Архитектурен проект

Тема:

Електронен училищен дневник

Група: 2

Иван Кръстев Кръстев 471219054

Йордан Илиянов Янков 471219026

Марина Валентинова Мендова 471219060

Марио Владимиров Костадинов 471219048

Силвия Пламенова Василева 471219002

Ясен Робертов Малинов 471219068

Дата: 30.10.2021 г.

Съдържание

Въведение ... 1

Цел на документа ... 1

За проекта ... 1

Предназначение ... 2

Обхват ... 2

Участници ... 2

Използвани термини и символи ... 2

Архитектурен обзор ... 3

Use-case изглед ... 4

Логически изглед ... 5

Изглед на данните 6
Изглед на имплементацията 7
Нефункционални изисквания 8

Въведение

Цел на документа

Целта на настоящия документ е да опише софтуерните изисквания и архитектурата, свързана с изпълнението на обществена поръчка с предмет: изготвяне на електронен училищен дневник. В настоящето техническо задание са описани и изискванията към проектната организация, документация и отчетност. Абстрактно са описани начините за реализация на поставените цели.

За проекта

Проектът е насочен към голяма група участници в академичната среда - учители, ученици и родителите. Софтуерът ще е с леснодостъпен интерфейс, за да бъде подходящ и разбираем за хора, които нямат висока компютърна грамотност (като по-възрастните учители, или по-малките ученици). Целта на проекта е да се създаде удобна платформа – онлайн училищен дневник, който да предоставя цялата необдходима информация, свързана с училищната среда. По този начин ще се достигне високо ниво на информираност между ученици, учители и родители. Друг водещ мотив е приложението да бъде ефективно и бързо за достъп за всички потребители, за това информацията ще е подредена по лесен и бърз за разбиране и виждане начин. Благодарение на електронния училищен днесвник учителите ще осъществяват по-лесно ежедневните си задачи, а родителите ще получават актуална информация във възможно най-бързи срокове.

Избраният архитектурен шаблон е MVC. Реализацията на неговите концепции е предвидена с ASP.NET Core на програмния език C#.

Предназначение

Обхват

Този документ служи за комуникация между заинтересованите страни, представя абстрактно реализирането на софтуера, дава възможност за справка по време на съпровождането на системата.

Участници

Заинтересованите страни са:

- Софтуерните разработчици
  - Use-case изглед
  - Логически изглед
  - Изглед на имплементацията
  - Нефункционални изисквания
- Проектантите на базата данни
  - Изглед на данните
- Възложителят
  - Use-case изглед
- Ползвателите на уеб приложението
  - Use-case изглед

Използвани термини и символи

MVC – архитектурен шаблон, предназначен за уеб приложения, чрез който се реализира разделение на данните, бизнес логиката и визуализирането на потребителския интерфейс

БД – релационна база данни, съхраняваща информацията, въведена в приложението

ASP.NET Core – фреймуърк на Microsoft, подходящ за различни платформи r предоставящ висока производителност

Архитектурен обзор

В документа са включени следните изгледи:

Use-case изглед

- **Цел:** Описва основната функционалност, предоставена от софтуера, и различните възможни сценарии, както и участниците в тях. Изгледът се изобразява чрез Use case диаграми.
- **Предназначен за:** Всички, които са заинтересовани от системата

Логически изглед

- **Цел:** Описва класовете на системата, техните полета и методи. Различните зависимости между класовете се демонстрират чрез клас диаграми.
- **Предназначен за:** Разработчиците

