LeeTUcode

Антон Дюлгеров, Антон Костов, Велислав Боянов, Ростислав Банков, Теодор Йосифов, Християн Кушев

21.10.2023 г.

Последна ревизия: 08.01.2024 г.

1. Описание на основните цели и характеристики, представяне на групите участници, участващи в проекта

Основната цел на LeeTUcode е да предостави една модулна, гъвкава и мащабируема архитектура, която може да обработва адекватно количество от потребителски заявки едновременно, да предлага висока ефективност и да осигурява адекватно ниво на защита на личните данни на потребителите.

Архитектурата ще се състои от различни отделни модули, които ще обработват отделните функционалности на приложението. Тези модули ще комуникират помежду си чрез специални функции предназначени за комуникация, което ще позволи лесното им скалиране.

Софтуерната архитектура ще поддържа един основен потребителски интерфейс - уеб страница, като ще осигурява регистрация и вход на потребители в системата с различни роли - учител, ученик и не-регистриран потребител. Информацията за потребителите ще бъде съхранявана в база данни, като ще има възможност за търсене и филтриране на потребителите.

Системата ще поддържа различни курсове и задача, информацията за които ще бъде съхранявана в база данни с възможност за търсене и филтриране.

Системата ще осигурява сигурност на данните, като ще поддържа криптиране на пароли използвайки “bcrypt” функцията за криптиране на пароли.

Като цяло, софтуерната архитектура ще бъде проектирана така, че да осигури максимална, ефективност, сигурност и мащабируемост, като същевременно да осигури лесна поддръжка и разширяемост и да предоставя необходимата функционалност за управление на потребителите, курсовете и задачите.

Работата по проектът ще бъде разделена на няколко основни екипа.

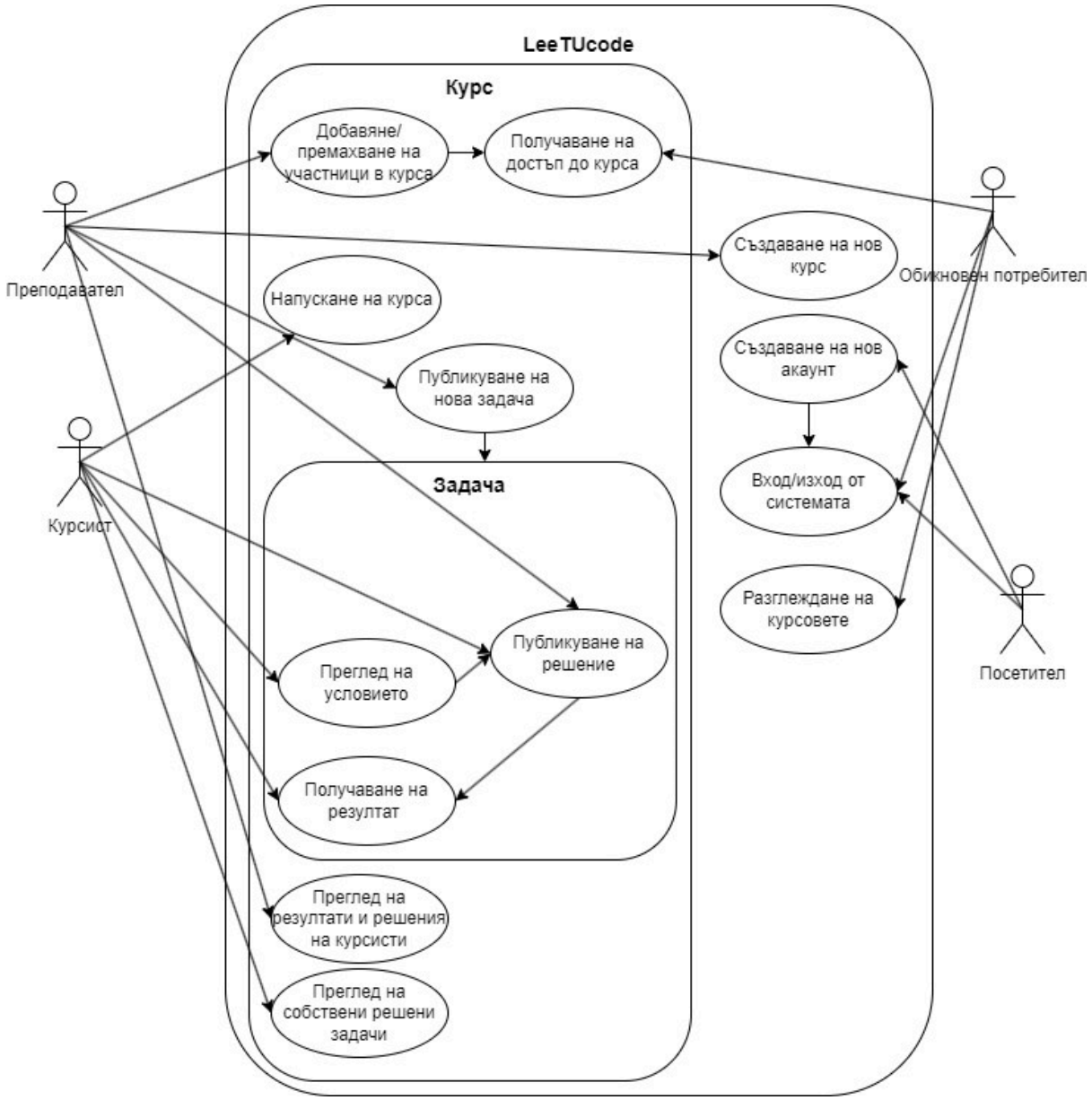
1. Frontend екип - основната задача на участниците в този екип ще бъде да измислят и имплементират интуитивен и лесен за използване дизайн на web приложението, така че всеки потребител да може лесно и бързо да намира частите, от които е заинтересован. Основните технологии, които ще използва този екип са HTML, CSS и JavaScript/Jquery

