Academic Devoir

Autor:

Suzana Santos

André Satoshi

Bruno Padilha

António Castro

Gustavo Coelho

Luciana Kayo

Susanna Rezende

Vinicius Rezende

Wallace Almeida

Data: 10/09/2011

Disciplina: MAC0332 Eng. Software

Professor: Marco Gerosa

Versão : 1.0

Universidade de São Paulo – IME – BCC

Visão geral

"Academic Devoir" é um sistema a ser desenvolvido para uso em plataforma Web, utilizando Java como linguagem básica de desenvolvimento, e adotando como tecnologias auxiliares os frameworks vRaptor e Hibernate.

O sofware tem como característica principal a gestão de tarefas acadêmicas das turmas de variadas disciplinas, baseadas em listas de exercícios, por sua vez constituídas de questões em diversas categorias, como dissertativa e "Verdadeiro ou Falso", e de correção automática.

O uso do sistema será destinado a professores e alunos. Os primeiros podem elaborar as atividades e visualizar a correção automática efetuada. Já os últimos poderão exibir as atividades disponíveis e solucioná-las.

Casos de Uso

De acordo com o tipo de usuário (aluno ou professor), existirão tarefas que devem ser atendidas pelo sistema. Podemos listar algumas das possiveis utilizações:

- Professor gerencia alunos
- Professor cadastra disciplina
- Professor cadastra turma
- Professor cadastra lista de exercício
- Professor cadastra questões
- Professor lista respostas de questão
- Professor pode reordenar questões de uma lista
- Professor determina o tipo de matricula
- Aluno se cadastra
- Aluno se matricula em uma turma de uma disciplina
- Aluno entra em uma disciplina e visualiza as listas
- Aluno resolve uma lista de exercícios

Plano de gerenciamento do projeto

Plano de iteração

Objetivos da iteração

A primeira iteração do projeto tem por objetivo a adaptação dos integrantes aos respectivos papéis e às práticas de desenvolvimento adotadas.

Após as 3 primeiras semanas de desenvolvimento, o grupo deverá apresentar um sistema funcional que realize o cadastro de alunos e professores e o login de usuários.
Não será exigido que o sistema permita o cadastro de disciplinas, turmas, questões ou listas de exercícios, porém, essas funcionalidades deverão estar trabalhando corretamente nos testes de unidade.

Ao longo da iteração, espera-se que todas as partes do desenvolvimento estejam bem documentadas, conforme a metodologia OpenUP e que seja feita uma boa modelagem das classes do sistema e um levantamento de requisitos detalhado.

Atribuições de trabalho
As tarefas foram atribuídas conforme o papel dos integrantes. Dependendo da dificuldade da tarefa e de sua relevância para o desenvolvimento do projeto, todos os integrantes devem contribuir.

Segue uma lista dos participantes e de seus respectivos itens de trabalho:

- André Satoshi Fujii de Siqueira (Desenvolvedor)
  Divisão das tarefas de implementação

  Implementação do CRUD de questões e listas de exercício.

- Antonio Junior (Documentrelacionadaador)
  Estudar documentação OpenUP e instruir os integrantes do grupo sobre como deve ser a documentação
  Acompanhar a documentação do projeto e dar as orientações necessárias.
  Reunir a documetação de todas as partes do desenvolvimento.
- Bruno Padilha (Desenvolvedor)
  Divisão das tarefas de implementação
  Implementação do cadastro de uma lista de exercícios.
- Gustavo Coelho (Analista de qualidade)
  Testes de Unidade para os CRUDS desenvolvidos.

Implementação do login no sistema

●———Luciana Kayo (Arquiteta)
Modelagem das classes do sistema.

Formatação das páginas com css.

●———Susanna Rezende (Arquiteta)
Durante essa iteração, a integrante não terá disponibilidade para executar as tarefas.

O tempo de ausência poderá ser compensado em outras iterações.

●———Suzana de S. Santos (Gerente)
Monitoramento das tarefas desenvolvidas e atribuição de novas tarefas

Modelagem das classes do sistema

Implementação do cadastro de uma resposta dada por um aluno no sistema.

●———Vinicius Rezende (Desenvolvedor)
Implementação do CRUD de aluno, professor, disciplina e turma.
●———Wallace Faveron de Almeida (Analista de requisitos)
Levantamento de requisitos
Implementação do Login no sistema.

Para maior detalhamento dos itens de trabalho, visite a página:
https://www.pivotaltracker.com/projects/355453#

Critérios de avaliação

Para avaliarmos o desenvolvimento, usaremos os testes de Unidade e a avaliação do cliente sobre o sistema.
Veja maiores detalhes sobre os testes na documentação do analista de qualidade.
Plano de projeto

Organização

O trabalho no projeto é dividido nas seguintes áreas:

Identificação	**Responsabilidades**	***Stakeholders***
Gerente do Projeto	Atribuições de caráter decisório e estratégico quanto aos rumos do projeto.	Suzana Santos
Analistas	Definir e aprovar os requisitos e especificações de negócio do sistema.	Wallace Almeida
Arquiteto do Projeto	Definir a arquitetura a ser	Luciana e Susanna Rezende

	utilizada no sistema.	
		Colaboradora: Suzana Santos
Documentação	Documentar	António Castro
		Colaboradores: Todos
Programadores	Implementar o sistema conforme as especificações.	André, Bruno e Vinícius
		Colaboradores: Todos
Testes	Padronizar os testes, homologar.	Gustavo .

