Архитектурен проект

Заглавие:
Училищен онлайн дневник

Изготвили:

Ангел Илиев (мениджър на проекта),
Зорница Стоянова,
Денис Спасов,
Любомир Сотиров,
Айдоан Балъкчиев,
Боян Ковачев

Линк към хранилището в Github:
https://github.com/Angel04172002/ST-Project18

Дата: 19.10.2023

Въведение

Предизвикателствата, пред които е поставен образователният процес, налагат използването на технологии, които да подпомагат образователната среда, правейки я по-достъпна, сигурна и ефективна. Нашият проект за училищен онлайн дневник предоставя лесната комуникация между учители, родители и ученици, осигуряване на по-лесен начин за учене и оценяване, както и спомага образователният процес в училище. Проектът предоставя различни видове потребители - ученик, родител, учител, директор (администратор). Всеки един от потребителите има различни правомощия в системата, като ролята ще бъде предоставена от администратора. Учениците имат достъп до учебните материали и индивидуалните им оценки. Родителите имат достъп до оценките и статистиката на учениците. Учителят има правомощие да оценява, изтрива, качва оценки и материали. Администратова ще управлява системата и ще дава правомощия на всеки потребител.

Предназначение

Обхват

Обхващат се основни части от жизнения цикъл на проекта като:

*инициране - изискванията определят целите на проекта

*планиране - изискванията предоставят информация за функциите, които ще бъдат включени в системата

*изпълнение - изискванията се използват за ръководство на развитието на системата

*контрол - изискванията се използват за оценка на напредъка на проекта

*приключване - изискванията се използват за оценка на успеха на проекта

Актьори

Ученици - имат достъп до оценки и отсъствия, могат да комуникират с учители и родители, могат да следят важни събития

Учители - могат да въвеждат и редактират оценки, могат да комуникират с ученици и родители, могат да следят важни събития

Родители - могат да преглеждат оценки и отсъствията на децата си, могат да комуникират с учители, могат да следят важни събития

Администратор - може да създава профили, може да управлява правата за достъп, може да отговаря за сигурност и поверителност

- Учители:
  - Функции за въвеждане и редактиране на оценки
  - Функции за комуникация с ученици и родители
  - Функции за проследяване на важни дати и събития
- Родители:
  - Функции за преглед на оценки и отсъствия на деца
  - Функции за комуникация с учители
- Администратори:
  - Функции за създаване на профили
  - Функции за управление на правата за достъп
  - Функции за сигурност и поверителност

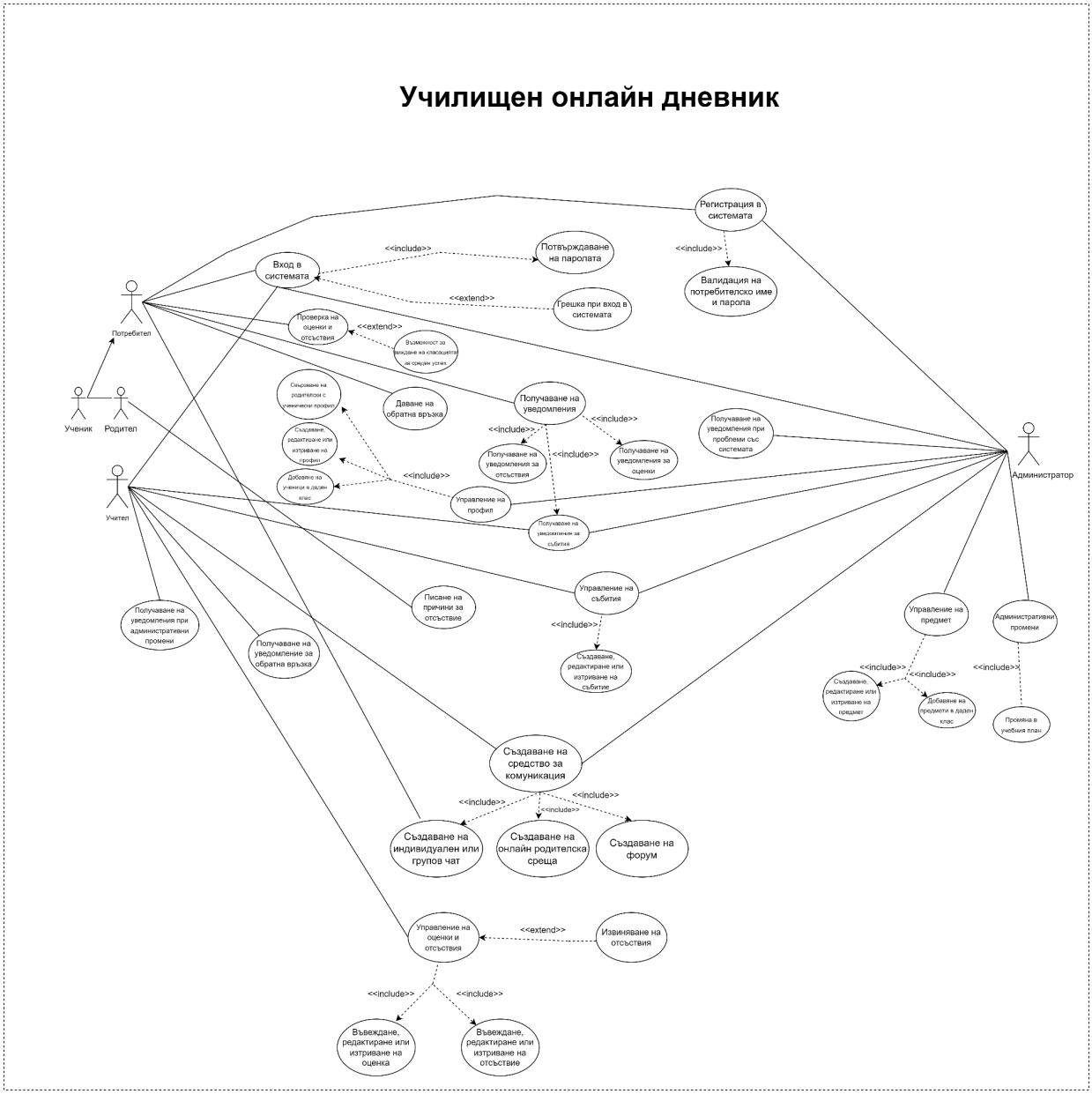
Използвани термини и символи

API - Application Programming Interface

XSS - Cross Site Scripting

Архитектурен обзор

1. Use-case изглед:



Потребителите въвеждат потребителско име и парола, за да получат достъп до системата. При регистрация на нов потребител е необходимо да се валидират потребителското име и паролата, за да може да получи достъп до системата Не трябва да съществува друг потребител със същото потребителско име. Паролата трябва да съдържа поне една малка буква, една главна буква, една цифра и един специален символ.

Учениците и техните родители имат възможност да виждат на кое място са в класацията по среден успех в училището.

