски интерфейс



*Ф и г ур а 2*

На фигура 2 се вижда потребителският интерфейс на страница за форма за вход на потребителя. В нея се съдържат полетата за Потребителско име( Username) и парола( Password). При правилно въведени данни и при натискане на бутона „ Вход“( Login) ви пренасочва към страница за добавяне на градина. Тестът, в който се съдържа линк с въпросът

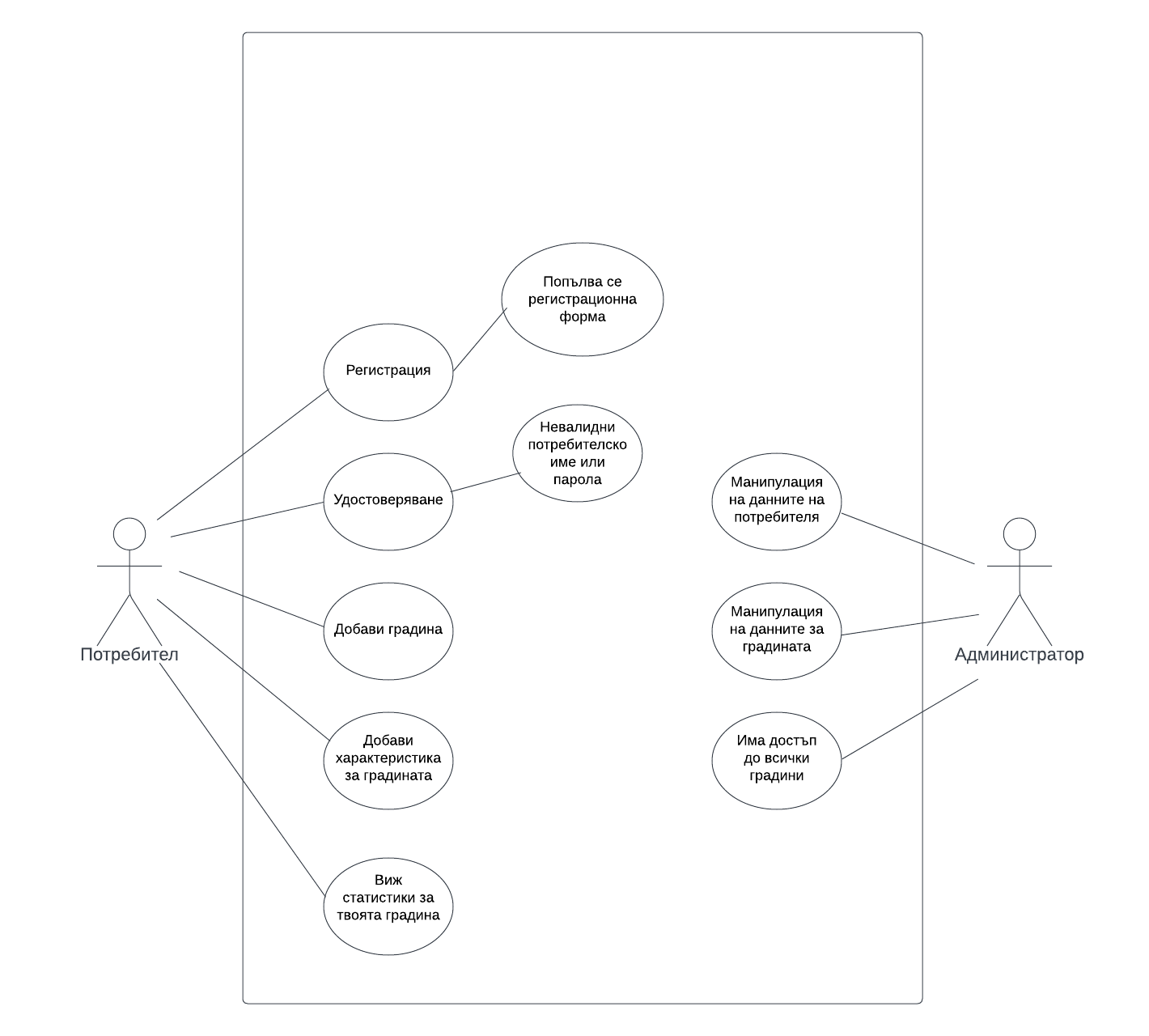
„ Не притежавате акаунт?“(Don’t have an account?) пренасочва потребителя към регистрационна форма.



*Ф и г у р а 3*

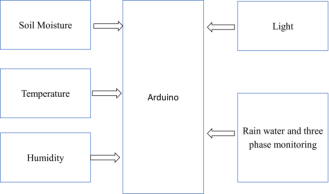
На фигура 3 виждате регистрационната форма на потребителския интерфейс. Има полета за потребителско име ( Username) , имейл ( Email), парола( Password) и потвърждаване на паролата( Confirm Password). На фигурата присъства и бутона „ Регистрирай се“( Register).

## 3 Диаграми на анализа



*Ф и г ур а 4*

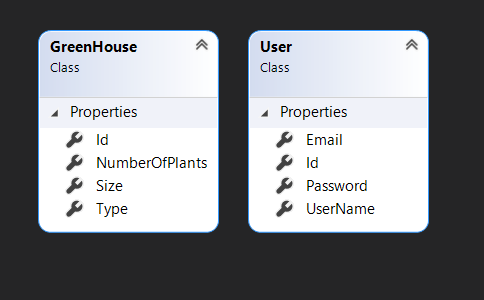
На фигура 4 се изобразява схема на целия проект чрез диаграма на случаите(Use-Case), в която се показва потребителят, различните сензори за различните характеристики. Тази диаграма на случаите на използване би илюстрирала различните взаимодействия между участниците и системните компоненти в интелигентна оранжерия на Arduino, както и допълнителната функционалност, предоставена от ролята на администратор. Това ще демонстрира способността на системата да осигури ефикасно и ефективно управление на оранжерийната среда, като същевременно предоставя инструменти за поддръжка и отстраняване на проблеми.



