Flowing Ideas

Изработено от Катерина Колева

и Надежда Събева

Програмата Flowing Ideas e Windows form приложение за записване на идеи в следните категории: артистична, философска и работна. Следва ORM, MVC, Enity Framework и Data.Sql.Client, като има class library за логиката /Controller; Business/ и датата /Model; Data - folder model, UserIdeaContext/ и клас FlowingIdeasApp /View/, в което са формите и външния вид на приложението.

Базата данни

Базата данни съдържа три таблици - Users, IdeaTypes и Ideas. В таблица Users се съхраняват акаунтите (потребителите), като те имат скрито от потребителя auto incremented int Id (primary key), и стринг Username (VARCHAR(15)). В таблица IdeaTypes имаме auto incremented int Id (primary key) и стринг IdeaTypeName, като те биват 3 - Artistic, Work и Philosophical, като добавяме тези данни още при създаването на базата. Таблица Ideas съдържа auto incremented int Id (primary key) и int UserId (foreign key към таблица Users, препратка към Id) - те остават скрити във формата визуализтор, IdeaType IdeaTypeId (foreign key към таблица IdeaTypes, препратка към Id), като това във визуализатора се превръща в стринг с дадения тип, използвайки метода GetIdeaTypeString() от UserIdeaBusinessLogic класа в Business логиката и string/VARCHAR(400) IdeaText, като това е визуализирано директно.

Потребителски интерфейс

Във входа /първият форм – IdeasFlow.cs/ има username, което първо трябва да се напише в текстовата кутия, за да се продължи с бутоните „Log in” или „Sign up”. То не може да е празно и за двете. При „Sign up” (създаване на акаунт) тексът трябва да е над 3 и под 15 знака. (За което съответно се показват грешки – error providers). Бутонът „Cancel“ затваря формата. Ако няма акаунт с това име и се опитате да влезете /log in/, както и когато опитате да създадете акаунт /sign up/ с вече съществуващо име, дава грешка, която излиза в нов прозорец със съответния текст. При създаването на акаунта се отваря прозорец, че е вече създаден (добавя се и в базата данни, таблица „Users“) и да се продължи с „Log in”.

Натискането на бутона „Log in” Ви отвежда във втори прозорец (форм - IdeasDataViewer.cs), който отговаря за визуализирането на вече съществуващи идеи (в таблица с колони Edit (едитване), Delete (триене), Idea Type (тип на идеята, който е низ /Artistic, Work, Philosophical/), и Idea Text (текст на идеята), като edit и delete са колони бутони, едитващи или триещи селектираната идея), както и бутон за създаване (Add an Idea). По default се изобразяват всички идеи на потребителя, извличайки ги от базата данни. При използването на бутоните с отметка се изобразяват артистични (Artistic ideas), работни (Work ideas) или философски идеи (Philosophical ideas). Бутонът „Cancel” затваря формата. Бутонът за триене премахва идеята от базата данни и гридът се ъпдейтва автоматично.

При натискане на бутона за добавяне на идея „Add an idea” сте отведени в нов прозорец (форм - AddIdea.cs). В него чрез радио бутони се избира типът на идеята (като тя може да бъде артистична, работна или философска), а в текстовото поле пишете текста на идеята си. Низът не трябва да е празен, по-малък от 3 и по-голям от 400 знака, като съответно за това се изобразяват грешки (error providers). При бутонът за запазване „Save” се показва нов прозорец, че идеята Ви е създадена, и при натискане на „Ok“ и отново се показва Формата визуализатор (IdeasDataViewer.cs), съдържаща новата идея. При бутонът „Cancel“ отново се отваря предишната форма.