2. Backend екип - целта на този екип е да имплементират различната функционалност на страницата. Те ще трябва да подсигурят надеждността

и бързината на използване на страницата. Основните технологии, които ще използва този екип са PHP и JavaScript/Ajax

3. Fullstack екип - този екип ще се занимава занимава с цялата система, там където са нужни. Целта на участниците ще е подпомагане и връзка между другите два екипа.

2. Use-case изглед



Ползвателите на сайта могат да бъдат:

- преподавател, който на практика има администраторски права върху дадена подсистема (в случая обособен курс)
- курсисти, които представляват обикновени потребители с достъп до курса
- обикновени потребители на системата, когато такива не се разглеждат спрямо даден курс
- посетители на сайта или неавтентифицирани потребители

Посетителите, които не са влезли в системата, ще бъдат поканени да създадат акаунт или да влязат в системата с вече съществуващ такъв. Вече влезли в системата, те могат да създадат нов курс, който да управляват като учител, или да

бъдат присъединени към вече съществуващ такъв като учащи. Последните могат да търсят вече създадени курсове и да се присъединят към такива. Системата не изисква документи, удостоверяващи легитимността на даден курс в образователната сфера.

Като ученици, потребителите трябва да бъдат известявани при публикуването на нова задача към курсовете, в които принадлежат, за да могат да я изпълнят. Те трябва да могат да разглеждат предишни свои задачи и при решаване на такива да получават директно резултат. Същите резултати трябва да са достъпни и за преподавателите. Преподавателите остава да са единствените, които да могат да създават или променят задачи в курса. Курсистите обаче не могат да виждат чужди резултати. Те нямат и никакъв достъп до дадена задача, докато не им бъде предоставен такъв от преподавателя.

Системата следва да удовлетворя това обособяване на ролите на потребителите и възможните начини, по които те могат да я използват.

3. Логически изглед

Основните класове нужни за изработката на LeeTUcode са:

