Отборен Проект по ООП

На

Nick – Name - #checkins - tasks

saykor – Димитър Тодоров – 35 checkins – координнира разработването на проекта и дооправя кода. WPF и MongoDB частта.

smvalkov – Станислав Вълков – 13 checkins – описание на класовете

sbaychev – Стефан Байчев – 5 checkins – тестване и създаване на custom exception

mortal – Николай Николов – 7 checkins – Намиране и добавяне в базата на временни имена на градове и хора за тестове. Четене от текстов файл.

gergana\_penova - Гергана Пенова – 4 checkins – документация.

Целта на настоящият проект е да създадем структурата за десктоп програма за университетска база данни.

http://jockerteam.codeplex.com/

Частта **TelerikUniversity.Data.** съдържа в себе си:

* Статичен клас AppCache – трябва да държи кеша на програмата който в момента е само StudentList. Също така има едно много важно property което е статично но инициализирва не статичен клас. По този начин хем достъп от цялата програма до методите и пропъртитата на този клас и в същото време в WPF мога да имам байдинг към тях. – Singleton pattern design.
* клас ApplicationViewModel – наследява клас PropertyChange който е наследник на INotifyPropertyChanged. Ползва се MVVM в WPF програмата
* клас city – има поле за име на град и конструктор
* клас DataBase – в него се създава лист от променливи от тип city и чрез конструктор се четата имена на градове от съществуващ файл имена на градове
* клас Configuration – държи връзката с базата данни и в последствие може да има всякакви пропъртита или методи нужни за дадената сесия.

Частта **TelerikUniversity.Data.Enumerators** В него са листовете с предварително дефинирани тип данни с информация. В случая:

* списък с държави
* списък със специалностите на студенти
* списък със катедри

В Частта **TelerikUniversity.Data.Extensions**са класове:

* Helpers – за момента има LoadFile() и ToInt() метод идеята на който е да ми конвертва стринг кум инт по-лесно.
* ApplicationViewModelExtensions – съдържа в себе си статичени екстенжъни за записване и зареждане на данни

Частта **TelerikUniversity.Data.Infrastructure** съдържа в себе си структури и абстрактни класове, с които програмата ще работи

* Абстрактен клас (ContactData) – там се взимат и се съхраняват първото и последното име, държава, град, e-mail, адрес и телефонен номер
* struct GeoLocation – взима и съхранява параметрите latitude и longitude
* Абстрактен клас UniversityPerson – той наследява всички параметри на Абстрактен клас (ContactData) и има един допълнителен параметър изработени часове.
* Клас Worker – взима информация университетският служителя от коя катедра и я съхранява
* Интерфейс Iworkable – има метод който дава възможност на всички служители в университета да им се смятат часовете за седмицата. На учителя се смята каква му е заплатата, на студента колко отсъствия има, на асистента колко бонус точки.
* Интерфейс Iprintable

Частта **TelerikUniversity.Data.Infrastructure** съдържа в себе си класове, с който програмата ще работи

* потребителския профил ( UserProfil ) - наследява всички параметри на Абстрактен клас ContactData и има полета за password и userName;
* учителски профил Teacher - наследява всички параметри на Абстрактен клас UniversityPerson и интерфейс Iworkable и има метод за извеждане заплатата му за седмица според изработените часове + бонус \*2 ако за надвишените часове.
* студентски профил(Student) - наследява всички параметри на Абстрактен клас UniversityPerson и интерфейс Iworkable, има полета за факултетен номер и специалност (взима се от предварително дефиниран тип данни enum). Има метод за override първото и последното име и метод за извеждане на REQUESTED\_HOURS\_PER\_WEEK -
* асистентски профил (Assistent).- наследява всички параметри на Абстрактен клас UniversityPerson и интерфейс Iworkable, има допълнителен метод за умножение на получената от интерфейса информация за часове по зададен коефициент
* клас Университет (University) – има полета за Име на университета и геолокация (използва struct)

**TelerikUniversity.Data.Exceptions**

* DataBaseConnectionException това е нашия custom exception с който прихващаме грешки при достъпване до базата. Ако примерно интернета прекъсне.