Администраторът може да:

*създава, редактира и изтрива профили и предмети
*създава всяко едно от средствата за комуникация
*добавя ученици или предмети в даден клас
*свързва родителските с ученическите профили
*добавя, редактира или изтрива събития в календара

Учителят може да :

*въвежда, редактира или изтрива оценки и отсъствия
*извинява отсъствия
*създава всяко едно от средствата за комуникация
*Вижда обратна връзка за начина си на преподаване, получена от родители и/или ученици

*Добавя, редактира или изтрива събития в календара, описан в 4та точка

Родителят може да:

*вижда оценките и отсъствията на своето дете

*пише причини за отсъствията на своето дете

*създава индивидуален или групов чат

*дава обратна връзка за начина на преподаване на учителите на своето дете

*получава уведомления за отсъствията на детето си и за всички събития в календара

Ученикът може да:

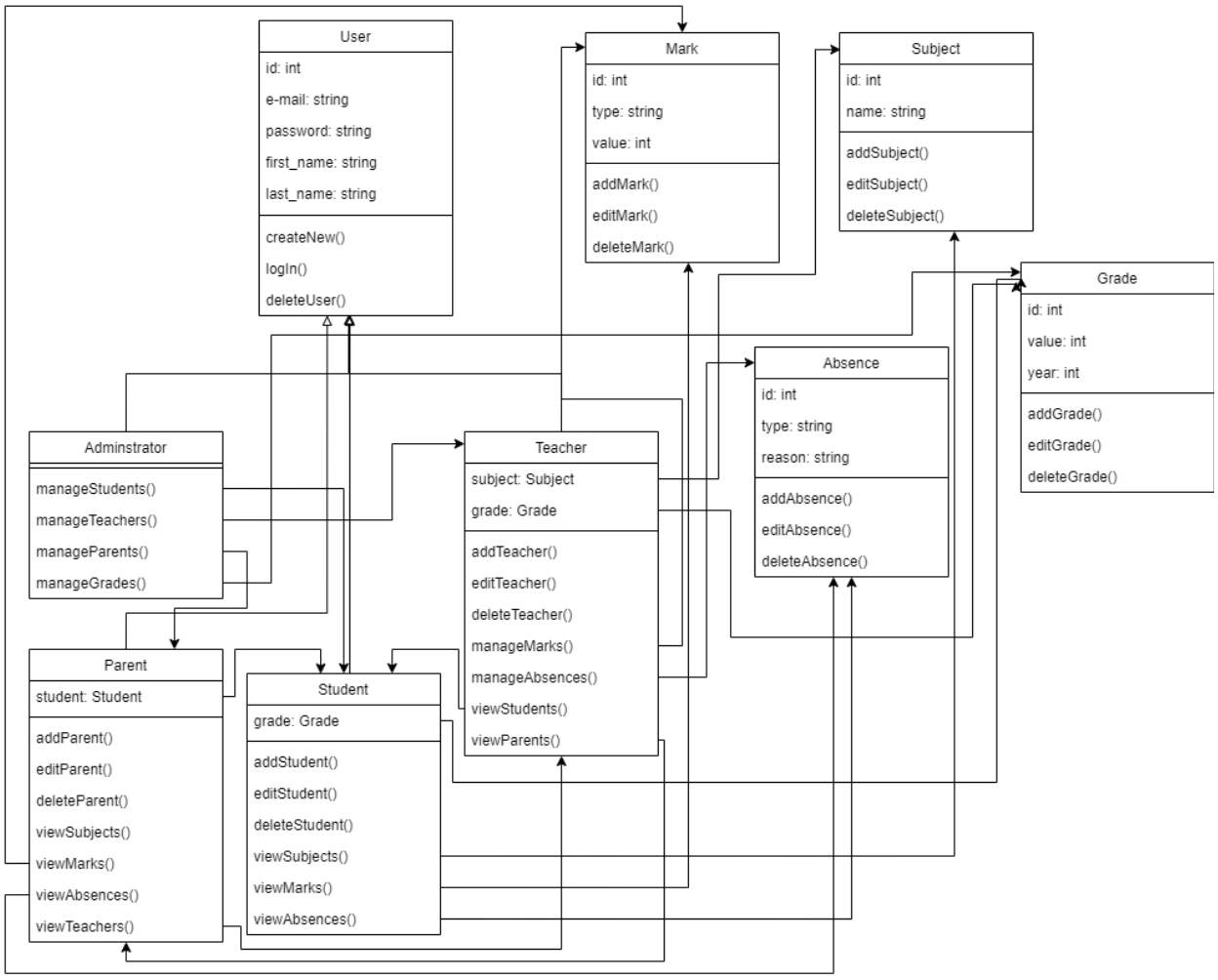
*да вижда своите оценки и отсъствия

*дава обратна връзка за начина на преподаване на всички негови учители

*създава индивидуален или групов чат

*Получава уведомления за отсъствията си и за всички събития в календара

2. Логически изглед:



Фиг. Клас-диаграма на онлайн училищен дневник

Клас-диаграмата на онлайн училищния дневник описва структурата на класовете на системата, техните атрибути, методи и връзките между компонентите.

Основни класове на онлайн дневника:

- **User**: Клас за потребител. Родителски клас на класовете Administrator, Teacher, Student и Parent.
    - **Administrator**: Клас за администратор. Наследява User класа. Управлява всички ученици, учители, родители и класове.
    - **Teacher**: Клас за учител. Наследява User класа. Управлява всички оценки и отсъствия, възможност за преглед на ученици и родители.
    - **Student**: Клас за ученик. Наследява User класа. Възможност за преглед на оценки, отсъствия и предмети.
    - **Parent**: Клас за родител. Наследява User класа. Възможност за преглед на предмети, оценки, отсъствия и учители.
- **Mark**: Клас за оценките на учениците.
- **Subject**: Клас за училищните предмети на учениците.
- **Grade**: Клас за училищните класове, в които принадлежат учениците.
- **Absences**: Клас за отсъствията на учениците.

Атрибути на класовете на онлайн дневника:

- **User**: id, e-mail, password, first_name, last_name;
    - **Administrator**: User атрибути;
    - **Teacher**: User атрибути + subject, grade;
    - **Student**: User атрибути + grade;
    - **Parent**: User атрибути + student;
- **Mark**: id, type, value;
- **Subject**: id, name
- **Grade**: id, value, year;
- **Absences**: id, type, reason;

Методи на класовете на онлайн дневника:

- **User**: createNew() – създаване на нов потребител;

  logIn() – влизане в системата;

  deleteUser() – изтриване на потребител;
    - **Administrator**: manageStudents() – управление на учениците;

manageTeachers() – управление на учителите;

manageParents() – управление на родителите;

manageGrades() – управление на класовете;

- **Teacher**: addTeacher() – добавяне на учител;

  editTeacher() – редактиране на учител;

  deleteTeacher() – изтриване на учител;

  manageMarks() – управление на оценки;

  manageAbsences() – управление на отсъствия;

  viewStudents() – разглеждане на ученици;

  viewParents() – разглеждане на родители;