*Ф и г ур а 5*

На фигура 5 се показва схема на Ардуиното, заедно със сензорите, които подават информация към базата данни и се изписват в приложението

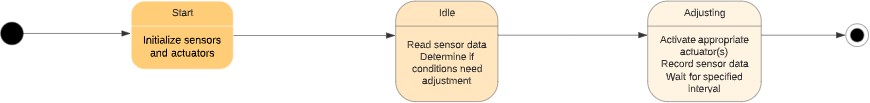
## 4 Модел на съдържанието / данните



*Ф и г ур а 6*

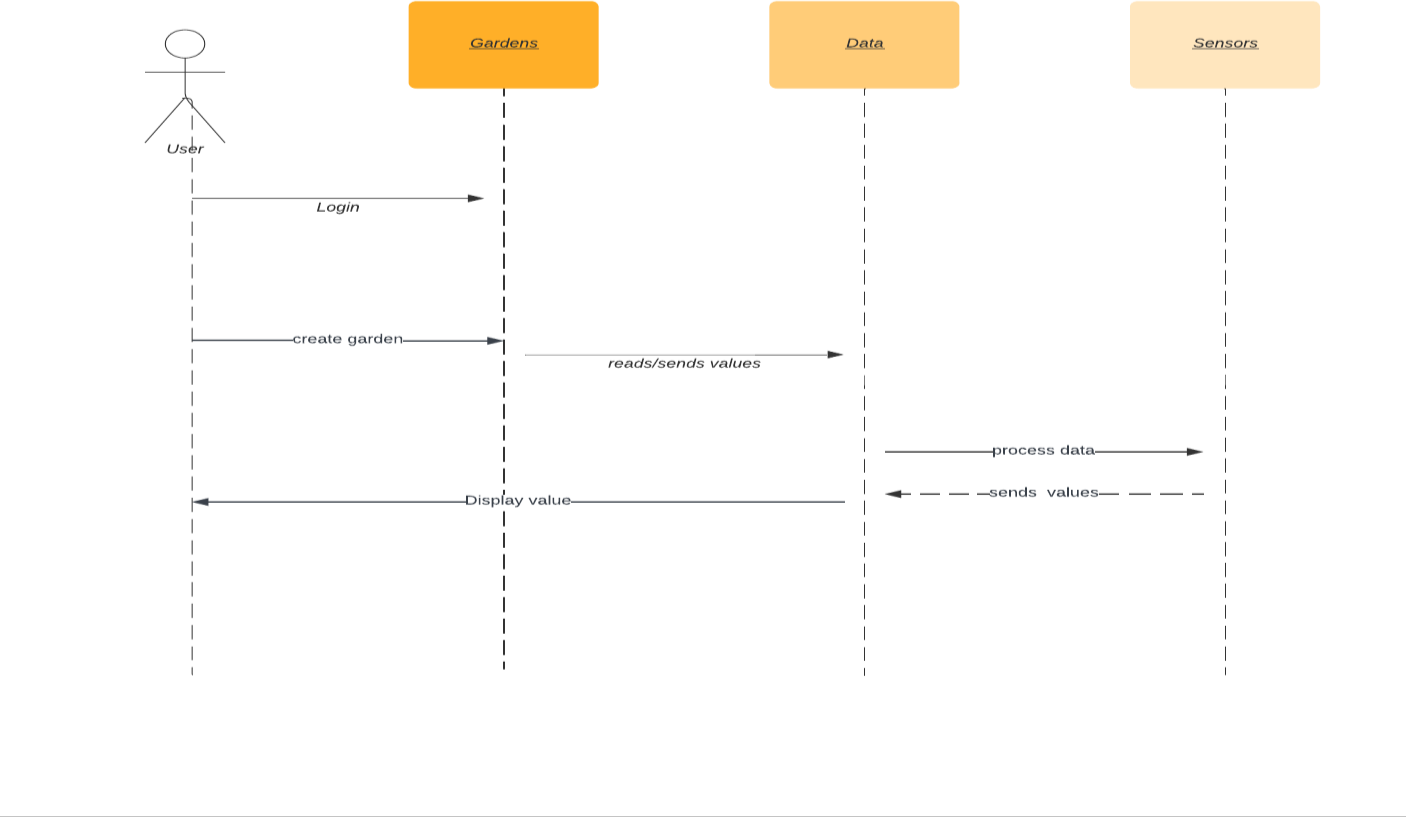
На фигура 6 са представени двата основни класа на проекта „ Зелена оранжерия“(Greenhouse) и „ Потребител“(User). Показват се свойствата им, в първия клас „ Зелена оранжерия“ присъстват „ Идентификация“ (Id),“ Брой растения“ (Number of plants) , „ Големина“(Size) и „ Тип на градината“(Type). Вторият клас „ Потребител“ ( User) съдържа в себе си свойствата –

„ Имейл“(Email), „ Идентификация“, (Id) „ Парола“ (Password) и „ Потребителско име“( Username).



*Ф и г ур а 7*

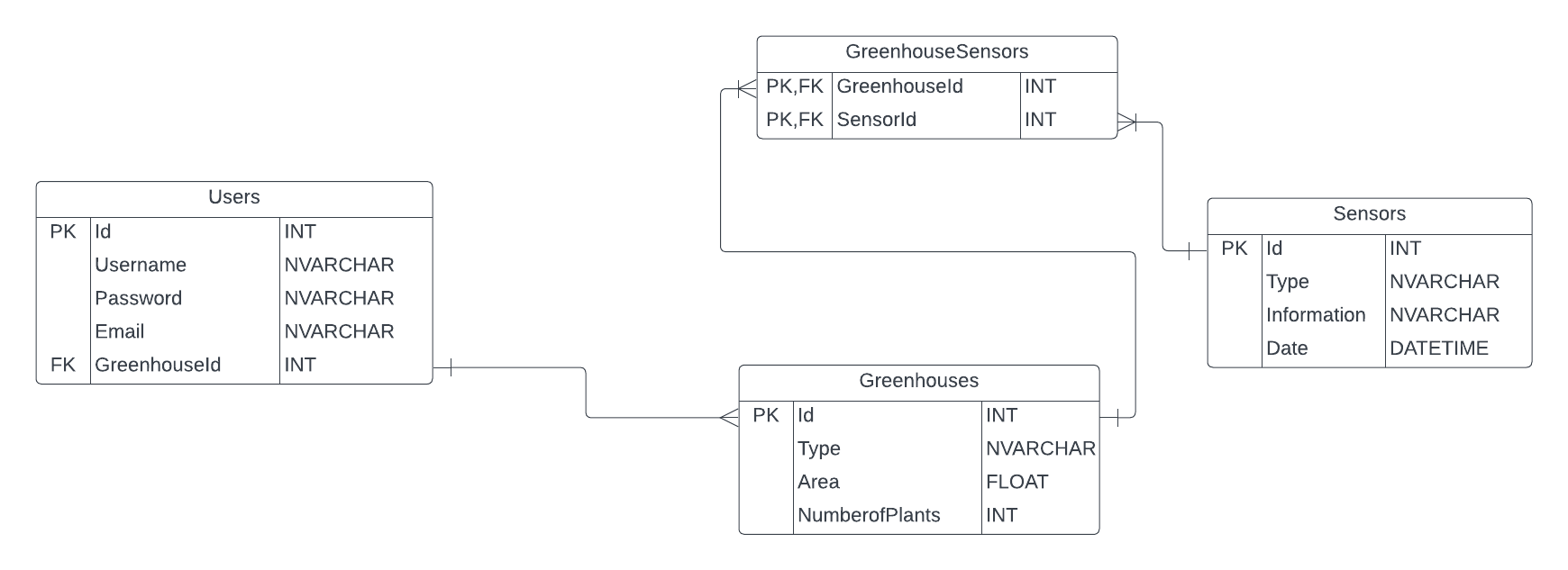
В тази диаграма системата стартира в състояние " Старт" ( Start), където инициализира сензорите и изпълнителните механизми. След това той преминава в състояние " Неактивен " ( Idle), където чете данни от сензора и определя дали условията се нуждаят от корекция. Ако условията се нуждаят от коригиране, той преминава в състояние " Настройка" (Adjusting), където активира подходящият задвижващ механизъм(и), записва данни от сензора и изчаква определен интервал. След интервала той преминава обратно в състояние „ Неактивен“ (Idle), за да продължи наблюдението на оранжерийната среда . Имайте предвид, че тази диаграма на машината на състоянието е опростен пример и може да не обхваща цялата функционалност на интелигентна оранжерийна система в реалния свят.



