My Last Project: Autorização, XML e Backups

Marco A. Almeida[1]

[1]Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros S/N – 40.170-110 – Salvador – BA – Brazil

marco062@dcc.ufba.br

Resumo. *O presente artigo apresenta algumas funcionalidades da ferramenta My Last Project utilizando recursos para aumentar a segurança e disponibilidade do sistema como criação de usuários e papéis no nível de banco de dados, exportação e importação de XML e criação de* backups online*.*

Abstract. *The following article introduces some features from the My Last Project tool using resources to increase the system's security and availability. User creation on the database level, XML export and import and online backups are some of what was developed in the latest version.*

1. Introdução

O objetivo da ferramenta My Last Project é promover a interação entre professores e estudantes, através de uma plataforma colaborativa para que os últimos atingam os seus interesses e aspirações na graduação, criando um projeto final de qualidade.

Em sua terceira versão, o software enfoca na segurança dos usuários ao associar papéis específicos à medida que visitantes se registram. Na seção 2 são descritos os detalhes de modelagem e implementação. Para oferecer maior integração com outras plataformas são fornecidos outros modelos de exportação, descritos em 4. Garantir a consistência do banco de dados independente das adversidades é necessário manter uma estratégia de *backup* que garanta que os registros criados por usuários não sejam perdidos. A seção 5 aborda essa estratégia. Por fim, 6 trata da etapa de gerenciamento para entregar a versão 3.0 do software My Last Project.

2. Criação de usuários

Até a versão 2.0 do software My Last Project, usuários utilizavam a aplicação com um mesmo usuário de banco de dados. Isso implica que qualquer usuário teria direito de realizar modificações em qualquer tabela no banco de dados do software. Entretanto, isso não reflete a realidade, já que apenas professores podem inscrever projetos, por exemplo. Esse problema de autorização no banco foi resolvido em uma estratégia de duas etapas: criação de papéis com suas respectivas permissões e criação de um usuário a nível de banco de dados e associação com o seu papel correspondente.

No projeto My Last Project existem três papéis diferentes: *guest* (visitante), capaz de se inscrever na plataforma; *student* (estudante), capaz de gerenciar referências e tarefas e o professor, que tem liberdade total do software com exceção da manipulação das notificações, onde é possível apenas buscar ou inserir registros. No PostgreSQL para definir papéis foi utilizado o seguinte código:

```
CREATE ROLE guests WITH LOGIN;
GRANT SELECT, INSERT ON users TO guests;
GRANT USAGE, SELECT ON SEQUENCE users_id_seq TO guests;

CREATE ROLE students;
GRANT ALL ON users TO students;
GRANT SELECT, INSERT ON notifications TO students;
GRANT ALL ON tasks, resources, quotations, notes, authors, authors_resources
  TO students;
GRANT SELECT ON professor_projects TO students;
GRANT USAGE, SELECT ON SEQUENCE users_id_seq, notifications_id_seq,
  tasks_id_seq, resources_id_seq, quotations_id_seq, notes_id_seq,
  authors_id_seq TO students;

CREATE ROLE professors;
GRANT ALL ON users TO professors;
GRANT SELECT, INSERT ON notifications TO professors;
GRANT ALL ON projects, tasks, resources, quotations, notes, authors,
  authors_resources TO professors;
GRANT USAGE, SELECT ON SEQUENCE users_id_seq, notifications_id_seq,
  projects_id_seq, tasks_id_seq, resources_id_seq, quotations_id_seq,
  notes_id_seq, authors_id_seq TO professors;
```

É importante notar que além de ser necessário dar permissões (*GRANT*) para acessar as tabelas, é imprescindível que sejam dadas permissões de seleção (*SELECT*) e uso (*USAGE*) para as sequências identificadoras onde a inserção de dados é possível. Isso porque nas tabelas que não representam relacionamentos é usado uma chave primária id que é sequencial.