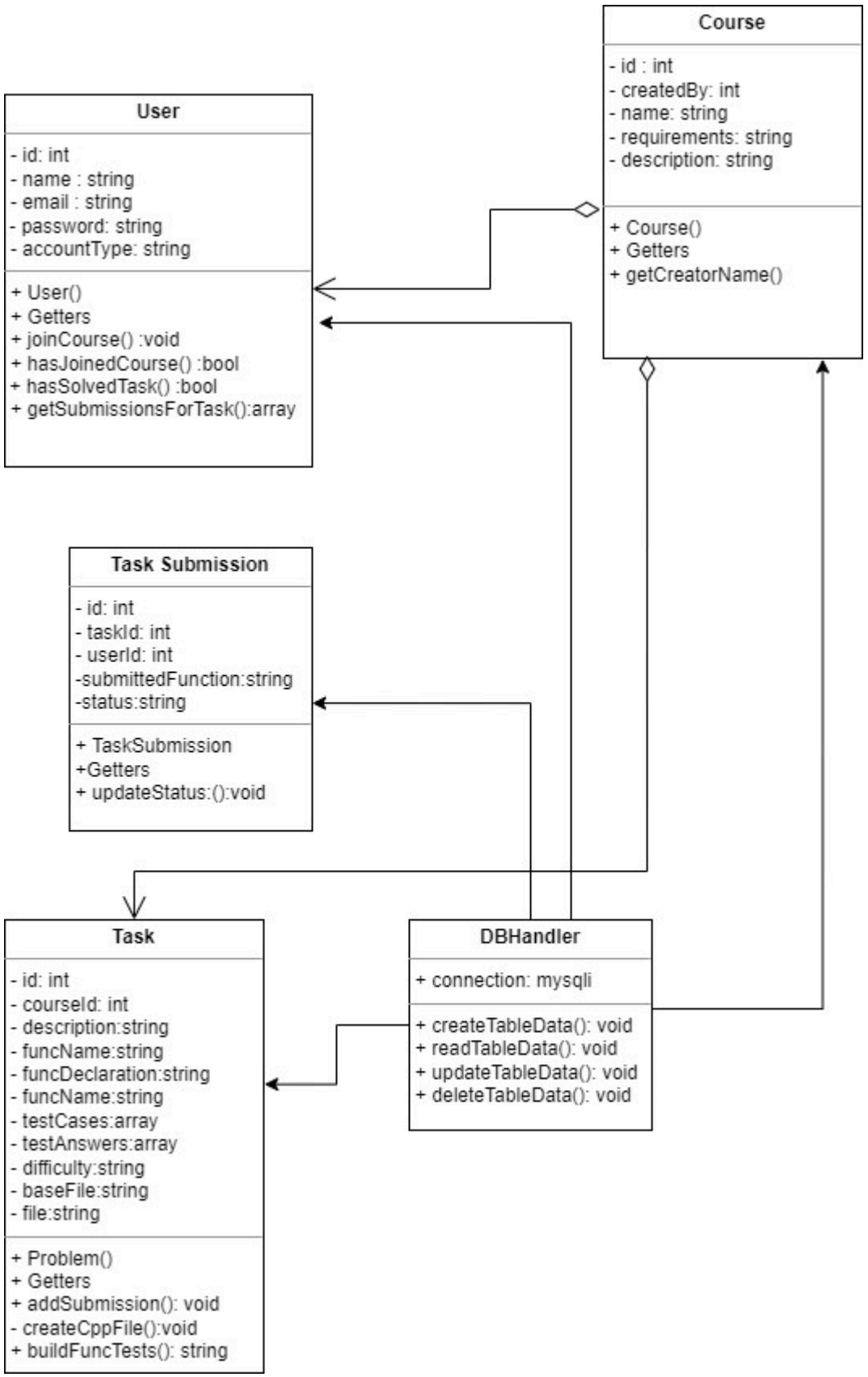
1.user - този клас съдържа информация за всички регистрирани потребители в системата. Съдържа информация за ID на потребителя, потребителско име, имейл, хеширана парола, вид на акаунта. Трябва да има функционалност за взимане и променяне на тези данни, както и начин за инициализиране на обект от класа с данни получени от база данни.

2.course - този клас съдържа информация за всички добавени от акаунти от тип учител курсове. Съдържа информация за ID на курса, име на курса, описание на курса и изискванията му, както и ID на акаунтът създал курса. Трябва да има функционалност за взимане и променяне на тези данни, както и начин за инициализиране на обект от класа с данни получени от база данни. Освен това трябва да съдържа функционалност за добавяне на участници и задачи към курса.

3.task - този клас съдържа информация за всички добавени от акаунт тип учител задачи. Съдържа информация за ID на задачата, име на задачата, описание на задачата, име и декларация на функцията, чиято имплементация трябва да се предостави за успешното и решаване, тестовите случаи и отговори, трудността и, както и ID на курсът към който принадлежи задачата. Трябва да има функционалност за взимане и променяне на тези данни, както и начин за инициализиране на обект от класа с данни получени от база данни.

4.dbHandler - този клас трябва да съдържа функционалност за писане и вземане на данни от дата база. Чрез класа трябва да може да се взима и съхранява информацията от другите класове в базата данни.

5.taskSubmission - този клас съдържа информация за публикувано решение на дадена задача. Съдържа информация за Id на задачата, към която принадлежи решението и на потребителя, който го е публикувал, за функцията, която е публикувана и резултата от решението.