*Ф и г ур а 8*

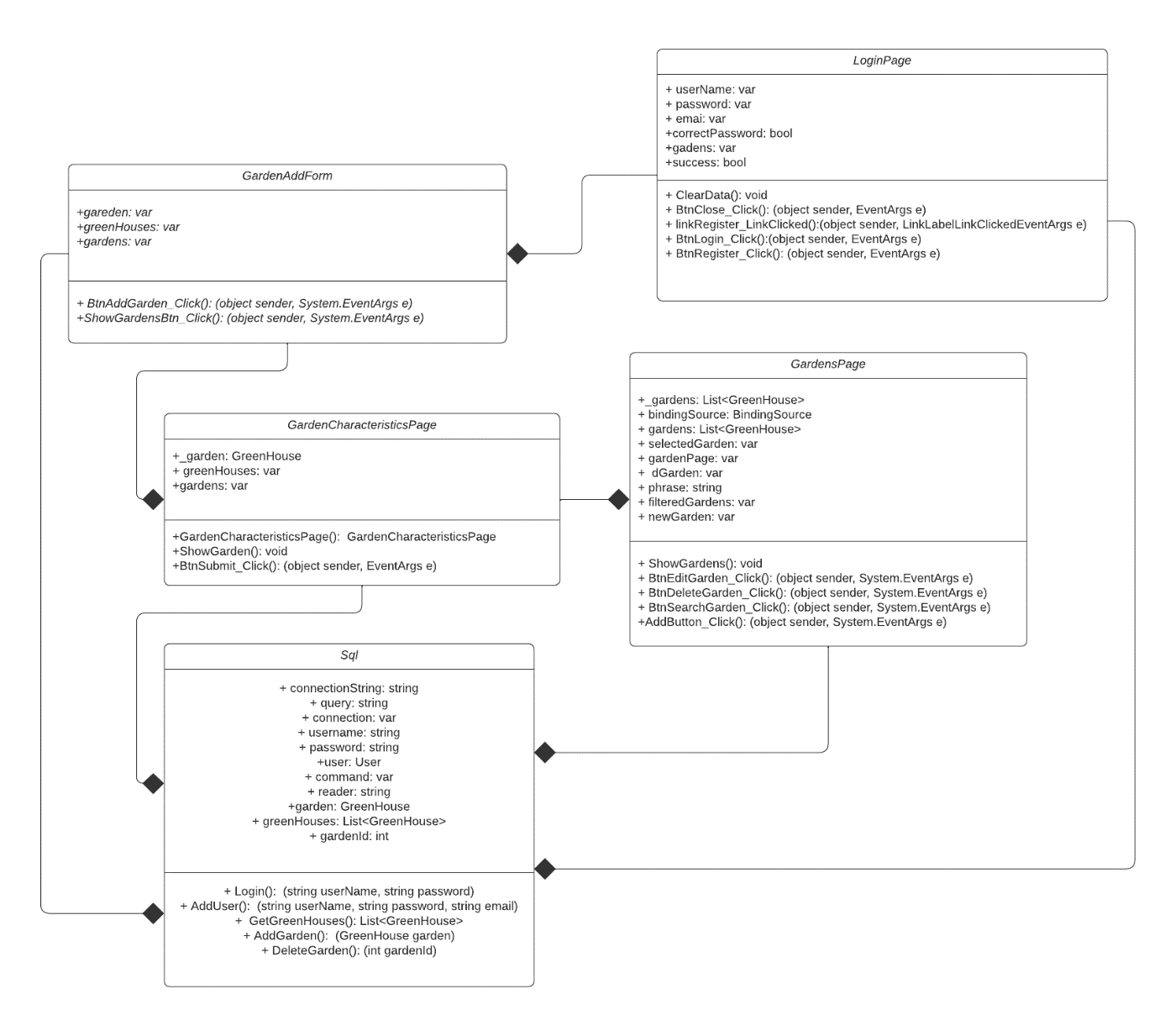
В тази комуникационна диаграма потребителят наблюдава за желаните условия за парниковата среда. Arduino чете данни от сензора и потребителят определя дали има нужда от корекция. Ако е така, той активира съответния сензор(и) и записва данни от сензора. След това Arduino изчаква определен интервал, преди да повтори процеса.

Тази комуникационна диаграма представя потока от комуникация и взаимодействия между потребителя, Arduino, сензорите и задвижващите механизми.



Фигура 9

На фигура 10 таблицата с потребители с полета идентификатор(Id), потребителско име(username), парола(password), имейл(email) и идентификатор на градината(greenhouseId) (външен ключ, препращащ към таблицата на парниците). Таблица за оранжерии(Greenhouses) съдържа в себе си идентификатор на полета(Id), тип оранжерия(Type), площ(Area) и брой растения(NumberOfPlants). Tаблицата за сензори включва в себе си полетата индентификатор(Id), тип(Type), информация(Information) и дата(Date). Таблицата GreenhouseSensors, служи като мост между таблиците за сензорите(Sensors) и оранжериите(Greenhouses). Той има два външни ключа, sensor\_id и greenhouse\_id, като и двата препращат към съответните си таблици. Таблицата GreenhouseSensors позволява връзка много към много между сензорите и таблиците за оранжерии, тъй като една оранжерия може да има множество сензори и един сензор може да се използва в множество оранжерии.



Фигура 10

Диаграмата на класовете представлява интелигентна оранжерийна система Arduino с няколко класа като *LoginPage, AddGardenForm, GardenCharacteristicspage, GardensPage и Sql*.

Класът *LoginPage* има методи като *ClearData(), BtnClose\_Click(), linkRegister\_LinkClicked(), BtnLogin\_Click() и BtnRegister\_Click().* Методът *ClearData()* изчиства данните на страницата за вход, методът *BtnClose\_Click()* затваря страницата за вход, методът *linkRegister\_LinkClicked()* насочва потребителя към страницата за регистрация, методът *BtnLogin\_Click()* регистрира потребителя в системата, а методът *BtnRegister\_Click()* регистрира нов потребител.

Класът *AddGardenForm* има методи като *BtnAddGarden\_Click()* и *ShowGardensBtn\_Click().* Методът *BtnAddGarden\_Click()* добавя нова градина към системата, а методът *ShowGardensBtn\_Click()* показва всички налични градини в системата.