Medições

A cada item de trabalho atribuímos pontos, que representam horas de dedicação (1 ponto equivale a uma hora).

Na página https://www.pivotaltracker.com/projects/355453#, temos os seguintes agrupamentos de itens de trabalho:
1 - Current (trabalho em desenvolvimento, concluído ou a ser desenvolvido em breve)
2 - Backlog (trabalho a ser desenvolvido em breve)
3 - Icebox (trabalho a ser desenvolvido sem data para início)

Objetivos

Criar um sistema de resolução e correção de listas de exercício na Web.

O desenvolvimento consistirá de 3 iterações (cada uma com 3 semanas de duração) durante as quais serão implementadas as histórias:

- Aluno se cadastra
- Login de aluno/professor
- Professor gerencia alunos (CRUD)
- Professor cadastra disciplina
- Professor cadastra turma
- Professor cadastra lista de exercício
- Professor cadastra questão de texto
- Professor cadastra questão de V ou F
- Professor cadastra questão de múltipla escolha

- Professor cadastra questão de submissão de arquivo
- Aluno se matricula em uma turma de uma disciplina
- Professor lista respostas de questão de texto que ainda não foram corrigidas e pode corrigi-la
- Professor pode reordenar questões de uma lista
- Aluno entra em uma disciplina e visualiza as listas a serem feitas e as já corrigidas
- Aluno resolve uma lista de exercícios
- Professor determina o tipo de matricula em uma turma (imediato ou com prazo definido)

Aguardamos mais informações do professor para completar a lista de histórias a serem implementadas.

Lista de riscos

Perguntar para o professor o que seriam os riscos do projeto.

Avaliação de status
Aguardando feedback do cliente.

Lista de Itens de trabalho
A lista dos itens de trabalho pode ser visualizada em:
https://www.pivotaltracker.com/projects/355453#

Testes

A maioria dos testes realizados até o momento são testes nas classes Controller, ou seja, que testam se os Controllers chamam os métodos apropriados dos DAOs e se o usuário é redirecionado para as páginas certas.
Há também testes que consistem em verificar se os diferentes objetos (Alunos, Professores, etc...) estão sendo corretamente inseridos em suas tabelas no banco de dados.
Para não afetar o banco de dados, todas os objetos que lidam com o banco de dados diratemente são mocks (imitações) dos objetos reais.

Caderno de arquitetura

No documento https://docs.google.com/document/d/1rVdzk-wq0-sQkVxT9n5PeDFcc7kIfKf34uIJjzSCdow/edit?hl=en_US encontra-se um modelo de classes do sistema e algumas observações sobre as decisões tomadas.

O diagrama correspondente ao modelo pode ser visualizado em doc/diagramas/diagrama de classes, no diretório raiz do projeto.

Diagrama de Classes

Obs.: Os nomes dados aos métodos e atributos nesse esboço são apenas descritivos. Na implementação, podemos ter nomes diferentes, seguindo os padrões do código.

Classe abstrata

Usuário {

----- *Atributos* -----

id

nome

login

senha

email

----- *Métodos* -----

fazerCadastro(),

fazerLogin()

listarTurmas()

}

Classes que implementam Usuário:

Aluno {

----- *Atributos* -----

lista de turmas (ids) [agregação n - n]

----- *Métodos* -----

entregarLista()

inscricaoNaTurma()

}

Professor {

----- *Atributos* -----

lista de turmas (ids) [agregação n - n]

----- *Métodos* -----
criarTurma()
cadastrarDsiciplina()
cadastrarNovaLsta()
cadastrarQuestao()
}

Turma {
----- *Atributos* -----
id
disciplina
tipoDeMatricula
períodoDeMatrícula
professor responsável
lista de alunos (ids)
lista de atividades (lista de lista de exercicíos) [agregação 1 - n]
----- *Métodos* -----
listarAlunos()
listarListasDeAtividades()}

Disciplina {
----- *Atributos* -----
id
nome
lista de turmas [composição 1 - n]
}

ListaDeExercicios {
----- *Atributos* -----
id
prazoDeEntrega
visibilidade
lista de questões (ids) [agregação n - n]
lista de listas de respostas (lista de objetos do tipo ListaDeRepostas) [composição 1 - n]
turma
}

ListaDeRespostas {
----- *Atributos* -----
aluno (id)
estado
lista de respostas. [composição 1 - n]
----- *Métodos* -----
notaDaLista()

[Sugestões de métodos: envia(), salva()]
}

Classe abstrata
Resposta {

----- *Atributos* -----
nota

corrigida?

comentário
}

Classes que implementam Resposta:
Multipla escolha {

----- *Atributos* -----
lista das alternativas escolhidas pelo aluno.

}

Submissão de arquivo: {

----- *Atributos* -----
arquivo enviado pelo aluno.}

Texto {

----- *Atributos* -----

String com a resposta dada pelo aluno.

}

Classe abstrata

Questão {
----- *Atributos* -----
id
valor
enunciado
tags predefinidas
tags definidas pelo usuário
}

Classes que inplementam Questões:

Multipla escolha {
----- *Atributos* -----
lista de alternativas
alternativa correta (+ de uma?) { V/F }
}

Submissão de arquivo: {

}

Texto {

----- *Atributos* -----

Resposta correta

}

BancoDeQuestões {

lista de questões

}