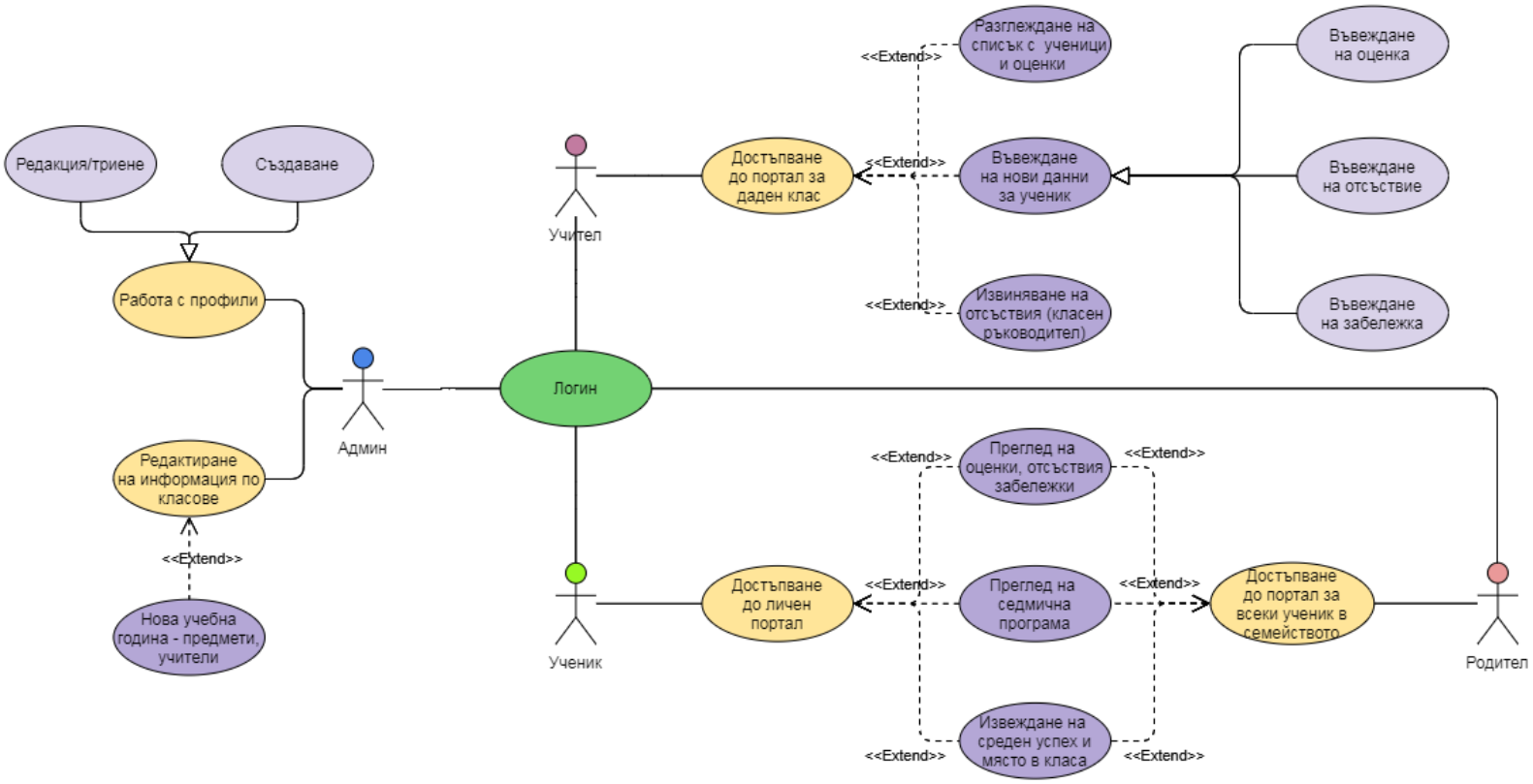
Изглед на данните

- **Цел:** Описва проектирането на БД. Представени са графично таблиците, атрибутите, ограниченията и зависимостите.
- **Предназначен за:** Проектантите на БД