Класът *GardenCharacteristicspage* има методи като *GardenCharacteristicsPage*(), *ShowGarden*() и *BtnSubmit\_Click().* Методът *GardenCharacteristicsPage*() инициализира страницата с характеристики на градината, методът *ShowGarden()* показва подробностите за градината, а методът *BtnSubmit\_Click()* записва характеристиките на градината в системата.

Класът *GardensPage* има методи като *ShowGardens(), BtnEditGarden\_Click(), BtnDeleteGarden\_Click(), BtnSearchGarden\_Click()* и *AddButton\_Click().* Методът *ShowGardens()* показва всички налични градини в системата, методът *BtnEditGarden\_Click()* редактира подробностите за градината, методът *BtnDeleteGarden\_Click()* изтрива градина от системата, методът *BtnSearchGarden\_Click()* търси градина, а методът *AddButton\_Click()* добавя нова градина към системата.

Класът *Sql* има методи като *Login(), AddUser(), GetGreenHouses(), AddGarden() и DeleteGarden().* Методът *Login()* удостоверява потребителя, методът *AddUser()* добавя нов потребител към системата, методът *GetGreenHouses()* извлича всички налични градини в системата, методът *AddGarden()* добавя нова градина към системата и *DeleteGarden()* метод изтрива градина от системата.

Като цяло класовата диаграма представя различните класове и методи, използвани в интелигентната оранжерийна система Arduino за удостоверяване на потребителя, добавяне в градината, изтриване и показване.

# Дизайн

Проектът е написан в платформата Visual Studio 2022 . Той е реализиран с помощта на сензори и части Arduino UNO, които се програмират на езика С, кодът е написан в платформата на Arduino. За приложението е използвано Windows Form App на езика C#, в коeто сa изобразени данните от сензорите за състоянието на градината и развитието на растенията. С Microsoft SQL Management Studio 18 e създадена базата, преведена до 3-та нормална форма и чрез библиотека, вкарана в кода на приложението . С Win Forms се свър зват двата компонента. Частите на Arduino са програмирани и сглобени, така че „ умната градина“ да се „ самоотглежда“, без човешка намеса.

Дизайнът е направен чрез Windows Form App , чрез бутони( buttons), етикети( l abels), табла( panels), текстови кутии( t extboxes) и пр. Цветовете на приложението са свързани и с темата на проекта, това значи, че преобладават зеленият и бежовият цвят. Софтуерният архитектурен шаблон на приложението е Модел - визуален Контролер ( Model - view- Controllers ) .

private void BtnLogin\_Click(object sender, EventArgs e)

{

string username = UserNameBox.Text; string password = PasswordBox.Text;

bool success = DataBase.Sql.Login(username, password);

if (success)

{

var gardens = DataBase.Sql.GetGreenHouses();

if (gardens != null && gardens.Count > 0)

{

}

else

{

}

}

else

{

Hide();

var gardensPage = new GardensPage(gardens); gardensPage.ShowDialog();

Hide();

var greenhouse = new AddGardenPage(); greenhouse.ShowDialog();

MessageBox.Show("Incorrect user name or password!");

}

}

В кода горе се изобразява формата за вход или т . нар. “Login form“ , където се показва бутонът, проверява се дали са верни потребителското име и паролата на потребителят. При правилно изписване на данните на потребителя преминава към следващата страница, в противен случай се показва съобщение, на което се изписва: „ Неправилно потребителско име или парола.“( “Incorrect username or password.”)

private void BtnSubmit\_Click(object sender, EventArgs e)

{

\_garden.Type = (TypeOfGreenHouse)TypeOfGardensComboBox.SelectedIndex;

\_garden.NumberOfPlants = int.Parse(TxtNumberOfPlants.Text);

\_garden.Size = double.Parse(TxtSizeOfArea.Text);

if (DataBase.Sql.AddGarden(\_garden))

{

var greenHouses = DataBase.Sql.GetGreenHouses(); var gardens = new GardensPage(greenHouses); Close();

gardens.Show();

}

}

Горе в кода виждате, който е за страницата, в която се създава градината чрез бутона „ Добави градина“ (Add Garden), при вече добавена такава може да се достъпи чрез бутон a „ Покажи моите

градини“( Show my gardens). Ако методът *Add Garden( \_garden)* върне t rue, той извлича списък с оранжерии от базата данни с помощта на *Data Base. Sql. Get Green Houses()* , създава нов обект *Gardens Page* със списъка с оранжерии като параметър, затваря текущия формуляр и показва формуляра *Gardens Page.*

private void BtnEditGarden\_Click(object sender, System.EventArgs e)

{

Close();

var selectedGarden =

\_gardens[GardenDataGrid.CurrentCell.RowIndex]; var gardenPage = new

GardenCharacteristicsPage(selectedGarden); gardenPage.Show();

}

private void BtnDeleteGarden\_Click(object sender, System.EventArgs e)

{

var selectedGarden =

\_gardens[GardenDataGrid.CurrentCell.RowIndex]; DataBase.Sql.DeleteGarden(selectedGarden.Id);

var dGarden = \_gardens.FirstOrDefault(g => g.Id == selectedGarden.Id);

GardenDataGrid.DataSource = null;

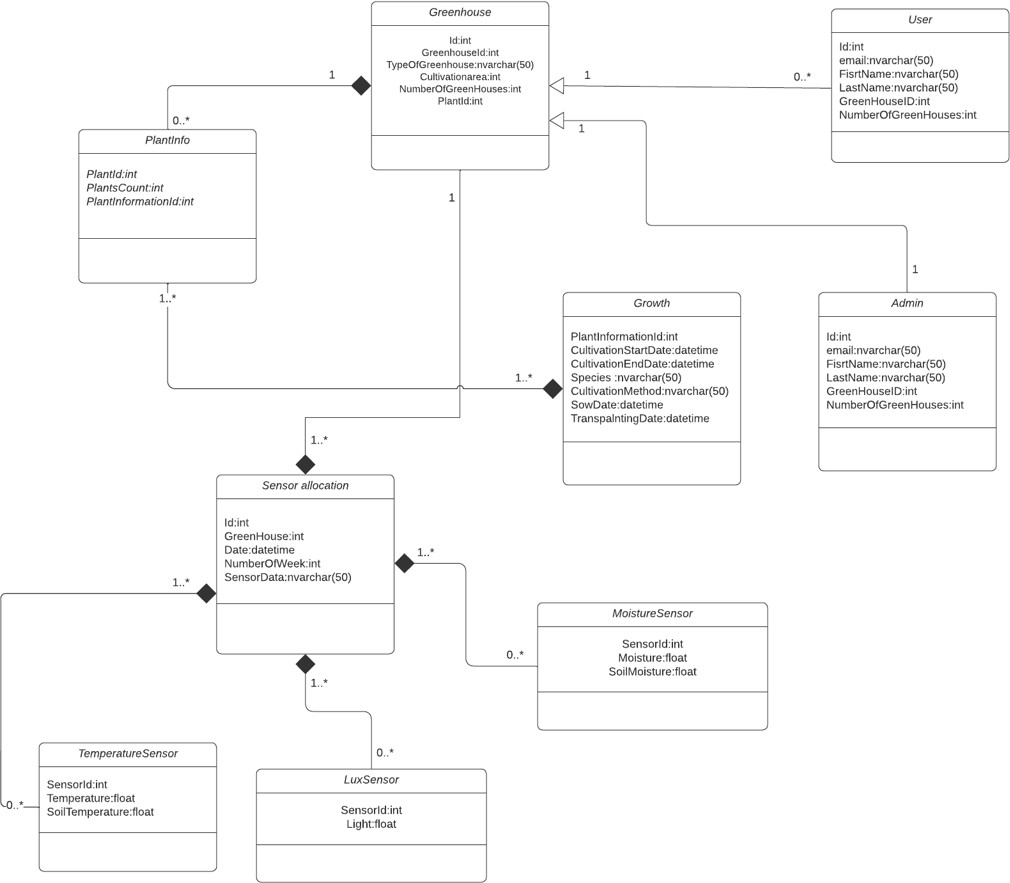
\_gardens.Remove(dGarden); ShowGardens();

