9- изпитен въпрос ПП-Разработка на софтуер

1. Описва и обяснява видове потребителски интерфейс.

Под потребителски интерфейс обикновено разбираме видимата част на приложението, или средата в която работим и чрез която осъществяваме диалога с програмата. С помощта на интерфейса потребителят осъществява въвеждането на данни и получаването на изходните резултати. От видът на интерфейса и неговата структура, от това дали е лесен и приятен за работа, зависи до голяма степен комфорта на потребителя, който работи с нашето приложение.

Според конвенционалната класификация, съществуващите интерфейси могат да бъдат разделени на практика в следните типове:

1. Интерфейсът на команда. Една от основните и най-старият е интерфейс на командния ред. Command (команден ред) интерфейс е получил най-голям развитието в разцвета на големи системи с много потребители с буквено-цифрови дисплеи. Тя се характеризира с това, че потребителят взаимодейства с компютъра чрез командния ред, където вход команда на определен формат, и след това се предават на изпълнение.

2. графичен потребителски интерфейс е важен компонент на повечето съвременни софтуерни продукти, насочени към опит за крайния потребител. Основните предимства на графичния потребителски интерфейс е ясен и интуитивен за потребителите, както и общия интерфейс на програми, написани специално за работа в графична среда.

3. SILK-интерфейс (Реч, изображение, език на знанието - реч, изображение, език, знания). В момента съществува SILK интерфейс само като "глас" (с изключение на биометрични интерфейси, които се използват не за контрол на компютъра, но само за идентифициране на потребителя). Това е един много обещаващ посока поради причината, че въвеждане на информация на глас - най-бързият и най-удобният начин. Но прилагането му все още не е станала доминираща защото на качеството на разпознаване на реч все още е далеч от идеалното.

2. Различава графичен и конзолен интерфейс

При графичния интерфейс подаваме командите с натискане на командни бутони и настройваме параметрите в диалогови прозорци. При конзолния интерфейс командите са последователности от знаци на клавиатурната азбука, завършващи със знака за нов ред (клавишът Enter). С интервали, командата се разделя на отделни думи.

3. Посочва компоненти на графичен интерфейс и обяснява предназначението им.

Основни елементи на потребителския интерфейс

В основата си потребителският интерфейс се състои от обекти, които поставяме на екрана и подреждаме по начин, по който екранът изглежда привлекателен и отговаря на задачата ни.

Основните елементи на потребителския интерфейс са:

Форми (прозорци)

Бутони – текстови, контролни и т.н.

Кутии – текстови, контролни и т.н.

Изображения – икони, графики, анимации и т.н.

Скролери

Други специализирани.

Основен компонент на графичното приложение е стандартният прозорец на Windows. В прозореца се разполагат контроли на GUI, които понякога се наричат интерфейс. Контролите са обекти, които дават възможност за визуализиране на информацията на екрана и за взаимодеиствието с приложението чрез мишка, клавиатураи др. средства за въвеждане на данни.

Осножни контроли на GUI :

етикет label – статичен текст

текстова кутия TextBox

бутон button

логически команди ChekBox

ComboBox

ListBox

Panel

Графичните приложения в ОС Windows се базират на концепцията Windows формите.

Това е концепция, която е взаимствана от средите за бърза разработка. Формата е графичен елемент, който може да е диалогов прозорец, стандартен или многодокументен прозорец. За да се създаде Windows приложение във Visual studio се избира от инсталираните шаблони Windows Application. Като резултат се създава приложение, което се състой от една форма. Формата е контейнер на контроли и компоненти. Контролите и компонентите се поставя във формата чрез теглене и пускане. Достъпните свойства на формата или на избрания контрол могат да се настроят по време на писане на програмата. При създаването на Windows приложението в браузърът на решението на Solution Explorer се създават няколко файла. Първият е Program.cs. На второ място се създава Form.cs, кадето се помества кодът, чрез който се управлява поведението на формата. Този код е най – важен от гледна точка на програмирането. Ключовата дума Partial означава, че дефиницията на класа продължава в друг файл. В конструктора на класа Form1 се извиква методът Initialize Component(), който задава името на формата, размера на контролите и т.н. Тези характеристики, които се задават при проектирането на формата вредактора на форми се генерират автоматично във файл с име Form1Desinger.cs. При добавяне на контрол във формата автоматично се генерира необходимият код, а при изтриване на формата, той се изтрива.

Обработка на събития:

GUI се управлява от събитията, това означава, че когато потребителят взаимодейства с контрол се получава събитие.

4. Описва стъпките за създаване на потребителски интерфейс с изучавана технология и ги демонстрира.

Използване на ASP.NET MVC за създаване на потребителски интерфейс

* ASP.NET MVC е рамка за изграждане на уеб приложения.
  + Базирана на MVC шаблона
  + Лесна и бърза за работа
  + Чудесен визуален интерфейс за нашето приложение

Благодарение на ASP.NET MVC ние ще дадем добър и удобен външен вид на нашето CRUD приложение

Структурата на един проект включва:

ASP.NET Web Application

* + Tова ще е проекта ни, който ще действа като презентационен слой

Data - това ще е проекта ни, който ще бъде слоят за данни. При създаването му от Visual C# -> Windows трябва да изберем Class library

Business - това ще е проекта ни, който ще бъде слоят за услуги. При създаването му от Visual C# -> Windows трябва да изберем Class library

Добавяне на EntityFramework

Добавяне на референции

Използване на приложението

* Компилирайте и стартирайте приложението.
* Добавете /Product в края на URL адреса и заредете страницата
* Пред вас вече е наличен уеб интерфейса за нашето приложение, в който може да извършите всички CRUD операции

5. Свързва потребителския интерфейс и бизнес-логиката на приложението.

Свързване на ViewModels към Views

Съществуват много подходи за свързването на един ViewModel с View, като например директна връзка или подход базиран на контейнер. Всички обаче споделят една и съща цел, която е на свойството DataContext на View да се присвои конкретен ViewModel. Тази връзка може да се направи както в code-behind файла, така и във самото View.

Code-Behind

Едно View може да има в своя code-behind файл код, който служи за задаване на ViewModel като стойност на DataContext свойството. Това може да е просто като инстанцирането на нов ViewModel и присвояването му на DataContext свойството на View, или с инжектиране на ViewModel във View, използвайки inversion-of-control контейнер. При всички случаи осъществяването на тази връзка в code-behind файла не е препоръчителна, тъй като може да доведе до проблеми на дизайнерите в Visual Studio и Microsoft Expression Blend®.

View

Ако ViewModel няма параметри в конструктора си, то той може да бъде инстанциран във View от неговия DataContext. Често използван подход за постигането на това е използването на ViewModel locator. Това е ресурс, който показва ViewModel-ите на приложението като свойства, към които отделните изгледи могат да се свържат. Този подход предполага, че приложението има един клас, отговарящ за връзката между ViewModel-и и View-т

6.Разработва програмен код (класове, методи и др.) спрямо UML и E/R диаграми. Разчита UML и E/R диаграми.

Реализирайте и анализирайте Use case диаграма за управление на Система за електронно обучение.

На пръв поглед се различават следните основни елементи (обекти):

- Курсове и Теми, които представляват елемент на курса;

- Лектор, който води курса;

- Администратор на системата, който може да редактира курса, да създава нов курс и да го управлява;

- Календарен график на курса;

- Студенти, които ръководят календарния график на курса, в който те участват.

1. Определете актьорите в системата за електронно обучение.

………………………………………………………………………………………………………………………..

2. Идентифициране на Use case.

………………………………………………………………………………………………………………..