Изглед на имплементацията

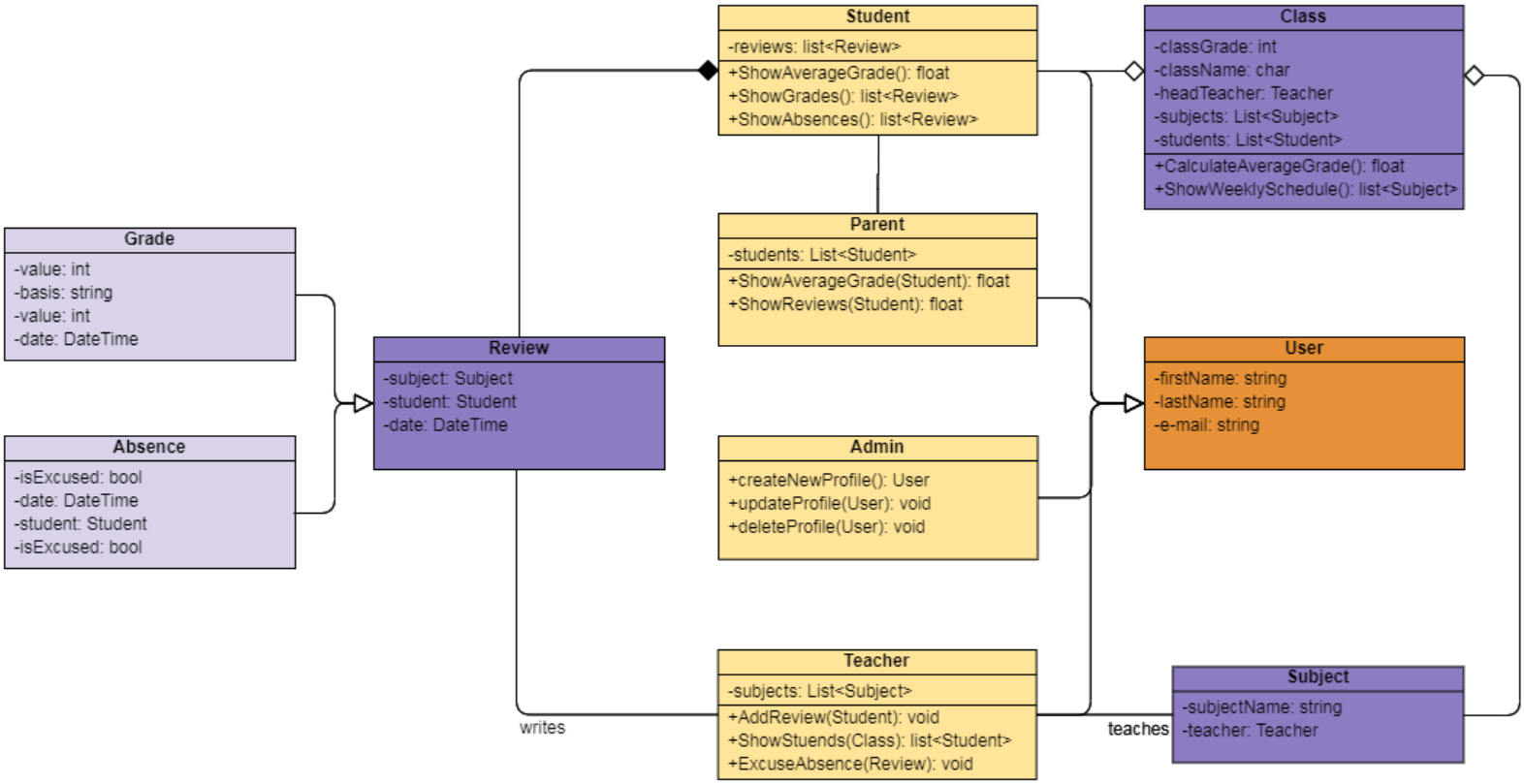
- **Цел:** Описва основните концепции, взети предвид по време на реализацията на софтуера. Дава информация относно използваните слоеве и предназначението им.
- **Предназначен за:** Разработчиците