- **Student**: addStudent() – добавяне на ученик;

  editStudent() – редактиране на ученик;

  deleteStudent() – изтриване на ученик;

  viewSubjects() – разглеждане на предмети;

  viewMarks() – разглеждане на оценки;

  viewAbsences() – разглеждане на отсъствия;

- **Parent**: addParent() – добавяне на родител;

  editParent() – редактиране на родител;

  deleteParent() – изтриване на родител;

  viewSubjects() – разглеждане на предмети;

  viewMarks() – разглеждане на оценки;

  viewAbsences() – разглеждане на отсъствия;

  viewTeachers() – разглеждане на учители;

- **Mark**: addMark() – добавяне на оценка;

editMark() – редактиране на оценка;

deleteMark() – изтриване на оценка;

- **Subject**: addSubject() – добавяне на предмет;

  editSubject() – редактиране на предмет;

  deleteSubject() – изтриване на предмет;

- **Grade**: addGrade() – добавяне на клас;

  editGrade() – редактиране на клас;

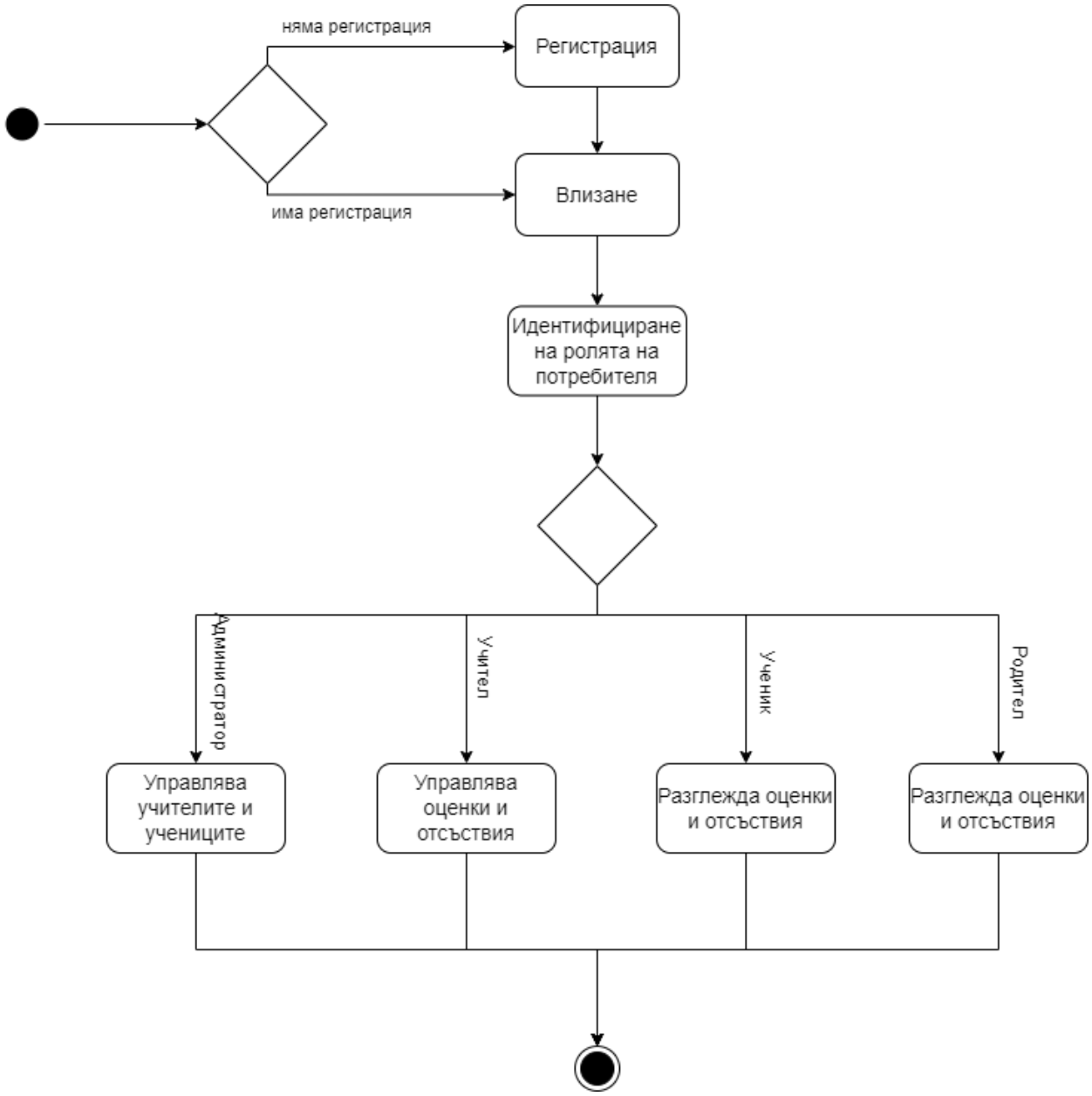
  deleteGrade() – изтриване на клас;

- **Absences**: addAbsence() – добавяне на отсъствие;

  editAbsence() – редактиране на отсъствие;

  deleteAbsence() – изтриване на отсъствие;

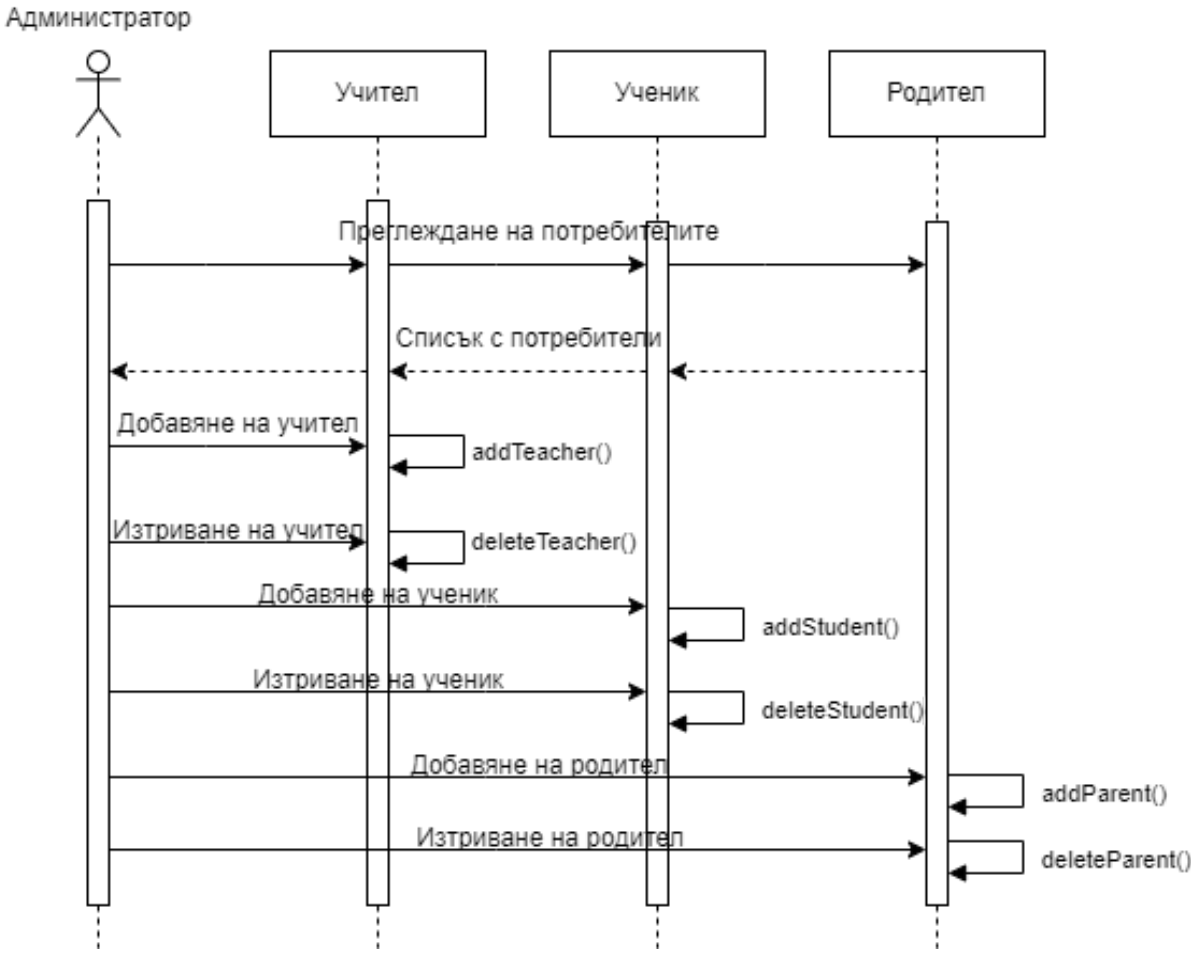
3. Процесен изглед:



Фиг. Обобщена диаграма на действията на онлайн училищен дневник

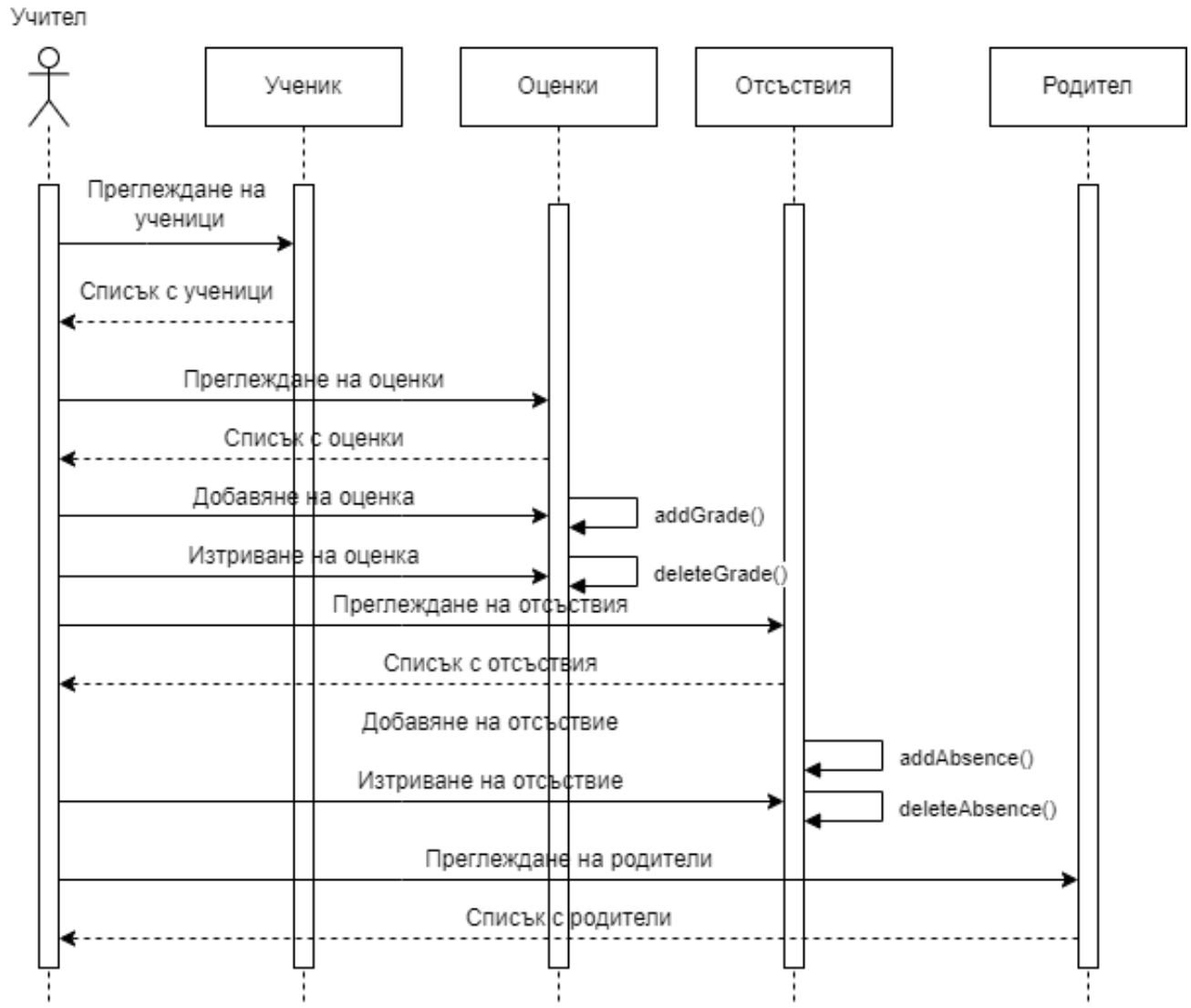
Това е обобщена диаграма на действията при влизане в системата на онлайн училищния дневник. Тя показва стъпките на дейността при влизане, където потребителят ще може да влиза, използвайки своят имейл и парола, ако има регистрация. При случая, в който няма регистрация, потребителят ще има възможност да се регистрира.

След влизане в системата се извършва идентификация на ролята му. В зависимост от ролята си, потребителят има достъп само до определени ресурси и може да извършва съответните определени действия.