Course
- id : int
- createdBy: int
- name: string
- requirements: string
- description: string
+ Course()
+ Getters
+ getCreatorName()
User
- id: int
- name : string
- email : string
- password: string
- accountType: string
+ User()
+ Getters
+ joinCourse() :void
+ hasJoinedCourse() :bool
+ hasSolvedTask() :bool
+ getSubmissionsForTask():array
Task Submission
- id: int
- taskId: int
- userId: int
-submittedFunction:string
-status:string
+ TaskSubmission
+Getters
+ updateStatus:():void
Task
- id: int
- courseId: int
- description:string
- funcName:string
- funcDeclaration:string
- funcName:string
- testCases:array
- testAnswers:array
- difficulty:string
- baseFile:string
- file:string
+ Problem()
+ Getters
+ addSubmission(): void
- createCppFile():void
+ buildFuncTests(): string
DBHandler
+ connection: mysqli
+ createTableData(): void
+ readTableData(): void
+ updateTableData(): void
+ deleteTableData(): void

4. Процесен изглед

1. Процес на регистрация на потребители: в този процес потребителите могат да се регистрират в системата на LeeTUcode чрез създаване на акаунт. Това може да включва въвеждане на лични данни, създаване на потребителско име и парола, и проверка на валидността на въведената информация. Резултатът от процеса е създаване на профил на потребителя в системата.

2. Процес вписване в системата - в този процес потребителите ще вписват своите данни, системата ще ги проверява за валидност и при успех, потребителят ще бъде вписан в своя профил.

3. Процес търсене и преглед на курс - в този процес потребителите ще могат да търсят и разглеждат вече създадените курсове. Това може да включва ключови думи и филтриране.

4. Процес включване към курс - в този процес потребителите ще могат да се включат в избран от тях курс, след като са го разгледали. Това може да включва изпращане на искане за достъп до курса към създателя на курса.

5. Процес приемане/отхвърляне на молба за включване към курс - в този процес създателите на курсовете ще могат да одобрят или да отхвърлят молбата на друг потребител за включване към дадения курс.

6. Процес създаване на курс - в този процес потребителите с акаунти от тип учител ще могат да създават свои курсове. Това може да включва добавяне на име и описание на курса.

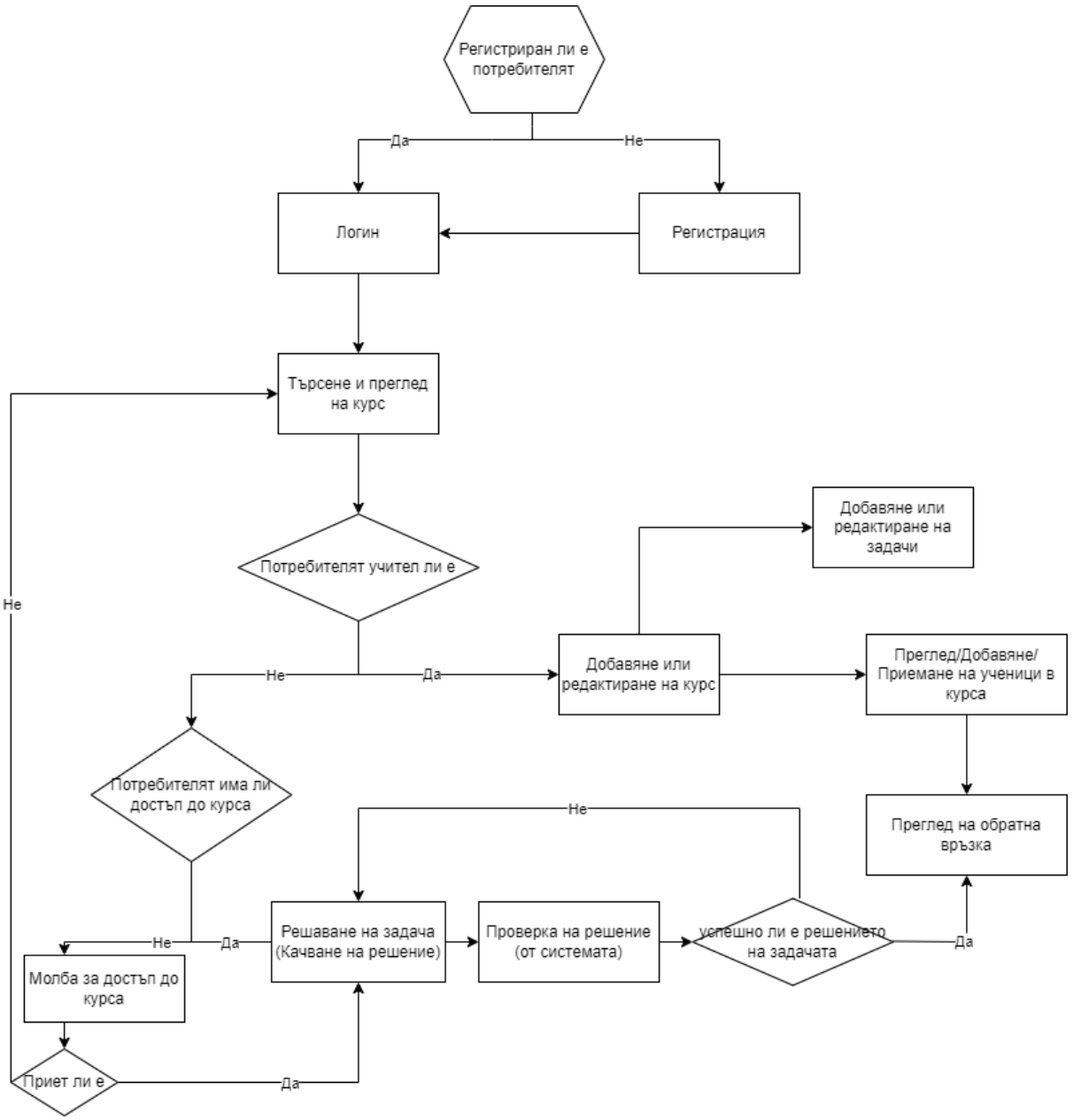
7. Процес добавяне на задача - в този процес потребителите с акаунти от тип учител ще могат да добавят задачи към своите курсове, които членовете на тези курсове да решават. Това може да включва добавяне на име и описание на задачата, както и добавяне на тестове за верифициране на решенията качени от членовете на курса.