Além disso, na transação de criação de usuários foi feita mais uma intervenção no código: a criação de um usuário no banco de dados, carregando uma formatação do nome e da senha, como visto no código a seguir.

```
CREATE USER #{db_username} WITH PASSWORD '#{encrypted_password}';
GRANT #{type.downcase.pluralize} TO #{db_username};
```

Os códigos entre #{} são interpolados com valores formatados pelo framework. Na primeira linha é criado um usuário com sua senha criptografada e nome de usuário formatado, que seria o email onde, em lugar da '@' é utilizada '_at_' e no lugar do '.' é utilizado o '_dot_'. Na segunda linha é dada a permissão baseada no tipo do usuário, ou seja, se um usuário criado é do tipo 'Student' então ele vai receber as permissões do grupo 'students'.

O fluxo de login da aplicação também foi modificado, já que agora é necessário que o usuário faça o login com o seu usuário no banco de dados. Qualquer usuário quando visita a página automaticamente se conecta ao banco com o perfil de *guests* e ao efetuar login o sistema irá conectá-lo com o seu respectivo usuário.

Dessa forma, os dados estão mais seguros, já que não são todos os usuários que podem manipular todas as tabelas no banco, evitando acidentes.

3. Auditoria

4. XML

A XML (*eXtensible Markup Language*) é uma linguagem cujo desenvolvimento tem por objetivo o transporte e armazenamento de dados. Similar a HTML, a XML não tem tags predefinidas, isso

é responsabilidade do desenvolvedor. Em My Last Project o XML foi utilizado para apresentar os dados dos recursos disponíveis e também para importação de recursos.

Os caminhos para apresentação de dados, sejam coleções ou membros de uma coleção podem ser vistos como .xml. Podemos ver a seguir a listagem de projetos por um professor no caminho '/projects.xml'

```
<?xml version="1.0" encoding="UTF-8"?>
<projects type="array">
  <project>
    <created-at type="datetime">2013-03-29T01:33:32Z</created-at>
    <due-at type="date">2013-07-13</due-at>
    <id type="integer">3</id>
    <professor-id type="integer">3</professor-id>
    <summary>Export everything</summary>
    <title>XML Project</title>
    <updated-at type="datetime">2013-03-29T01:33:32Z</updated-at>
  </project>
  <project>
    <created-at type="datetime">2013-03-29T01:33:01Z</created-at>
    <due-at type="date">2013-05-31</due-at>
    <id type="integer">2</id>
    <professor-id type="integer">3</professor-id>
    <summary>Save everything</summary>
    <title>Backup Project</title>
    <updated-at type="datetime">2013-03-29T01:33:01Z</updated-at>
  </project>
</projects>
```

De maneira análoga é a apresentação do membro da coleção. No caso, a única diferença é o membro que não estaria encapsulado na raiz do tipo *array*.

O recurso de importação está disponível apenas para o modelo 'recursos', que é visto como a parte mais importante da plataforma, já que alunos e professores estarão colaborando através dessa página. O XML do arquivo de importação é similar ao de exportação, com a diferença que, os objetos importados usam uma tag models_attributes para objetos associados, como citações, notas, e autores.

```
<?xml version="1.0" encoding="UTF-8"?>
<objects type="array">
  <object>
    <relevance type="integer">5</relevance>
    <title>Alice no País das Maravilhas</title>
    <type>Article</type>
    <authors_attributes>
      <author>
        <name>Marco Antonio</name>
      </author>
      <author>
        <name>Thiago Dias</name>
      </author>
    </authors_attributes>
```

```
  </object>
</objects>
```

Esse processo de importação é uma forma que facilita usuários a registrarem seus recursos na página sem a necessidade de ingressar na página de criação de recursos para cada recurso existente.

5. Backups

Manter os dados seguros na plataforma é uma das atividades mais importantes para o desenvolvimento. Isso significa que os dados são os bens mais valiosos em My Last Project. Perdê-los significa perder clientes. Dessa forma, na versão 3.0 foi implementado o backup online que utiliza arquivos de log para registrar quaisquer modificações no banco.

A estratégia utilizada foi a de PITR (*Point In Time Recovery*), cujo objetivo é ter um backup base do banco de dados (mesmo que inconsistente) e um conjunto de arquivos de log que registram cada modificação feita no banco. Quando algo inesperado acontece ao banco, póde-se usar o backup base e depois executar os arquivos de log, para recuperar os arquivos perdidos.