}

Горе в кода се показват функции от типа CRUD( CREATE, READ, UPDATE, DELETE), за създаване, четене, актуализация и изтриване.

Вижда се в кода, че се използват и полетата, създадени в базата данни. Които се проверяват за нулеви стойности и се изтриват след това.

## 1 Реализация на архитектурата на приложението

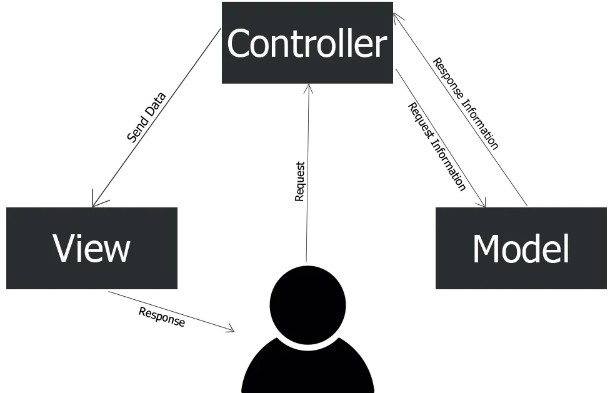


*Ф и г ур а 11*

Както се вижда от диаграмата на фигура 9 , има създаден един голям клас Greenhouse , който събира класовете Администратор( Admin) и Потребител( User), които имат еднакъв достъп до отделните класове свързани с растежа, отглеждането, сензорите, информацията за растенията, които се отглеждат в самоподдържащата се оранжерия. Изобразява се информацията за растенията, растежа и тяхното състояние след постъпването в конзолното приложение. Потребителят чрез акаунт в приложението, може да създаде или провери своята градина от разстояние. Така потребителят може да разбере как да подобри условията в оранжерията си .

## 2 Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой.

Моделът MVP, по който е създаден проектът се състои от 3 слоя.

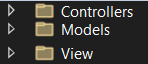


*Ф и г у р а 10*

На фигура 10 се показва схемата, която представя реализацията и функционалността на трислойния модел Модел - Изглед - Контролери( MVC). Схемата показва, че се използват събития ( Events), за функционалността.

* + - В папката Модели( Models ) се съхраняват данните, които трябва да се показват или да се действа в потребителския интерфейс. Той също така определя бизнес правила за данни, което представлява как данните могат да бъдат променяни и манипулирани.
    - Изгледът( Views) e пасивен потребителски интерфейс, който показва данните н а модела( Models) и насочва събития, инициирани от потребителя, като команди за щракване н а мишката, към контролерите ( Controllers ), за да действа спрямо тези данни.
    - В софтуерната архитектура Model - View- Controller ( MVC) контролерите играят решаваща роля в управлението на

потребителските взаимодействия със системата. Контролерът отговаря за получаването на заявки от потребителя или от външни източници, обработката им и извършването на съответните промени в компонентите на модела (Model) или изгледа (View) на системата. Контролерите (Controllers) действат като посредници между потребителя и бизнес логиката на системата. Когато потребител ( User) взаимодейства с уеб приложение или която и да е софтуерна система, базирана на модела MVC, контролерът ( Controller) получава заявката и решава как да я обработи. Контролерът ( Controller) може да извлече данни от модела (Model) , да го модифицира и след това да актуализира изгледа ( View), за да отрази промените. Алтернативно, той може да инициира действия върху модела ( Model) или да актуализира изгледа ( View) директно, без да взаимодейства с модела( Model) .



*Ф и г у р а 11*

На фигура 11 се изобразяват папките „ Модели“ ( Models), в които се съдържат данните, които трябва да се показват или да се действа в потребителския интерфейс. В папката „ Изглед“( Views ) се изобразява потребителски интерфейс, който показва данните на модела ( Models) и насочва събития, инициирани от потребителя . Папката

„ Контролери“( Controllers ) e създадена з а обработката н а всички събития oт потребителския интерфейс oт името на изгледа.

Методът *BtnDeleteGarden\_Click* е кодов блок, който се задейства, когато потребителят щракне върху бутон с име *BtnDeleteGarden* . Целта на този код е да изтрие избрана градина от източник на данни и съответно да актуализира потребителския интерфейс.

Кодът първо идентифицира текущо избраната градина от списък с градини, \_ *gardens* , въз основа на индекса на текущо избрания ред в мрежа с данни, *Garden Data Grid* . След това извиква метод с име Delete Garden от клас с име Sql в база данни, като предава идентификатора на избраната градина, за да я изтрие от източника на данни.

След това кодът търси изтритата градина в списъка *\_gardens* и я премахва от списъка. Той също така изчиства източника на данни на *Garden Data Grid* , за да гарантира, че актуализираният списък с градини се показва на потребителя.

Накрая се извиква методът *ShowGardens* , който актуализира потребителския интерфейс с актуализирания списък с градини.

В обобщение, методът *BtnDeleteGarden\_Click* обработва логиката на изтриване на избрана градина от източник на данни и съответно актуализиране на потребителския интерфейс.

Целта на метода *BtnEditGarden\_Click* е да се справи с логиката за редактиране на характеристиките на избрана градина и да отвори нов формуляр за потребителя, за да направи промените. Този метод се задейства, когато потребителят щракне върху бутон с име *Btn Edit Garden* .

Първият ред на метода извиква метода *Close* , за да затвори текущия формуляр.

Вторият ред извлича избраната градина от списъка \_ *gardens* въз основа на индекса на текущо избрания ред в мрежа с данни, наречена

*Garden Data Grid* , и я присвоява на променлива, наречена *selected Garden* .

Третият ред създава нов екземпляр на формуляр *Garden Characteristics Page* , като предава избраната градина като параметър за предварително попълване на формуляра със съществуващите характеристики на градината.

Четвъртият ред показва новия формуляр *gardenPage* на потребителя, което му позволява да редактира характеристиките на градината.