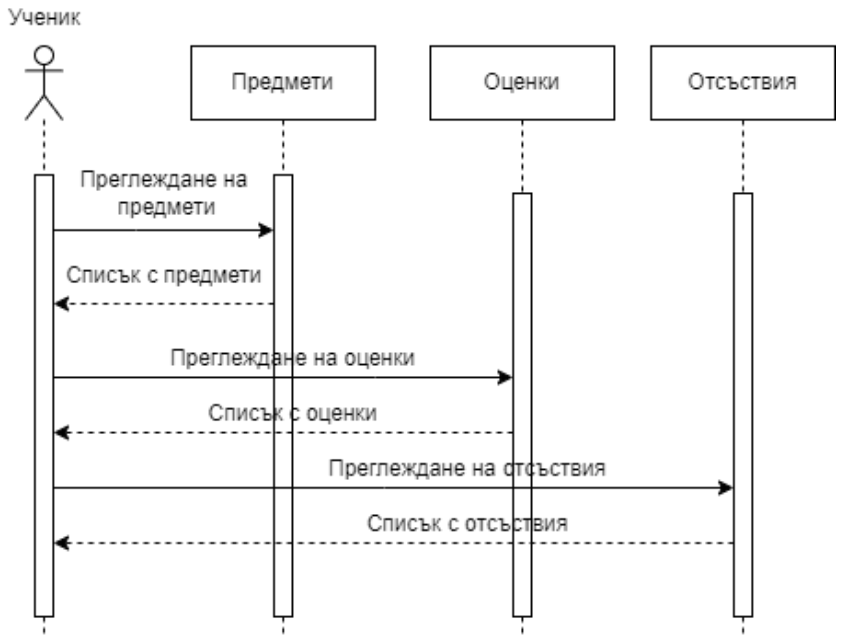
Фиг. Диаграма на последователността на роля Администратор

При идентифицирана роля Администратор, потребителят има правото да преглежда и управлява всички други потребители на системата (да добавя и изтрива съответен потребител).



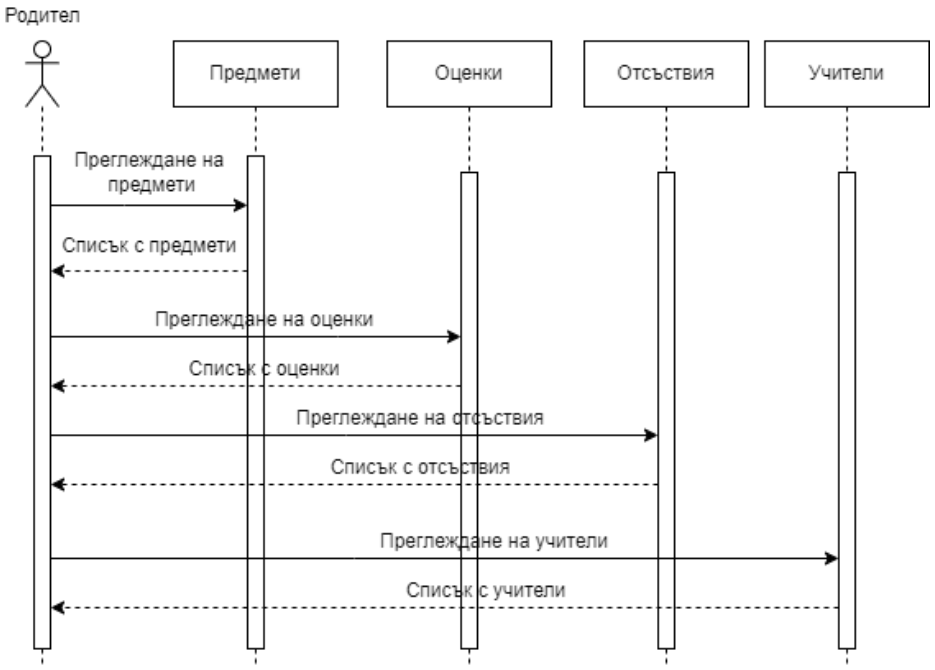
Фиг. Диаграма на последователността на роля Учител

При идентифицирана роля Учител, потребителят има правото да преглежда учениците, оценките, отсъствията и родителите, както и да управлява оценките и отсъствията.



Фиг. Диаграма на последователността на роля Ученик

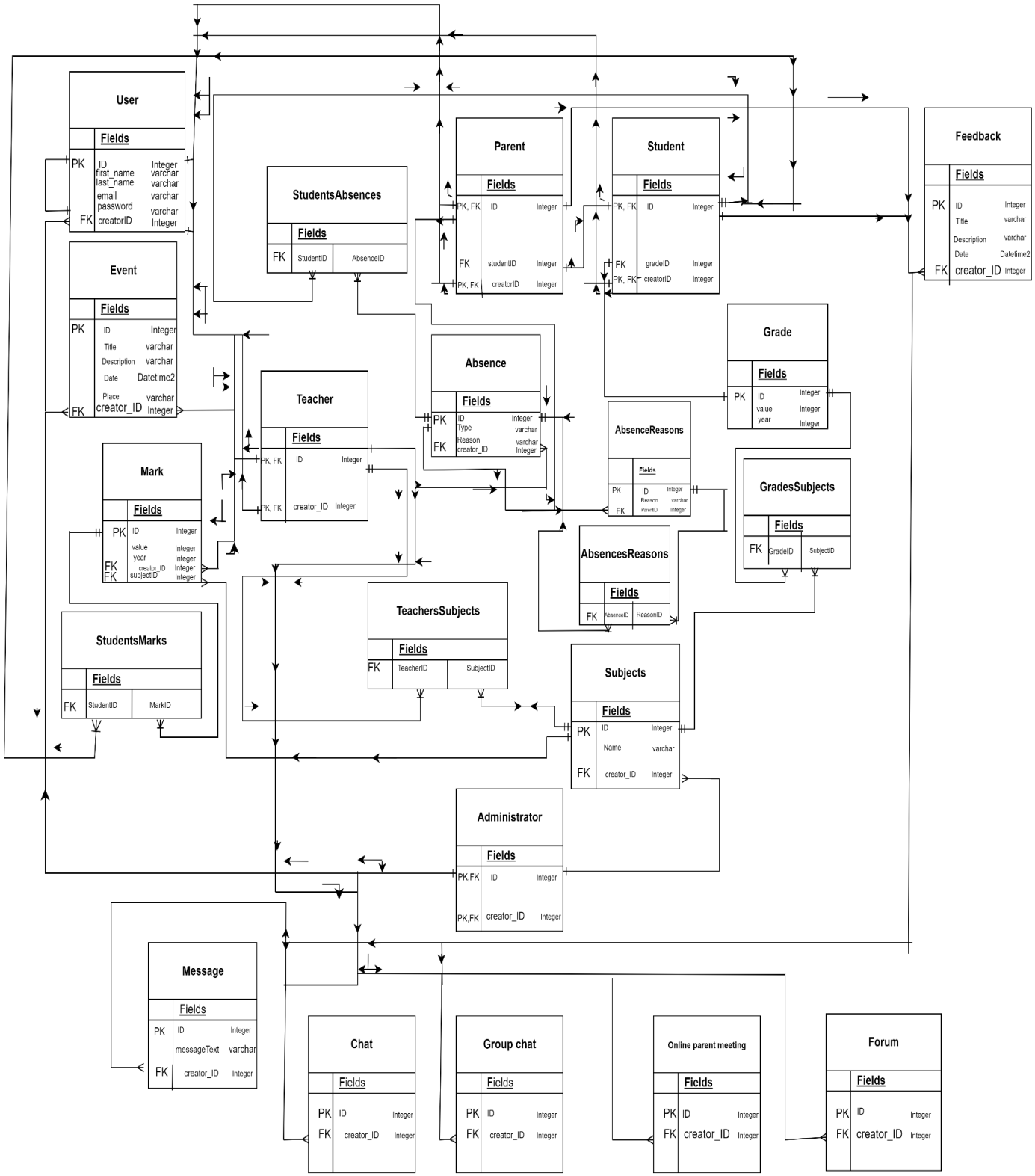
При идентифицирана роля Ученик, потребителят има правото да преглежда своите предмети, оценки и отсъствия.