При натискане на бутонa “Edit” се отведени в нов прозорец (форм - EditIdea.cs), съдържащ текстово поле, за да се промени текстът на идеята. Низът отново не трябва да е празен, по-малък от 3 и по-голям от 400 знака, като за това се изобразяват грешки (error providers). При бутона „Cancel“ сте отведени във формата визуализатор (IdeasDataViewer.cs). А при бутона за запазване изскача нов прозорец, казващ, че идеята Ви е запазена. При натискането на бутона „Ok“ формата визуализатор (IdeasDataViewer.cs) се изобразява отново, показвайки и новата идея.

Данни (Data)

Класът (class library) представлява Model (**M**VC), мапинга и връзката към таблиците от базата данни (БД).

Съдържа:

* Папка Model с класове: Idea, IdeaType, User;
* UserIdeaContext, който наследява DbContext и дава конструктор, имащ името на connection string-a, съдържащ се в App.config. Също прави и мапва DbSet<> от папка Model, към таблици съдържащи се в БД.

Логика (Business)

Класът (class library) представлява Controller (MV**C**), логиката, която стой зад приложението. Дава ни връзка с датата (има dependency към Data). Играе ролята на посредник между потребителския интерфейс и данните.

Съдържа: UserIdeaBusinessLogic.cs.

* Има private field от тип UserIdeaContext.
* Методи:
  + CheckIfAccountExists() – проверява дали акаунт с такова име вече съществува.
  + CreateAccount() – създава акаунт в базата от данни – таблица Users.
  + GetUserId() – взема ID-то на потребителя от БД по username-а му.
  + IdeaType() – взема ID-то на типа идея.
  + GetIdeaTypeString() – превръща ID-то на идеята в името на типа идея.
  + AddIdea() – добавя идеята в БД.
  + GetAll() – взема всички идеи на потребителя от БД.
  + GetIdeasByType() – взема идеите на потребителя от БД спрямо типа.
  + Delete() – трие идеята от БД.
  + Edit() – променя текста на специфично посочената от потребителя идея.

Потребителски интерфейс (View)

Класът Flowing Ideas app представлява View (M**V**C) – потребителският интерфейс на приложението. Дава ни връзка към бизнес логиката и датата (има dependencies към Data и Business).

Съдържа:

* Папка Properties.
* Папка Resources – използваните изображения.
* App.config – в него е връзката към БД. (connection string-a)
* AddIdea.cs – формата за добавяне на идея.
* EditIdea.cs – формата за променяне на идея.
* IdeasDataViewer.cs – формата визуализатор на съществуващите идеи в БД. Връзка към AddIdea.cs и EditIdea.cs.
* IdeasFlow.cs – формата, която се показва, когато пуснем приложението – в нея се създават акаунти и се влиза в IdeasDataViewer.cs.
* StartUp.cs – от тук се стартира програмата.

IdeasFlow.cs:

* Има field инстанция на UserIdeaBusinessLogic.
* Конструктор, които инциализира формата.
* Методи:
  + buttonLogIn\_Click() – проверява дали текстовото поле е празно (error provider), дали такъв акаунт съществува (CheckIfAccountExists() – метод от бизнес логиката (БЛ)) и ако съществува използва метода GetUserId() от БЛ и инциализира форма от тип IdeasDataViewer, скривайки сегашната форма. Ако акаунт с това име не съществува в БД дава грешка.
  + buttonCancel\_Click() – затваря приложението.
  + IdeasFlow\_Load() – зарежда формата.
  + buttonSignUp\_Click\_1() – проверява дали текстовото поле е празно, дали низът е по-малък от 3 и по-голям от 15 (error providers), дали акаунт с това име вече не същестува (CheckIfAccountExists() от БЛ) – дава съобщение, ако вече същестува. Ако всичко е изпълнено без да има грешки, създава нов акаунт в БД (метод CreateAccount() от БЛ) и изписва съобщение, че е създаден и да се продължи с „Log in”.