Use-case изглед



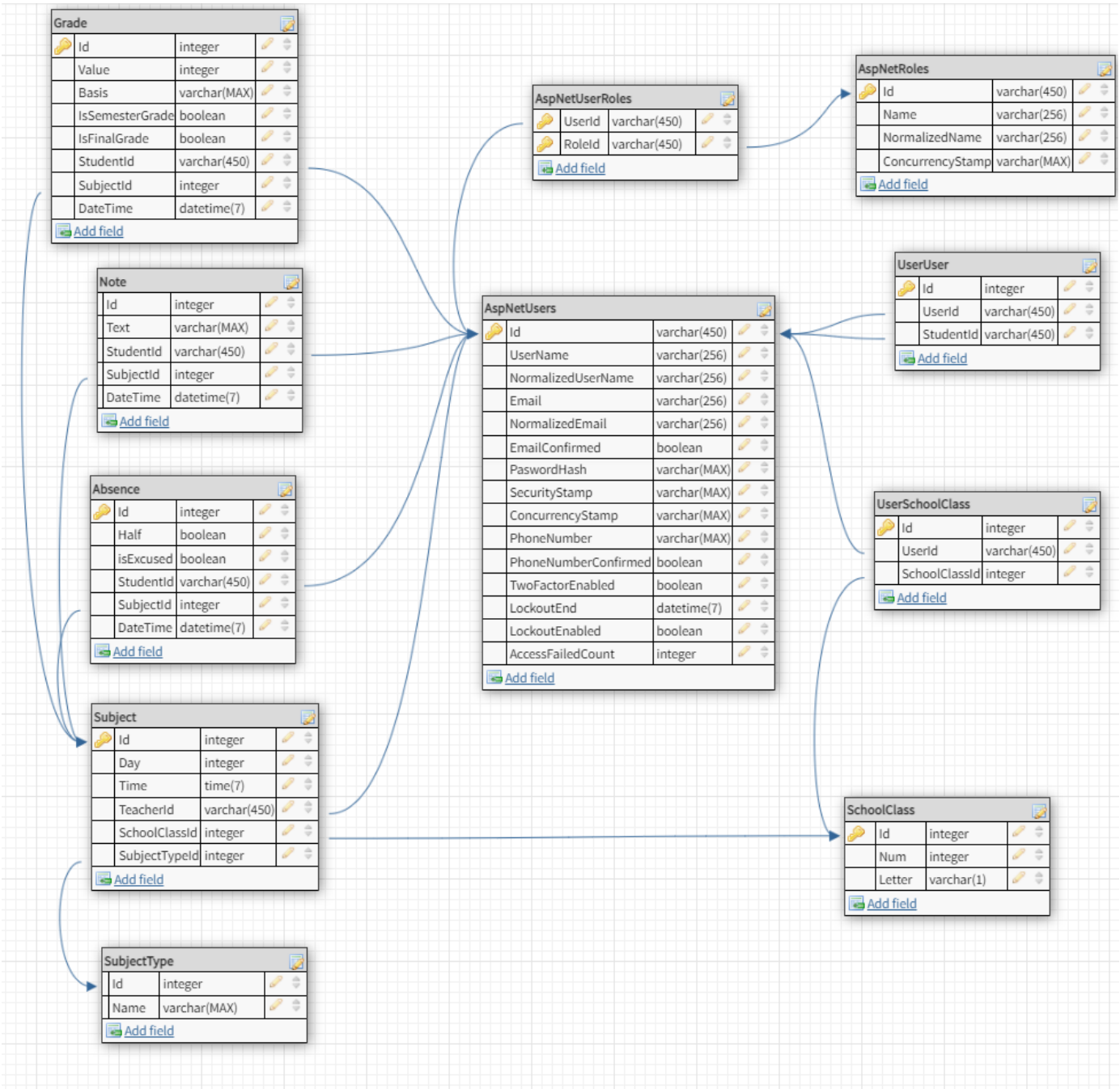
Актьорите, взаимодействащи със софтуера, са четирите различни вида потребители- учители, ученици, родители и админи. Всеки ученик има достъп до индивидуален изглед, представящ неговите оценки и отсъствия, както и данни за класа му (седмична програма, среден успех и т.н.). Данните за класа в контекста на определен предмет са достъпни за съответния учител. Всеки учител може да преглежда и въвежда данни (оценки, отсъствия, забележки) за своите ученици по съответния предмет. В допълнение всеки учител може да извинява отсъствия на ученици от класа, на който е класен ръководител. Родителите имат достъп до същата информация, която е видима и за децата им, но събрана на едно място. Администраторите имат правото да управляват профилите, за да се избегне неконтролираното добавяне на профили от обикновени потребители. Също могат да променят информация за цели класове.