Inicialmente configura-se os arquivos de log para que seja feito o seu backup. São feitas três modificações no arquivo postgresql.conf listadas à seguior.

```
wal_level = archive # ou hot_standby
archive_mode = on
archive_command = \
  'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/archivedir/%f'
```

A última linha de configuração é um comando para onde os arquivos de log, também chamados de WAL (*Write Ahead Logs*) serão copiados. Depois de ter sido verificado como funcional é necessário configurar o arquivo base de backup.

Primeiramente executamos o comando `SELECT pg_start_backup('my_backup');`, onde 'my_backup' é um marcador para o seu backup. Neste momento se inicia o backup. A partir daí utilizamos um comando como o tar para fazer o backup convencional e armazenarmos todos os arquivos de dados. Ao terminar a compressão nos conectamos ao banco novamente e realizamos o comando `SELECT pg_start_backup('my_backup');` para confirmar que o backup já foi realizado.

Supor agora que algo inesperedo aconteceu com o banco e que é necessário a recuperação do banco de dados. Os passos a seguir apresentam o procedimento a ser realizado para a recuperação.

1. Parar a conexão com o banco;
2. Remover todos os arquivos na pasta de dados, verificando apenas se existem arquivos de log que não foram arquivados;
3. Recuperar os arquivos base de backup;
4. Remover os arquivos presentes na pasta pg_xlog/;
5. Caso existam arquivos de log não arquivados copiar manualmente para a pasta pg_xlog/;
6. Criar um arquivo de recuperação (recovery.conf) no diretório de dados. O arquivo de recuperação contém o seguinte comando: `restore_command = 'cp /mnt/server/archivedir/%f %p' ;`

7. Reiniciar o servidor de banco de dados. Ele iniciará em modo de recuperação e ao finalizar renomeará o arquivo de recuperação para recovery.done;
8. Inspecionar o conteúdo do banco. Caso algo esteja faltando, repetir o processo novamente;

A aplicação então garante ao usuário final a consistência e a segurança dos seus dados.

6. Gerenciamento

A ferramenta utilizada para gerenciar o projeto de software é o Github. A url para visualização das tarefas é https://github.com/marcoafilho/my-last-project/issues. A ferramenta não conta com funcionalidades avançadas, serve de suporte para o desenvolvimento.

A metodologia seguida é baseada nos modelos de gerenciamento de projetos open source. São definidos objetivos para cada versão e as tarefas para alcançar os objetivos são descritas. No total, o software está planejado para ser desenvolvido em seis versões e se encontra atualmente na segunda, a etapa de adição de recursos avançandos de banco de dados. Todas as versões são estimadas em uma semana, sendo encerradas sempre na quinta-feira. A seguir estão descritas as versões, contemplando o que já foi desenvolvido, o que está atrasado e o que virá a seguir.

Versão 1.0: Interface, Mapeamento e Modelagem

Era esperado dessa versão, a criação de protótipos para a interface do sistema e a modelagem do SGBD.

Como produtos dessa versão deveriam ser gerados um artigo e uma apresentação da primeira fase do sistema. Esta versão está atrasada, pois os produtos esperados pela finalização desta ainda estão em construção.

Versão 2.0: Views, Optimizataion, Stored Procedures, Triggers, Transactions

É esperado dessa versão que a aplicação já acesse os recursos do banco de dados. Funcionalides como *log in*, gerenciamento de projetos e de materiais devem estar funcionais.

Além disso, é esperado que a aplicação contenha recursos avançados de banco de dados, como a implementação de visões, índices, *stored procedures*, *triggers* e *transactions*.

O presente artigo, unido à apresentação e ao software parcialmente funcional são os produtos desta versão. Esta versão foi concluída com sucesso.

Versão 2.3

Além de serem implementadas técnicas de recuperação e backup, as funcionalidades básicas do software devem ser implementadas.

Versão 2.5

Etapa de implementação de práticas de segurança e autorização.

Versão 2.8

Os recursos serão exportados para outros formatos de dados, XML por exemplo.

Versão 3.0

Versão final do software. Consta da preparação de um relatório de testes realizados com a aplicação.

É esperada nessa versão que o software esteja completamente desenvolvido e que esteja seguro.

Os produtos dessa versão são: Um relatório de testes mais um artigo, um seminário e a publicação do software em um link externo.