Като цяло този кодов блок е отговорен за обработката на редактирането на характеристиките на избрана градина и отварянето на нов формуляр за потребителя, за да на прави промените.

Целта на метода *BtnSearch\_Garden\_Click* е да управлява логиката за търсене на градина въз основа на предоставена от потребителя фраза за търсене и актуализиране на потребителския интерфейс с резултатите от търсенето. Този метод се задейства, когато потребителят кликне върху бутон с име *Btn Search Garden* .

Първият ред на метода извлича въведената от потребителя фраза за търсене от текстово поле с име *Search Box* и я присвоява на променлива с име фраза.

Вторият ред проверява дали фразата за търсене не е нула или празна.

Ако фразата за търсене не е нулева или празна, методът филтрира списъка *\_gardens* въз основа на това дали фразата за търсене съвпада с идентификатора на градината, типа, размера или броя на растенията. След това присвоява филтрирания с писък на променлива *filteredGardens* и задава източника на данни на обекта *bindingSource* към този филтриран списък.

След това методът задава източника на данни на мрежа с данни, наречена *GardenDataGrid*, към обекта *bindingSource* и опреснява мрежата с данни, за да покаже резултатите от търсенето.

Ако фразата за търсене е нулева или празна, методът извиква метода *ShowGardens* , който актуализира потребителския интерфейс с оригиналния списък с градини.

Като цяло този кодов блок е отговорен за обработката на търсенето в градина въз основа на предоставена от потребителя фраза за търсене и актуализирането на потребителския интерфейс с резултатите от търсенето.

Целта на метода *Btn\_Login\_Click* е да управлява логиката на потребителя да влезе в приложе нието. Този метод се задейства, когато потребителят щракне върху бутон с име *Btn\_ Login* .

Първите два реда на метода извличат потребителското име и паролата, въведени от потребителя от текстовите полета, наречени съответно *User Name Box* и *Password Box* .

Третият ред извиква метод с име *Login* от *DataBase. Sql* обект, като предава потребителското име и паролата като параметри. Този метод връща булева стойност, указваща дали влизането е било успешно или не, която се съхранява в променлива за успех.

Ако влизането е било успешно, методът извлича списък с оранжерии от базата данни с помощта на метода *GetGreenHouses* от обекта *Data Base. Sql.* Ако в списъка има оранжерии, методът създава нов екземпляр на формуляр *Gardens Page* , като предава списъка с градини като параметър и показ ва формуляра на потребителя с помощта на метода *Show Dialog* .

Ако в списъка няма оранжерии, методът създава нов екземпляр на формуляр *Garden Characteristics Page* , като предава нов обект

*GreenHouse* като параметър и показва формуляра на потребителя с помощта на метода *Show Dialog* .

Ако влизането не е било успешно, методът показва прозорец със съобщение на потребителя, показващ, че потребителското име или паролата са неправилни.

Като цяло този кодов блок е отговорен за обработката на процеса на влизане на потребителя, извличане на подходящите данни от базата данни и отваряне на подходящия формуляр въз основа на успеха на влизането и наличността на данните.

Целта на метода *BtnRegister\_Click* е да управлява логиката за регистриране на нов потребител в приложението. Този метод се задейства, когато потребителят щракне върху бутон с име *Btn Register* .

Първите два реда на метода извличат потребителското име и имейла, въведени от потребителя от текстовите полета, наречени съответно *Txt Username R* и *Txt Email R* .

Третият ред проверява дали паролата, въведена от потребителя в текстовото поле *TxtPasswordR*, съвпада с паролата, въведена в текстовото поле *TxtCofirmPassR*, и съхранява резултата в булева променлива с име *correct Password* .

Ако паролите съвпадат, методът извлича паролата, въведе на от потребителя в текстовото поле *TxtPasswordR* , и извиква метод с име *Add User* от *DataBase. Sql* обект, като предава потребителското име, паролата и имейла като параметри. Този метод връща булева стойност, показваща дали потребителят е добавен успешно към базата данни или не.

Ако потребителят е добавен успешно към базата данни, методът *ClearData* се извиква за изчистване на текстовите полета и се извиква методът *BtnClose\_Click* , като се предават бутонът *BtnRegister* и нов обект *Event Args* като параметри. Този ме тод е отговорен за

затварянето на формата за регистрация и връщането на потребителя към формата за влизане.

Ако паролите не съвпадат, методът показва прозорец със съобщение на потребителя, показващ, че въведената парола е неправилна.

Като цяло този кодов блок отговаря за обработката на процеса на регистрация на нов потребител, проверката дали въведената информация е правилна, добавянето на потребителя към базата данни и изчистването на данните от формуляра или показването на съобщение за грешка, ако е необходимо.

Създава се потребителски клас в пространството от имена на *GreenhouseManagementSystem.Models* . Този клас има четири свойства, а именно Id *, UserName* , *Password* и *Email* , всички от които са публични и имат гетери и сетери. Свойството *Id* е цяло число, което съхранява уникалния идентификатор за всеки потребител. Свойството *User Name* е низ, който съдържа потребителското име на потребителя. Свойството *Password* е низ, който съдържа паролата на потребителя. Свойството *Email* е обект, който съдържа имейла на потреби теля. Този клас е проектиран да се използва заедно с други класове в проекта за управление на потребители и техните идентификационни данни.

Създава се клас *GreenHouse* в пространството на имената GreenhouseManagmentSystem.Models. Този клас има четири свойства - *Id* , *Type* , *Size* и *NumberOfPlants* , всички от които са публични и имат гетери и сетери. Свойството Id е цяло число, което съхранява уникалния идентификатор за всяка оранжерия. Свойството *Type* е enum на *TypeOfGreenHouse* , което съдържа типа на оранжерията. Свойството Size е двойно, което държи размера на оранжерията. Свойството *NumberOfPlants* е цяло число, което съдържа броя на растенията, които могат да бъдат настанени в оранжерията. Този клас е проектиран да се използва заедно с други класове в проекта за управление и съхраняване на данни, свързани с оранжерии.

Този код е манипулатор на събитие за действие при щракване на бутон в потребителски интерфейс. Когато потребителят щракне върху бутона, методът извлича списък с оранжерии от базата данни с помощта на метода *GetGreenHouses().* След това извлеченият списък се предава на нов екземпляр на класа GardensPage, който показва списъка с оранжерии в нов прозорец. Методът *Show()* се извиква в екземпляра *GardensPage* , за да покаже страницата на потребителя. Този код предоставя лесен начин за потребителя да види всички налични оранжерии, съхранени в системата.

Този метод се задейства, когато потребителят кликне върху бутона

„ Покажи градини“. Той извлича всички оранжерии от базата данни с помощта на метода " GetGreenHouses" от обекта " DataBase.Sql". След това създава нов екземпляр на класа „ GardensPage“ с извлечените оранжерии като аргумент и показва формуляра „ Gardens Page“.

Параметри:

" подател": Обектът, който е предизвикал събитието ( т . е. бутонът " Покажи градините").

" e": Данните за събитието. Върната стойност:

Нито един.