8. Процес на качване на решение на задача - в този процес членовете на даден курс ще могат да качат своите решения за дадена задача под формата на функции с определено име и входни данни.

9. Процес проверка на решение качено от потребител - в този процес системата ще проверява каченото решение на дадена задача от даден потребител на база на тестовете качени от създателя на курса, към който принадлежи дадената задача.

10. Процес на даване на обратна връзка относно решението качено от потребител - в този процес системата ще дава обратна връзка за успешното/неуспешното решение на дадена задача след проверката на каченото решение. Това може да включва потвърждение, че задачата е била решена правилно или обратна връзка за това кой тест решението качено от потребителя не е преминал успешно.

11. Процес преглед на обратна връзка - в този процес потребителите качили решение на дадена задача, ще могат да прегледат обратната връзка дадена им от системата.


Регистриран ли е потребителят
Да
Не
Логин
Регистрация
Търсене и преглед на курс
Потребителят учител ли е
Не
Да
Добавяне или редактиране на задачи
Добавяне или редактиране на курс
Преглед/Добавяне/ Приемане на ученици в курса
Не
Потребителят има ли достъп до курса
Преглед на обратна връзка
Не
Да
Решаване на задача (Качване на решение)
Проверка на решение (от системата)
успешно ли е решението на задачата
Да
Не
Молба за достъп до курса
Приет ли е
Да

5. Изглед на данните

Данните ще бъдат съхраняване в MySQL база данни със следната структура на таблиците:

1.users:

int: id (Уникално ID, primary key)

varchar: username

varchar: email

varchar: password

varchar: accountType (студент, учител)

2.courses:

int: id (Уникално ID, primary key)

varchar: name

text: description

text: requirements

int: ownerID (ID на създателя на курса, foreign key)

3.problems:

int: id (Уникално ID, primary key)

varchar: name

varchar: function_name

varchar: function_declaration

varchar: name

text: description

text: test_cases

text: test_answers

int courseID (ID на курсът, към който принадлежи задачата, foreign key)

4.courseMembers

int: id (Уникално ID, primary key)

int courseID (ID на курсът, към който принадлежи потребителят, foreign key)

int: memberID (ID на потребителя член, foreign key)

5.solvedProblems

int: id (Уникално ID, primary key)

int: problemID (ID на решената задача, foreign key)