Логически изглед



Класовете Student, Parent, Admin и Teacher наследяват User, който държи цялата информация, необходима за един профил. Всеки един от наследниците има допълнителни функции. Teacher може да нанася отзив (оценка или отсъствие) на ученик, на когото преподава, като отзивът се запазва в списък, съдържан в съответната инстанция на класа Student. При изтриване на профила на даден ученик, всички отзиви за него също се изтриват, което означава, че двата класа са в отношение „композиция”. В един учебен клас (Class) има много ученици (Student), като инстанции на двата класа могат да съществуват поотделно- отношението е „агрегация”. За реализирането на основни функции като показване на седмичен разпис класът Class съдържа в себе си списък от предметите, които се преподават на учениците в него. Класът и предметите могат да съществуват и поотделно, следователно те също са в отношение „агрегация”.

Изглед на данните



Таблицата SubjectType е за конкретен предмет на клас предаван от конкретен учител.

В таблицата Absence, стоят данните за отсъствията на учениците.

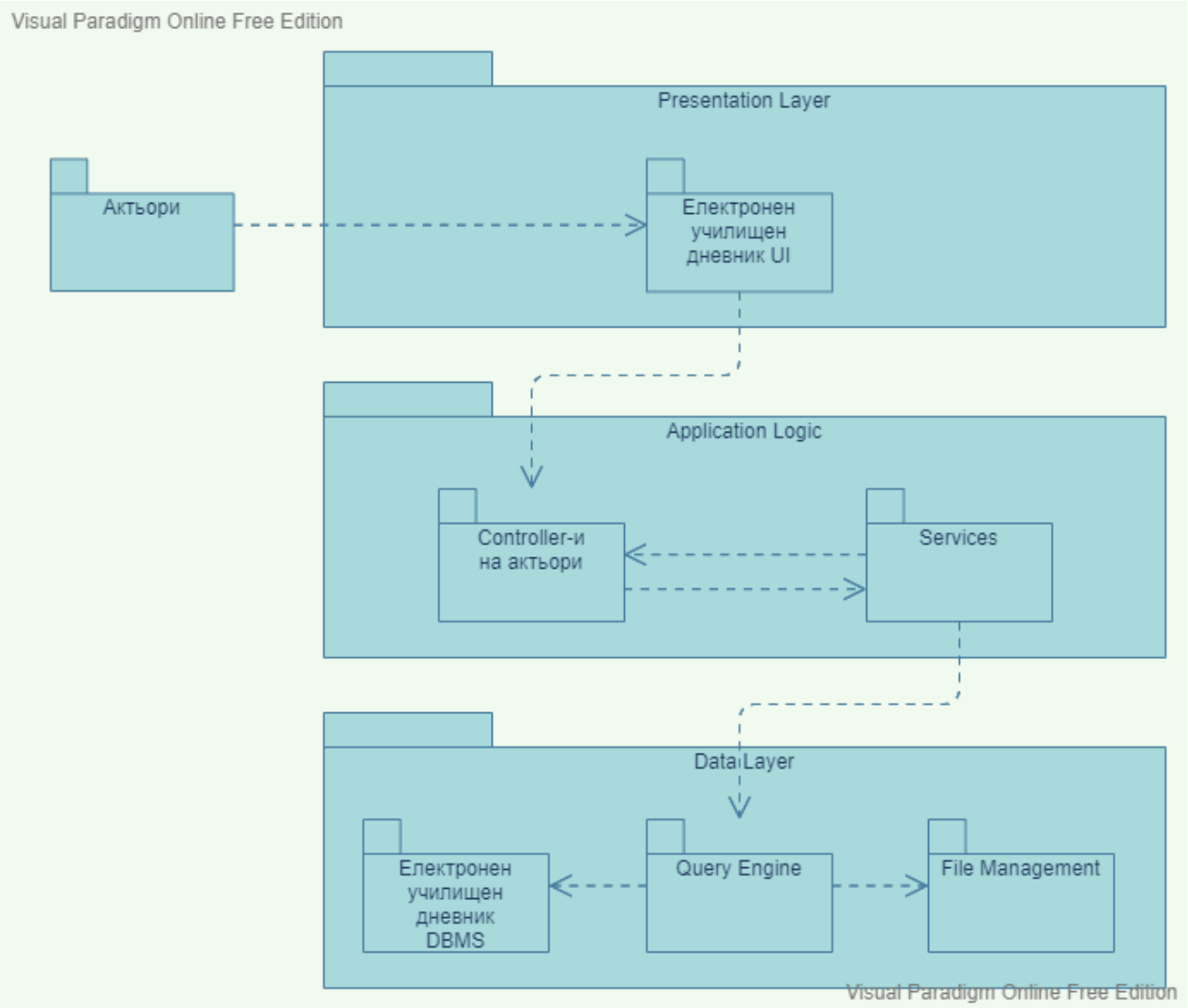
Свързващата таблица UserUser е за създаване на профили на родителите.