Фиг. Диаграма на последователността на роля Родител

При идентифицирана роля Родител, потребителят има правото да преглежда предметите, оценките, отсъствията и учителите на своят ученик/ученици.

4. Изглед на данните:



5. Изглед на внедряването:

Хардуерни ресурси:

- **Уеб сървър:** За предоставяне на онлайн достъп до ученическия дневник.
- **База данни:** Съхранява информацията за оценките, отсъствията, отзивите и всички данни, описани подробно в раздела "Бизнес процеси в организацията".
- **Устройства:** Предоставят достъп до уеб приложението за родителите, учениците и учителите.

Комуникация:

- **Интернет:** Осигурява свързването между училището и външните устройства като устройствата на родителите, за достъп до онлайн дневника.
- **Локална мрежа:** Използва се за свързване между уеб сървъра и базата данни. Това улеснява устраняването на забавянето, предизвикано от мрежата, като се използва една инстанция на сървъра и базата данни, които отговарят на изискванията.

Предположения за внедряването:

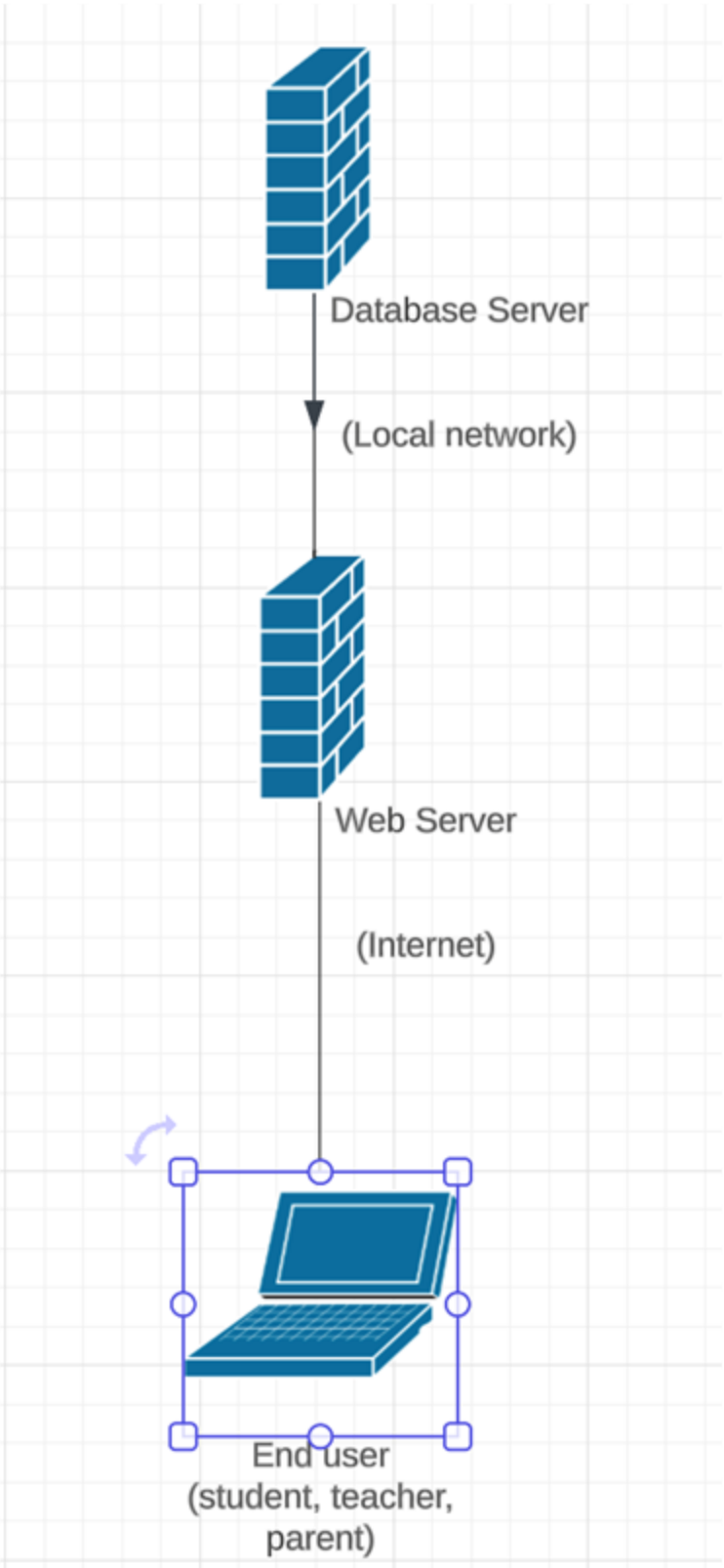
- **Сигурност:** Използване на шифроване на данни и сигурностни мерки за защита на чувствителната информация на учениците.
- **Бекъп и Възстановяване:** Редовно архивиране на данни и процедури за възстановяване в случай на загуба на информация.
- **Обновления и Поддръжка:** Периодични актуализации на софтуера и поддръжка, за да се гарантира стабилност и безпроблемна работа.
- **Скалируемост:** Инфраструктурата е гъвкава, за да се справи с увеличаващия се брой на потребители и данни.

Същност на Връзката:

Поставянето на базата данни и уеб сървъра на една локална мрежа помага за устраняване на забавянето, свързано с мрежата. Чрез една инстанция на сървъра и базата данни се удовлетворяват изискванията.

Съхранение на Данни:

За да се гарантира непрекъснат достъп и атомарност на транзакциите, уеб сървърът използва механизъм за повторни опити, който валидира и гарантира целостта на данните, записани в базата данни.


Database Server
(Local network)
Web Server
(Internet)
End user
(student, teacher,
parent)

6. Изглед на имплементацията:

За имплементацията на ученическия дневник се изисква добре структурирана архитектура, която да обхване всички необходими слоеве и ресурси за ефективно функциониране на системата.

Слоеве на архитектурата:

1. **Потребителски интерфейс (UI):** Този слой се отнася до визуалната част на приложението, която взаимодейства с потребителите (родители, учители, ученици). Тук се включват екрани, дизайн и интерактивността на приложението.

2. **Бизнес логика:** Слой, който съдържа логиката на приложението - обработка на данни, правила за достъп, изчисления на оценки и управление на информацията за учениците.

3. **Достъп до данни (Data Access – object-relational mapping):** Този слой е отговорен за свързване с базата данни, извличане и запис на информацията. Тук се извършва връзката със сървъра и базата данни.

