Instituto de Matemática e Estatística
Universidade de São Paulo, Brasil

Academic Devoir

André Satoshi
António Castro
Bruno Padilha
Gustavo Coelho
Luciana Kayo
Susanna Rezende
Suzana Santos
Vinicius Rezende
Wallace Almeida

Professor: Marco Gerosa
Disciplina: MAC0332 Engenharia de Software Versão: 2.0

23 de outubro de 2011

Sumário

1 Analista

1.1 Visão

"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.

1.2 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

1.3 Modelo de Casos de Uso

1.4 Glossário

G

Gerenciar: Entende-se que um usuário está apto a gerenciar uma dada estrutura do sistema quando possui permissões e interfaces para inclusão, edição, visualização e remoção de elementos dessa estrutura.

T

Turma: Conjunto composto por alunos (e monitor) que cursam uma disciplina determinada oferecida por um professor em um dado semestre. Podem existir mais de uma turma, com professores distintos, para uma mesma disciplina e semestre.

U

Usuário: Aquele que interage e utiliza o sistema. Nas especificações atuais, pode ser um professor ou um aluno.

2 Gerente

2.1 Plano da Iteração

Objetivos da iteração

Após as 2 primeiras iterações, que nos deram um esboço final de como será o sistema e que implementaram parte considerável de suas funcionalidades, o objetivo agora é a melhoria de tais funcionalidades, bem como a implementação do fator diferencial do sistema, que é o seu sub-sistema de cadastro e resolução de questões e listas. Os objetivos propostos para a atual fase de desenvolvimento são:

- Complementar e aprimorar as funcionalidades elaboradas durante as duas iterações anteriores;
- Realizar o cadastro de questões e listas, bem como sua resolução;
- Melhorar a navegabilidade do sistema para fins de depuração;
- Acrescentar mais informações a documentação do projeto.

Apesar de parte dos objetivos desta iteração, que visam fazer o sistema vir a ser completo, não se faz necessário, pelo menos temporariamente, um sistema visualmente navegável e agradável visualmente. Por enquanto, apenas o mínimo necessário de tais partes foi implementado. Isso será refinado apenas quando o sistema tido como completo do ponto de vista funcional.

Atribuições de trabalho

A cada iteração, os cargos e responsabilidades devem ser trocados. 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, para essa terceira iteração:

- André Satoshi Fujii de Siqueira (Arquiteto)
  - Modelagem das classes do sistema.
  - Refinamento do código do sistema.
  - Implementação do cadastro e resolução de questões e listas.

- Antonio Junior (Desenvolvedor)
  - Refinamento do código do sistema.
  - Implementação do cadastro e resolução de questões e listas.
- Gustavo Coelho (Gerente)
  - Monitoramento de algumas atividades realizadas pelos integrantes do grupo.
  - Levantamento da visão do projeto e dos requisitos cumpridos até o momento.
  - Implementação de connection pooling.
- Luciana Kayo (Analista de Requisitos)
  - Levantamento de requisitos do sistema.
  - Manutenção dos arquivos jsp.
- Susanna Rezende (Arquiteto)
  - Refinamento do código do sistema.
  - Modelagem das classes do sistema.
- Suzana de S. Santos (Documentador)
  - 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.
  - Refinamento do código do sistema.
- Vinicius Rezende (Analista de Qualidade)
  - Mapeamento ORM no hibernate.

  - Implementação de testes para fins de depuração e manutenção.

- Wallace Faveron de Almeida (Desenvolvedor)
  - Refinamento do código do sistema.
  - Criação de renderizadores para as diferentes questões.

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.

2.2 Plano do 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.	Gustavo
Analistas	Definir e aprovar os requisitos e especificações de negócio do sistema.	Luciana
Arquiteto do Projeto	Definir a arquitetura a ser utilizada no sistema.	André e Susanna
Documentação	Documentar	Suzana Colaboradores: Todos
Programadores	Implementar o sistema conforme as especificações.	Wallace e Antònio Colaboradores: Todos
Testes	Padronizar os testes, homologar.	Vinícius

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.

2.3 Lista de Riscos

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

2.4 Avaliação de status

Aguardando feedback do cliente.

2.5 Lista de Itens de Trabalhos

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

3 Arquiteto

3.1 Caderno de Arquitetura

Esboço do 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.

```
Esboço do diagrama de classes
Obs.: Os nomes dados aos métodos e atributos nesse esboço são apenas descritivos. Na imp

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
    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
}


Classe sugeriada pelo André
QuestaoDaLista{
    Long id;
    Questao questao;
    Integer valor;
    Algum outro atributo que eu não pensei
}
```

4 Desenvolvedor

4.1 Design

Design structure

Subsystems

Patterns

Overview
Structure
Behavior
Example

Requirement realizations

View of participants

5 Testador

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.

5.1 Test Case

Test Case ID - Test Case Name:
Descrição:
Pré-condições:
Pós-condições:
Dados requeridos:

5.2 Test Log

Produzido automaticamente.