Таблицата UserSchoolClass свързва двете таблици SchoolClass и AspNetUsers и не е с Composite Key и единствено тази връзка не е направена с миграции.

Използвани са свързочни таблици, означени с „таблица_таблица”, показващи връзката „много към много” между двете свързани таблици.

Изглед на имплементацията

Спрямо използваната MVC архитектура, изгледа на имплементацията се представя чрез **Presentation Layer**, **Application logic** и **Data Layer**. **Presentation Layer**-а е това, което потребителите виждат. Той играе важна роля, защото улеснява използването на софтуера като го представя в лесна за разбиране светлина. Чрез него потребителите взаимодействат със системата. В **Application Logic** се случва обработването на информацията (нововъведени оценки и отсъствия и други потребителски заявки). **Data Layer**-а предоставя връзка към базата данни и извършва взимането, редактирането или изтриването на информация.



Нефункционални изисквания

За реализирането на проекта е използвана MVC архитектурата. Тя е структурирана в три отделни компонента: **Model, View** и **Controller**, които взаимодействат помежду си. Компонентът **Model** управлява системните данни и свързаните с тях операции. Компонентът **View** определя и управлява как данните се представят на потребителя. Компонентът **Controller** управлява взаимодействието с потребителя (пр. натискане на клавиши и т.н.) и предава тези взаимодействия на компонента **View** или **Model**. Самата архитектура е лесна за поддръжка и осигурява възможност за бъдещо разширяване и развитие. За да подобрим сигурността сме предвидили идентификация на потребителите чрез име и парола и вече в зависимост от ролята си (Учител, Ученик, Родител) ще имат различни права в системата. Също така сме предвидили роля Admin, която да се грижи за регистрирането на различните ученици, учители и родители в системата, като по този начин целим да избегнем създаването на фалшиви профили и да спрем достъпа на външни лица до дневника.

В обобщение изискванията, изпълнени от избраната архитектура, са:

- **Разширяемост –** Архитектурата води до разделянето на софтуера на компоненти (Model, View и Controller), които могат да се модифицират и променят поотделно. Това улеснява добавянето на нови функционалности към проекта.

- **Използваемост –** Интерфейсът трябва да е консистентен (еднакви размери, цветове, символи и стилове за сходни действия) и не изисква много нови знания за научаване на функционалностите му. Изгледът трябва да се определя от ролята на потребителя така, че да не се показва информация, която не е предназначена за него. Трябва да са заложени упътвания и възможност за връщане след евентуално допускане на грешка. Действия от рода на изтриване на профил (от страна на админа) или нанасяне на грешна оценка трябва да са обратими в известен период от време.

- **Сигурност –** достъпът до информацията в системата става след автентикация, реализирана от логин форма. Само определени потребители (админи) имат правото да създават нови профили и да променят информацията за тях. Така се избягва създаването на фалшиви профили и достъпът на външни лица до дневника.

- **Възможност за тестване –** Model, View и Controller компонентите могат да се тестват поотделно. Това улеснява откриването на грешки и увеличава възможността за бърза сервизна поддръжка.