IdeasDataViewer.cs:

* Има field инстанция на UserIdeaBusinessLogic.
* Има field int userId.
* Конструктор, които инциализира формата и за който е необходим userId-то (взето от формата IdeasFlow.cs при GetUserId()) + метода UpdateGrid()).
* Методи:
  + UpdateGrid() – показва кутия с всички идеи на потребителя, като използва метода GetAll() от БЛ. Прави колоните „Id” и „UserId” скрити. Форматира клетките от „IdeaTypeId” от int в string, ако вече не са, използвайки метода GetIdeaTypeString() от БЛ.
  + FilterAndDisplayIdeas() – прави лист от филтрирани идеи, като отговаря при чекирането на кутиите за различните видове идеи, използвайки метода GetIdeasByType() от БЛ. Прави колоните „Id” и „UserId” скрити.
  + FlowingIdeaForm\_Load() - зарежда формата.
  + dataGridView1\_CellContentClick() – проверява дали кликната клетка е бутонът „Delete“ – ако е, взема Id-то на идеята и използва метода Delete() от БЛ, или „Edit“ – отново взема Id-то на идеята и инициализира форма от тип „EditIdea“ и скрива сегашната. След извършване на една от двете използваме метода UpdateGrid().
  + PhlosophicalCheckBox\_CheckedChanged() – кодът зад чек бокса на „Philosophical ideas”, използва метода FilterAndDisplayIdeas().
  + artisticIdeaCheckBox\_CheckedChanged\_1() - кодът зад чек бокса на „Artistic ideas”, използва метода FilterAndDisplayIdeas().
  + workIdeaCheckBox\_CheckedChanged\_1() - кодът зад чек бокса на „Work related ideas”, използва метода FilterAndDisplayIdeas().
  + AddAnIdeaButton\_Click() – инициализира форма от тип „AddIdea“ и крие сегашната.
  + buttonCancel\_Click() - инициализира форма от тип „IdeasFlow“ и крие сегашната.

AddIdea.cs:

* Има field инстанция на UserIdeaBusinessLogic.
* Има field int ideaTypeId.
* Има field int userId.
* Конструктор, които инциализира формата и за който е необходим userId-то (взето от формата IdeasFlow.cs при GetUserId()).
* Методи:
  + AddIdea\_Load() - зарежда формата.
  + ClearTextBoxes() – изчиства текст бокса.
  + TypeArtisticIdea\_CheckedChanged() – кодът зад радио бутона „Artistic”, приравнява IdeaType на Artistic.
  + TypePhilosophicalIdea\_CheckedChanged() - кодът зад радио бутона „Philosophical”, приравнява IdeaType на Philosophical.
  + TypeWorkIdea\_CheckedChanged() - кодът зад радио бутона „Work related”, приравнява IdeaType на Work.
  + buttonCancel\_Click() - инициализира форма от тип „IdeasDataViewer“ и крие сегашната.
  + buttonSave\_Click() – проверява дали текстовото поле е празно, под 3 и над 400 символа (error providers) и ако критериите са изпълнени използва метода AddIdea() от БЛ, изписва в нов прозорец, че идеята е създадена, скрива сегашната форма и инициализира форма от тип „IdeasDataViewer“.

EditIdea.cs:

* Има field инстанция на UserIdeaBusinessLogic.
* Има field int ideaIdToEdit.
* Има field int userId.
* Конструктор, които инциализира формата и за който е необходим userId-то (взето от формата IdeasFlow.cs при GetUserId()), както и ideaIdToEdit (взето от формата IdeasDataViewer.cs).
* Методи:
  + ClearTextBoxes() – изчиства текст бокса.
  + buttonCancel\_Click() - инициализира форма от тип „IdeasDataViewer“ и крие сегашната.
  + buttonSave\_Click() - проверява дали текстовото поле е празно, под 3 и над 400 символа (error providers) и ако критериите са изпълнени използва метода Edit() от БЛ, изписва в нов прозорец, че идеята е променена, скрива сегашната форма и инициализира форма от тип „IdeasDataViewer“.
  + EditIdea\_Load() – зарежда формата.