Странични ефекти:

Формулярът " Gardens Page" се показва на потребителя. Грешки:

Нито един. Пример:

Да предположим, че потребител кликне върху бутона „ Покажи градини“. Този метод ще извлече всички оранжерии от базата данни и ще създаде нов екземпляр на формуляра „ GardensPage“ с извлечените оранжерии като аргумент. След това на потребителя ще се покаже формулярът „ Gardens Page“, показващ всички налични градини.

## 3 Организация и код на заявките към база о т данни

SQL Server Management Studio ( SSMS) е интегрирана среда за управление на всякаква SQL инфраструктура, от SQL Server до Azure SQL база данни. SSMS предоставя инструменти за конфигуриране, наблюдение и администриране на екземпляри на SQL Server и бази данни. Платформата SSMS се използва за внедряване, наблюдение и надграждане на компонентите на ниво данни, използвани от приложения та, и изграждане на заявки и скриптове.

Базата данни е информация, която е създадена за лесен достъп , управление и актуализиране. Компютърните бази данни обикновено съхраняват съвкупности от записи на данни или файлове, които съдържат информация, като транзакции за продажба, данни за клиенти, финансови данни и информация за продукта .

Базите данни се използват за съхраняване, поддържане и достъп до всякакъв вид данни. Те събират информация за хора, места или неща. Тази информация се събира на едно място, за да може да се наблюдава и анализира. Базите данни могат да се разглеждат като организирана колекция от информация.

Използвайте SSMS за заявки, проектиране и управление на вашите бази данни и хранилища за данни, където и да са - на вашия локален компютър или в облака

Базата данни предоставя информация от сензорите за влажност, температура, светлина и пр. Извлича информацията и се показва нагледно и обновява в конзолно приложение написано на C#, Windows

Form App , свързано с базата данни, която е създадена в платформата Microsoft SQL Management Studio 18 и кода от Arduino, заедно с частите и сензорите . Данните се обработват и при наличие на промяна се подновяват в реално време.

CREATE TABLE [dbo].[Users](

[Id] [int] IDENTITY(1,1) NOT NULL,

[UserName] [nvarchar](50) NOT NULL, [Password] [nvarchar](50) NOT NULL, [Email] [nvarchar](50) NOT NULL, PRIMARY KEY CLUSTERED

(

[Id] ASC

)

По- горе в кода се показва примерна заявка от базата данни “ Зелена оранжерия ”( Greenhouse), в която се създава таблицата

„ Потребители“( Users) , в нея се съдържат полетата за

„ Идентификация“( Id), „ Потребителско име“( Username),

„ Парола“( Password) и „ Имейл“( Email).

## 4 Наличие на потребителски интерфейс ( конзолен, графичен, уеб)

Потребителският интерфейс е конзолен, направен с Windows Form App, който ще се достъпва с страница за регистрация или при вече налична, чрез входна страница. При правилно изписване на потребителското име и парола, които преди това потребителят е измислил и се е регистрирал , потребителят може да види състоянието на своята градина.

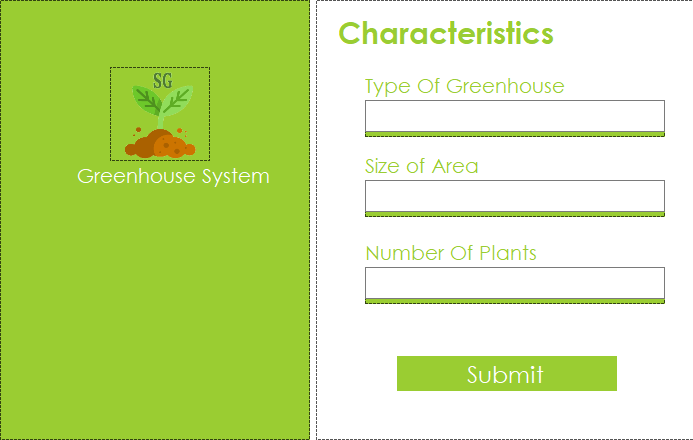
След успешното влизане в профила на потребителя, той има право да създаде нова градина или при наличието вече на такава, да избере вече създадената по-рано от него. Ако не е задал своя градина и

тепърва я създава, се отваря страница, в която той трябва да зададе конкретни характеристики на своята градина, като например големина на площта, брой на засадените растения и пр.

След създаването на градината и поставянето на Ардуиното в нея, потребителят има възможност за следенето на своята реколта и нейното благосъстояние. Собственикът може по всяко време да знае какво се случва с оранжерията му, тъй като данните се актуализира в реално време, и при нарушен баланс може да се предотврати загуба на реколтата.

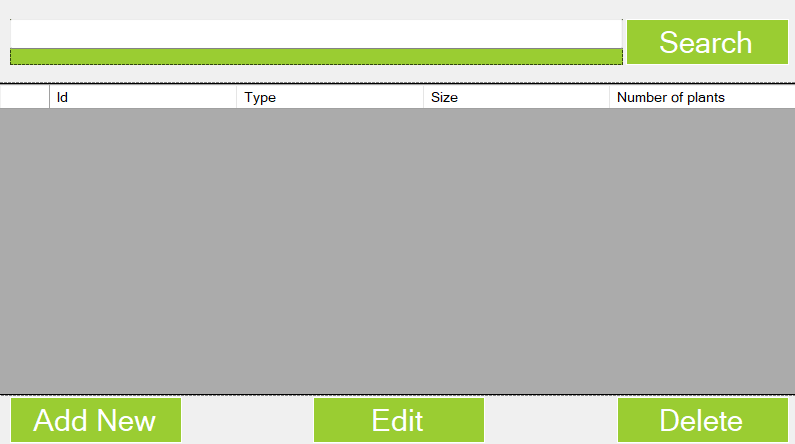
Чрез функциите за създаване, четене, осъвременяване и изтриване( CRUD) са възможни редакции и добавяне и пр. За по-лесно редактиране или добавяне на информацията. Дизайнът е изключително удобен, опростен и лесноразбираем, за да може хора от всякаква възраст да ползват приложението.

Таблиците, които представят данните от сензорите на Ардуиното са лесноразбираеми и максимално опростени, за да е лесно за потребителя да намери нужната информация за него.



*Ф и г ур а 13*

На фигура 13 се изобразява потребителския интерфейс за страница, в която се записват характеристиките на градината. В нея се изобразяват типа на градината, големината на градината в метри и броят на растенията в самата оранжерия. Има и бутон за „ изпращане“, с който данните се прехвърлят към базата.



*Ф и г ур а 14*

На фигура 14 се показва страницата, на която се изобразяват данните от оранжерията, представени чрез Ардуиното. В нея може има поле за търсене, където се филтрират данните. Има и бутони за добавяне, редактиране и изтриване. Чрез тях потребителят е способен да редактира своите данни.