Преизползваеми елементи:

- **Библиотеки и Функционалности:** Вече съществуващи компоненти на софтуера, като аутентикация, системи за управление на сесии и библиотеки за визуализация на данни. Тези елементи се преизползват като модули за сигурност, сесии и визуализация.

- **Наши Кодови Ресурси:** Съществуващ код за работа с база данни, обработка на заявки и обща функционалност. Тези ресурси ще бъдат използвани в новия проект за осигуряване на бърза и ефективна имплементация.

Общ вид на архитектурата:

Архитектурата е базирана на модулна структура, която използва различни слоеве за разделение на функционалността. UI слоят осигурява интеракцията с потребителите, бизнес логиката управлява обработката на данни, а слоят за достъп до данни се грижи за комуникацията с базата данни.

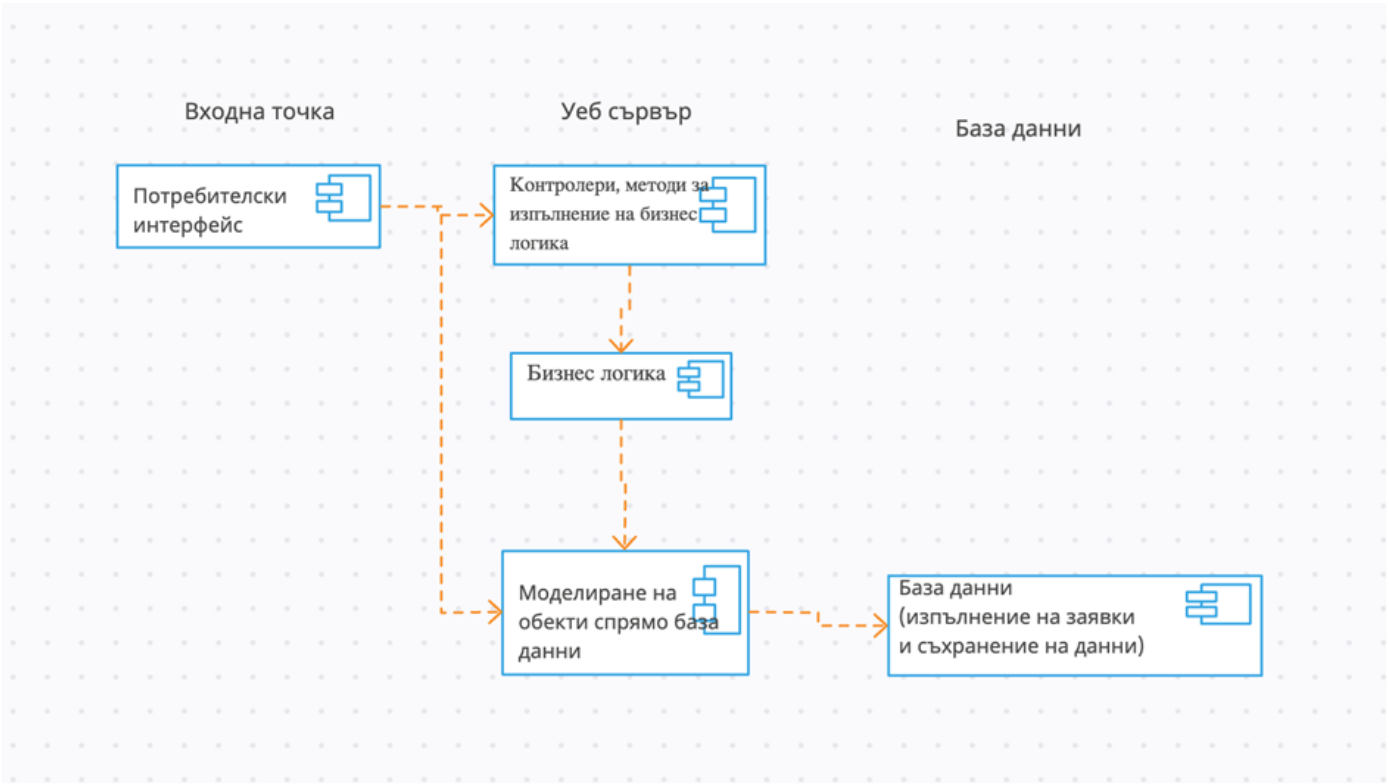
Ключови елементи и връзки между тях:

1. **Модул за Управление на Потребителски Интерфейс:** Отговаря за представянето на информацията към потребителите, предоставя възможност за вход, визуализация на оценките и информацията за отсъствията.

2. **Бизнес Модул за Обработка на Данни:** Включва основните функции за обработка на данни, като изчисления на оценки, генериране на отчети и обработка на отзиви.

3. **Модул за Достъп до База Данни:** Отговаря за свързване с базата данни, извличане и запис на информацията, взаимодействайки със слоя на бизнес логиката.

Контролът и координацията между тези модули гарантират гладкото функциониране на цялостната система за ученически дневник.


Входна точка
Уеб сървър
База данни
Потребителски интерфейс
Контролери, методи за изпълнение на бизнес логика
Бизнес логика
Моделиране на обекти спрямо база данни
База данни (изпълнение на заявки и съхранение на данни)

Нефункционални изисквания

1. Достъпност

1.1. Възможности за отказ на системата

1.1.1. Технически откази

1.1.1.1. Хардуерен отказ – неизправност на сървъра или друго хардуерно устройство

1.1.1.2. Софтуерен отказ – проблем със софтуера на системата

1.1.2. Човешки грешки

1.1.2.1. Административна грешка

1.1.2.2. Човешка грешка – грешка на потребител, който въвежда неправилна информация

1.2. Възможности за възстановяване на работоспособност

1.2.1. Грешките трябва да бъдат обработвани програмно

1.2.1.1. Ако ползвателят получи грешка, в базата данни няма да се запишат данните

1.2.1.2. Грешката се описва в JSON формат и съдържа свойство error, което съдържа кода и описанието на грешката

1.2.2. Системата трябва да има механизми за бързо откриване на откази, които да бъдат докладвани на администратора

1.2.1.1. Редовно архивиране на данни

1.2.1.2. Автоматично рестартиране на системата когато системата показва грешки или предупреждения

1.2.1.3. В случай на загуба на информация се изпълняват процедури за възстановяване

1.2.3. Съвместимост с различни платформи: Системата трябва да бъде съвместима с различни операционни системи и уеб браузъри, за да осигури равен достъп за всички потребители.

2. Разширяемост

2.1. Архитектурен дизайн

2.1.1. Системата трябва да бъде проектирана с гъвкава и мащабируема архитектура, която позволява лесно добавяне на нови функционалности и модули.

2.1.1.1. Restful API – улеснява добавянето на функционалности без да оказва влияние върху останалата част от системата

2.1.1.2. Използване на добре документирани и тествани API

2.1.1.3. Използване на модулна структура

2.1.1.3.1. Улеснява добавянето или актуализирането на функционалности без промени в останалите части на системата.

