Утвърдил:
доц. др. инж. С. Стефанова
Директор ТУЕС към ТУ-София

Поправителен изпит по Технология на програмирането - практика

(05 септември 2018)

Трите имена:

Клас:

Номер в клас:

Sha256 стойност на предадения архив:

Име на родител:

Подпис на родител/настойник:
(целта на името и подписа на родителя е да съм сигурен, че родителите са запознати с това как са изградени задачите поставени на изпита. Начинът, по който са поставени задачите е описан по-долу и начинът на провеждане също)

Изпитните задачи се състоят от три елемента

1. Взима се задача дадена на миналата поправка през юлската сесия. Тя самата е изградена на база на задачи давани минимум 8 пъти в периода от март до юни 2018. Буквално са копирани задачите от миналия път. Предположението е, че ако те са решени успешно за последните 40 дни ще бъдат решени успешно и сега.
2. Към нея се добавят допълнителни условия от домашни давани в периода от септември 2017 до юни 2018. Взима се домашно и се добавя към задачата за изпита. Задачите от домашните са решени през учебната година, има техни решения в хранилището на предмета. Ако човек се е запознал с решенията на задачите от домашното за последните месеци може директно да ги използва.
3. Правят се допълнения на база предоставения конспект. Една или две точки върху equals и hashCode.
4. Учениците не могат да ползват интернет по време на работа. Възможно е задачата им да бъде решена от друг ученик и да им бъде изпратена и това няма как да бъде контролирано.

5. Възможно е обаче да ползват **материали, които да донесат на изпита**. Това е възможно да бъдат решения на задачите от домашните работи, задачите от поставените контролни и задачите от поправителните сесии, цялото хранилище от задачи от годината. Възможно е да вземат допълнителни библиотеки, документации и всички необходимо.
6. В тази връзка задачите ако човек не е подготвен са обективно трудни, но ако са решени всички домашно от година, ако са решение предишните задачи от поправката и ако ученикът се е сетил да вземе тези материали за поправката, задачите се превръщат в обективно постижими.
7. На края на изпита трябва да се създаде архив съдържащ работата на ученика и всичко върху което е работено и този архив да се предаде.

Задание

Вариант 1

Tasks Objects

1. Create Task on url /tasks/new with the following field
    - name
    - description
    - solution_required - true/false value if a solution should be provided for the task to me marked as completed. A check box in the form

    Implement the TasksController to have a show, update, create, edit, destroy, index methods that follow the rails conversion for behaviour of these methods

2. Create TaskSolution /task_solutions/new with
    - picture solution - field for providing a picture solution with an upload button in he form
    - status - enum of values {:not_started, :started, :completed}. A drop down in the form
    - confirmed - true/false value if the task is confirmed

    Implement the TaskSolutionsControler to have a show, update, create, edit, destory, index methods that follow the rails conversion for behaviour of these methods

3. One task could have many task solutions. One task solution is for a specific task
4. Validate that the number of tasks solutions created for a given task that have a blank text solution is not more than 3
5. Validate that no more than 3 tasks requiring a solution could be created

6. Validate that for a task to be marked as completed when a solution is required than there should be a picture solution provided and that after a task is marked as confirmed the picture and status could not be changed.
7. Show a list of tasks completions and for which task they are
    - this is available at /#{my_index}tasks/ where my_index is your class_number_firstname_lastname
    - show a table with the first column the task name, the second column the specialty description, the third column whether a solution is required, the fourth column the text task solution, the fifth column the status of the solution and the six column whether the task is confirmed

User Objects

8. Allow a user to be registered on the platform by providing a user name and password. Users should not be valid unless confirmed. Confirmation is done by setting the value of a field “confirmed” in the user model to ‘true’.
9. Each TaskSolution could be completed by a many users. Each user could have many task completions
10. Show all the TaskSolution for a given user on the /users/:id/task_completions url
11. Make sure that each user could have only one TaskSolution for a given task.
11.1 Provide a correct implementation of methods myEquals and myHashCode for TaskSolution following the principles for correct behaviour of equals and hasCode. The TaskSolution should be compared based on the size of its picture in bytes.

JSON/XML/RSA

12. Allow a user to generate a new RSA key with a POST request on /users/:id/rsas. The private key of the user should be returned. Just the n and e. Each time a request is made a new key should be generated.
13. Store the generated RSA public key for the given user on the server.
14. Show all the rsa public keys for all users on /rsas
15. Allow a user to have more than one RSA key. It could be created with POST at /users/:id/rsas. They could all be shown with GET /users/:id/rsas. Specific RSA public keys could be shown with GET on /uses/:id/rsas/:rsa_id
14. When the user completes a task by submitting a POST request on /task_solutions the user should send an additional param with the form that is the digit 123 encrypted with one of the private keys of the user previously returned by the server. The server should check if at least one of the public keys of this user could decrypt the encrypted message and if the result is 123 than the TaskSolution could be created.
15. Implement a validation logic in the TaskSolution that would not allow for the creation of the TaskSolution and would mark the record as invalid if the passed encrypted value of 123 can not be decrypted successfully with at least one of the public keys of the user to gain 123. Add a

column for what is the encrypted value passed to the TaskSolution and with which keys was the value decrypted.
The validation logic should happen when calling record.valid?. Proper error message should be displayed in the form.
16. Allow for all controllers to accept requests that return the information as JSON.
17. Allow for all controllers to accept requests that return the information as XML.

Categorization

18. Add a Model Categorization
19. Each TaskSolution that was created with a valid RSA encrypted value of 123 with a user key, could be in a Category
20. When visiting a Category with a GET on /categories/:id show all the TaskSolution that are in this category
21. When creating a TaskSolution allow for choosing many categories in which the solution is.
22. Add a validation that a TaskSolution could not be in more than 4 categories

Изготвил:_____
/ ас. Кирил Митов /