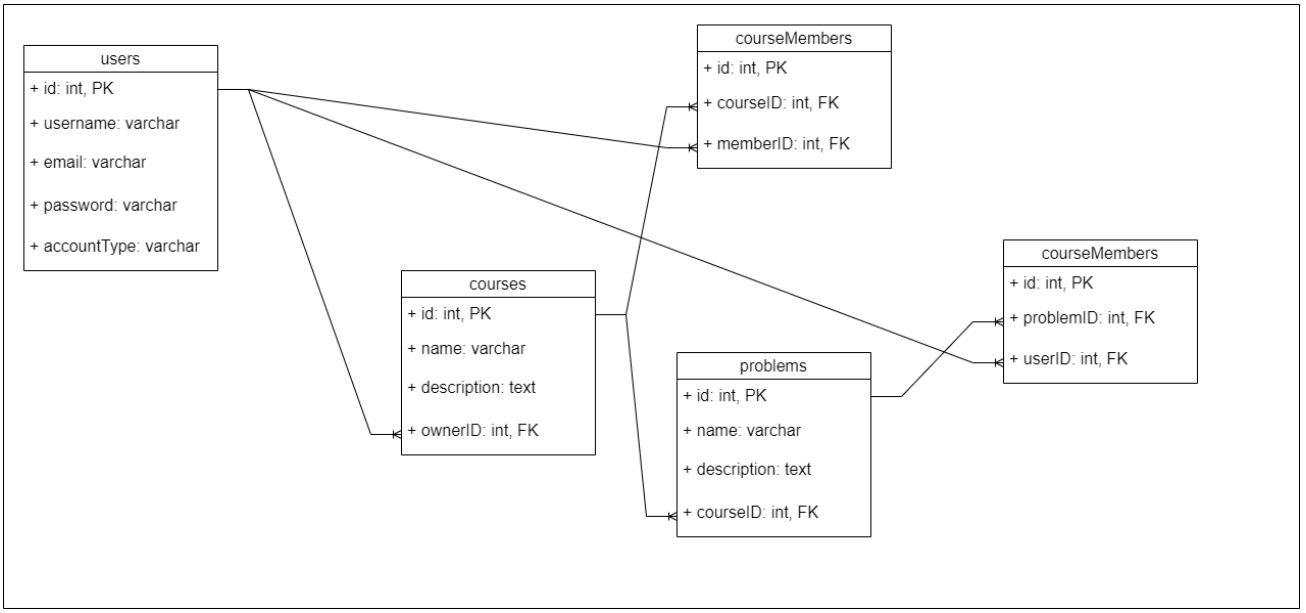
int: userID (ID на потребителя решил задачата, foreign key)

varchar: status

text: submitted_function

text: response

Следващата схема представя базата данни графично:



6. Обосновка за това, как избраната архитектура и подход осигурява адекватна реализация на въпросната софтуерна система

1. Достъпност – Използват се тактики за възстановяване на работоспособността на системата така нареченият disaster recovery plan. Това включва откриване на проблема, възстановяване на системата, и превенция на откритите проблеми. Тук може да се включи и анализиране (monitoring) на системата с цел превенция на проблеми.

2. Разширяемост – Тук се анализира възможностите за разширяване на системата и софтуера, тяхната цена. Може да се използват тактики с които да се смени цялостната архитектура чрез шаблони които да намалят цената и ресурсите използвани, както и технологиите на системата.

3. Производителност- Анализират се ресурсите нужни на системата за постигане на желаните резулати. Тук се използват похвати за планиране на натовареността и репликация на системите с идея намаляване на натовареността.

4. Сигурност- Анализират се средствата за защита, информацията както и оторизирания достъп. Използват се тактики за криптиране на чувствителни данни, ограничаване на достъпа до системата и наемане на екип по тестване на защитата с цел одит.

5. Възможност за тестване- Тук се анализира средствата за тестване чрез различни системи и/или хора с различни обхвати. Използват се тактики за одитиране на системата както и одитиране на самият програмен код. Може да се използват интегрирани тестове в системата за тестване.

6. Интероперабилност – Анализират се входовете и изходите на информация. Използват се похвати за тестване на информацията (дали е криптирана между изхода и даден компонент)и се използват тактики за превенция на течове.

7. Използваемост – Анализира се интерфейса за комуникация между софтуера и потребителя. Ползват се специалист в областта за тестване на интерфейса (QA) както и дизайнери за подобряване на качеството на интерфейса.