2.1.1.3.2. Улеснява идентифицирането и отстраняването на грешки

2.1.1.3.3. Взаимодействието между отделните модули е малко, което означава че те са по-независими

2.1.1.3.4. Прави системата по-лесна за разбиране и поддръжка, което води до по-бързо внедряване на промени

2.1.2. Използването на описаните похвати за проектиране може да помогне за намаляване

на разходите при промяна или актуализация на софтуера.

2.2. Дизайн на данни

2.2.1. Използване на добре документирани и тествани данни

2.3. Дизайн на интерфейс

2.3.1. Използване на интуитивен и лесен за използване интерфейс

2.4. Технологии за разработка

2.4.1. Използване на технологии, които са независими и могат лесно да бъдат заменени, без да се налага да се правят съществени промени в основната функционалност на системата

2.4.1.1. Езици за програмиране –JavaScript, TypeScript

2.4.1.2. Библиотека – Angular

2.4.1.3. Инфраструктура – Node.js

2.4.2. Възможност за бързо и лесно интегриране на нови технологии и библиотеки, които да помогнат за подобряването на системата

3. Прозводителност

3.1. Тактики за подобряване на производителността

3.1.1. Ограничаване на натоварването

3.1.1.1. Подходи за намаляване на количеството работа, която системата трябва да изпълнява

3.1.1.1.1. Ограничаване на броя на потребителите, които могат да получат достъп до системата едновременно

3.1.1.1.2. Ограничаване на количеството данни, които системата трябва да обработва.

3.1.1.2. Примери

3.1.1.2.1. Системата трябва да може да обслужва 1000 потребители едновременно.

3.1.1.1.2. Системата трябва да може да обработва 1 милион заявки на час

3.1.2. Планиране

3.1.2.1. Фактори при планирането

3.1.2.1.1. Брой потребители

3.1.2.1.2. Брой заявки

3.1.2.1.3. Сложност на заявките

3.1.2.2. Тактики за подобряване на производителността при планиране

3.1.2.2.1. Използване на подходяща архитектура

3.1.2.2.2. Оптимизиране на кода

3.1.2.2.3. Използване на инструменти за планиране и мониторинг, които позволяват проследяване на производителността и оптимизиране на ресурсите в реално време

3.1.3. Репликиране на системите

3.1.3.1. Тактики за подобряване на производителността при репликация

3.1.3.1.1. Използване на репликация за балансиране на натоварването (load balancing) между различните сървъри за да се осигури равномерно разпределение на заявките за данни и намаляне на времето за реакция за всеки отделен потребител

3.1.3.1.1.1. Уеб страниците трябва да бъдат оптимизирани за бързо зареждане, минимизирайки използването на bandwidth и ресурси на клиентските устройства

3.1.3.1.2. Системата трябва да бъде оптимизирана за ефективно използване на процесорна мощност, памет и дисково пространство, за да осигури бърза обработка на заявките и ниска латентност

3.1.3.1.3. Системата трябва да бъде оптимизирана за ефективно използване на хардуерните ресурси на сървъра, като памет и процесорна мощност

4. Сигурност

4.1. Използване на шифроване на данни и сигурностни мерки за защита

4.1.1. Всички лични данни, включително потребителска информация, оценки и съобщения, трябва да бъдат криптирани, за да се осигури поверителност и да не се позволява разглеждането на чужди данни.

4.1.1.1. Всички данни, предавани между клиента и сървъра, трябва да бъдат криптирани, за да се осигури защита

4.1.2. Системата трябва да гарантира целостта на данните, предотвратявайки грешки или загуба на информация при трансфер или обработка.

4.1.3. Редовно създаване на резервни копия на данните, за да се осигури възможно най-бързо възстановяване в случай на атаки или загуба на информация.

4.2. Управление на достъпа

4.2.1. Ограничаване на достъпа до системата в зависимост от ролите на ползвателите

4.2.2. Разграничаване на правата за четене, писане и изтриване

4.3. Защита от уеб уязвимости

4.3.1. Защита от атаки като SQL injection и XSS атаки.

4.3.1.1. Системата трябва да включва мерки за превенция на атаки като SQL injections, за да се гарантира защита на услугата от злонамерени атаки.

4.4. Сигурност и удостоверяване

4.4.1. Използване на механизми за удостоверяване като автентикация с потребителско име и парола и авторизация.

4.4.2. Разработка на стратегия за сигурност и редовно актуализиране на системата.

5. Възможност за тестване

5.1. Тактики за тестване

5.1.1. Дефиниране и налагане на различни тестови случаи, които включват функционални тестове, тестове за сигурност, тестове за производителност и тестове за устойчивост

5.1.1.1. Гарантира се , че системата е стабилна и работи безпроблемно.

5.1.1.2. Помага за идентифициране на проблеми, които могат да повлияят на производителността.

5.1.2. Тестване на интеграцията между различните компоненти и модули на системата.

5.1.3. Тестване на цялата система в различни условия и сценарии

5.1.3.1. Провеждане на тестове с потребители, за да се оцени използваемостта на интерфейса и да се направят корекции според реалните потребителски нужди

5.1.3.2. Интегрираните технологии могат да помогнат за автоматизиране на процеса на тестване, както и за по-бързото и ефективно изпълнение на тестовете

5.1.4. Изисквания за подобряване на производителноста чрез тестване

5.1.4.1. Използване на подходяща инфраструктура

5.1.4.2. Използване на правилните инструменти и техники

5.2. Цена за провеждане на тестовете

5.2.1. Фактори, които определят цената за провеждане на тестове

5.2.1.1. Обхватът на тестовете

5.2.1.2. Използваните технологии и инструменти

5.2.2. Категории за цена за провеждане на тестове

5.2.2.1. Цена за идентифициране на проблеми

5.2.2.1.1. може да бъде значителна, особено ако се изискват обширни тестове

5.2.2.2. Цена за отстраняване на проблеми

5.2.2.2.1. може да бъде по-висока от цената за идентифициране на проблеми

6. Интероперабилност

6.1. Начини за подобряване на производителността

6.1.1. Намаляване на дублирането на процеси

6.1.2. Намаляване на дублирането на данни

6.2. Тактики за подобряване на производителността

6.2.1. Използване на стандарти

6.2.1.1. Използване на стандартизирани протоколи и формати за обмен на данни, като JSON и RESTful APIs, за да се позволи лесния обмен на информация с други системи

6.2.2. Лесна интеграция с други системи: Да има възможност за лесна интеграция с други образователни платформи и софтуерни приложения за обмен на данни и ресурси.

6.2.2.1. Възможност за лесно импортиране и експортиране на данни, позволяващо на потребителите да прехвърлят информация между различни системи и приложения.

7. Използваемост

7.1. Начини за подобряване на производителността

7.1.1. Подобрена ефективност и производителност

7.1.2. Намаляване на грешките при използване на софтуера

7.2. Тактики за подобряване на производителността

7.2.1. Използване на стандартни потребителски интерфейси

7.2.1.1. Добра организация на информацията, лесна навигация и ясна структура на интерфейса, за да се улесни достъпът до различните функции и материали на платформата