# Ефективност и бързодействие на решението

Кодът е написан на C# и е манипулатор на събития за бутона *BtnSearchGarden*. Той търси градини в списък въз основа на фраза за търсене, въведена от потребителя в текстово поле. Алгоритъмът в този код е:

Декларира се низова променлива, която да съдържа фразата за търсене, въведена от потребителя Присвоява стойността, въведена в текстовото поле *SearchBox* , на променливата на фразата. Проверява дали фразата за търсене не е нула или празна. Ако фразата за търсене не е нулева или празна, се филтрира списъка с градини с помощта на

метода *FindAll* на LINQ . Използва се ламбда израз, за да търсите градини, чийто идентификатор, тип, размер или брой растения съответстват на фразата за търсене. Ламбда изразът търси съвпадения с помощта на метода *Contains* на фразата за търсене. Присвоява се списъка с филтрирани градини към свойството *Data Source* на обекта *bindingSource* . Задава се свойството *Data Source* на *GardenDataGrid* на обекта *bindingSource* . Обновява се контролата *GardenDataGrid*, за да покаже филтрирания списък с градини. Ако фразата за търсене е нулева или празна, извикайте метода *ShowGardens* , за да покажете всички градини.

Времевата сложност на този алгоритъм е O(n), където n е броят на градините в списъка. Това е така, защото трябва да обходи всички градини в списъка, за да ги филтрира въз основа на фразата за търсене. Сложността на пространството също е O(n), тъй като филтрираният списък с градини се съхранява в паметта.

# Тестване

Единичното тестване включва тестване на отделни единици или компоненти на кода изолирано от останалата част от системата. Целта на модулното тестване е да се гарантира, че всяка единица код работи по предназначение и да се идентифицират всички грешки или грешки в кода.

Ползите от модулното тестване включват:

1. Ранно откриване на грешки и бъгове: Единичното тестване помага да се открият грешки и бъгове в кода в началото на процеса на разработка. Това позволява на разработчиците да коригират проблемите, преди да станат по-трудни за разрешаване.
2. Подобрено качество на кода: Единичното тестване помага да се гарантира, че всяка единица код работи по предназначение, подобрявайки цялостното качество на кода.
3. По-бързо развитие: Единичното тестване помага да се намали времето, изразходвано за отстраняване на грешки и тестване чрез откриване на грешки в началото на процеса на разработка.

Единичното тестване е метод за тестване на софтуер, който включва тестване на отделни кодови единици, като функции или методи, изолирано от останалата част от софтуерната система. Това е план на стратегията използвана за осигуряване успешно тестване на единици:

1. . Планиране предварително: Преди да се започне тестване на единица, е важно да се създаде план за процеса на тестване. Това включва идентифициране кои единици се тестват, какви входове се използват и какви изходи да се очакват. Наличието на солиден план ще помогне да се гарантира, че тестването е изчерпателно и че са обхванати всички съответни аспекти на кода.
2. . Използване на инструменти за автоматизирано тестване: Инструментите за автоматизирано тестване могат значително да опростят процеса на тестване на единици, като автоматично изпълняват тестови случаи и сравняват очакваните резултати с действителните резултати. Използването на инструменти като NUnit може да помогне за рационализиране на процеса на тестване и да го направи по - ефективен.
3. . Тестване за крайни случаи: Важно е да се тества за крайни случаи или ситуации, в които кодът може да се държи различно от очакваното. Това включва тестване на екстремни входове, като много големи или много малки стойности, и ситуации, при които могат да възникнат неочаквани входове или изключения.
4. . Тестване често: Единичните тестове трябва да се изпълняват често, в идеалния случай след всяка промяна или актуализация на кода. Това помага да се открият проблемите в началото на процеса на разработка, когато те са по - лесни и по - евтини за отстраняване.
5. . Поддържане на тестовете прости и изолирани: Всеки единичен тест трябва да се фокусира върху тестването на единичен, изолиран аспект на кода. Тестовете трябва да са прости и лесни за разбиране, с ясни входове и очаквани резултати.
6. Използване на „ Провокиране “( mocking) и

„ Подбутване “(stubbing): Те са техники за симулиране на поведението на външни зависимости, като бази данни или Интерфейс за програмиране на приложения ( API), по време на тестване. Тези техники могат да помогнат за изолиране на тествания код и да направят тестването по - ефективно.

1. . Непрекъснато усъвършенстване и актуализиране на тестовете: Единичните тестове трябва непрекъснато да се усъвършенстват и актуализират с развитието на кода. Това включва актуализиране на тестове при добавяне на нови характеристики или функционалност, както и усъвършенстване на тестове за улавяне на крайни случаи или неочаквано поведение.

Като се следват тези стратегии, кодът би бил е щателно тестван и че всички или повечето проблеми или грешки са уловени в началото на процеса на разработка, което спомaга за подобряване на цялостното качество на софтуерната система.

Това са някои методи, за тестване, които се използват:

Тестване на черна кутия: Този метод включва тестване на единица код без никакви познания за основната структура на кода. Достъп има само до входовете и очакваните изходи на модула и оценяват изхода въз основа на очаквания резултат. Този метод е полезен за тестване и

функционалността на кода и проверка дали той отговаря на необходимите спецификации.

Тестване на бяла кутия: Този метод включва тестване на единица код с познаване на основната структура на кода. Достъп до кода и оценяване модула въз основа на неговата вътрешна логика и дизайн. Този метод е полезен за идентифициране на дефекти, свързани със структурата и дизайна на кода.

Тестване на сива кутия: Този метод е комбинация от тестване на черна кутия и бяла кутия. Има ограничени познания за структурата на кода и оценява не на модула въз основа на неговите входове и очаквани изходи, както и на неговата вътрешна логика и дизайн. Този метод е полезен за тестване на единици, където както функционалността, така и структурата на кода са важни.

# Заключение и възможно бъдещо развитие

* + Проектът е насочен към наблюдението и контрола на качеството в параметрите на „ умната градина“ . Всички действия се извършват с помощта на съвременни технологии. ( т . е . Arduino) . Тъй като проектът помага за подобряване на производството на конкретната култура, в бъдеще може да бъде концентрирано за повишаване на качеството и количеството на реколтата. Също така въздействието на ефекта върху околната среда също да бъде по- малко.
  + Резервно захранване о т слънчеви клетки може д а бъде включено в системата за осигуряване електричество в случай на прекъсване на тока, а също и по - ниски разходи за експлоатация прототип на оранжерия.
  + Надеждността на проектираната система може да се използва за изграждане на мрежа от такива системи за наблюдение и контрол за няколко оранжерии.

# Използвани литературни източници и Уеб сайтове

(н.д.). Извлечено от от https:// adrianmejia. com/ most - popular- algorithms - t ime- complexity - every- programmer - should - know- f ree- online-

(н.д.). Извлечено от https:// www. engineersgarage. com/ green - house- monitoring- using-

(н.д.). Извлечено от https:// bg. theastrologypage. com/ model

(н.д.). Извлечено от https://adrianmejia.com/most-popular-algorithms-time-complexity-every-programmer-should-know- free-online-

SQL\_Server\_Management\_Studio, h. b. (н.д.).