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 dados dos usuários. Através de uma estratégia *backup online* vistos na seção 2 e de sua respectiva recuperação em 3 é garantida a capacidade do sistema de retornar a estados consistentes. Nem todos os usuários podem realizar as mesmas tarefas, por isso em 4 está descrita a definição de papéis e suas permissões.

É importante para os usuários visualizarem as modificações feitas em seus recursos, por isso a auditoria foi implementada nesse modelo. Mais detalhes na seção 5. Para oferecer maior integração com outras plataformas são fornecidos outros modelos de exportação, descritos em 6. Por fim, 7 trata da etapa de gerenciamento para entregar a versão 3.0 do software My Last Project.

2. Backup

Uma das atividades mais importantes para o desenvolvimento de qualquer plataforma comercial é garantir que os dados sejam segurados. É através deles que se pode extrair informações importantes para o contexto do cliente. Diante disso, na versão 3.0 da plataforma My Last Project foi implementada uma estratégia de *backup* para manter os consistente caso ocorra alguma falha no sistema.

A estratégia utilizada chama-se PiTR (do inglês *Point in Time Recovery*) e é definida por duas etapas: a criação de um *backup* base e o arquivamento de *logs*, também conhecidos como WAL (*Write Ahead Logs*). Quando uma transação é requisitada no banco ela é escrita no arquivo de *log* antes mesmo de ser executada. Apesar de ser um processo mais difícil de administrar existem vantagens significativas ao utilizar essa abordagem:

- O *backup* base não necessita estar consistente. As inconsistências são resolvidas com a execução dos arquivos de log;
- Útil para bancos de dados grandes, já que não é necessário fazer um backup completo frequentemente;
- É possível recuperar o banco de dados para qualquer ponto depois da tomada do *backup* base;
- Se mantermos o arquivamento dos logs em outra máquina podemos fazer uma replicação do sistema a qualquer momento.

Primeiramente, é importante garantir que o arquivamento dos arquivos de *log* estão sendo realizados corretamente. É necessário alterar algumas variáveis na configuração do arquivo postgresql.conf. Essas alterações são vistas na listagem à seguir.

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

A primeira linha define o nível de armazenameno no log. A segunda habilita o modo de arquivamento e a última linha explicíta o comando *shell* a ser realizado para arquivamento. Neste caso, os arquivos de log estão sendo copiados para uma pasta externa. É necessário reiniciar o banco de dados para carregar as novas configurações.

Ao terminar a configuração para arquivamento contínuo, é necessário realizar o *backup* base. No console do PostgreSQL é necessário informar o banco que está sendo feito um backup do banco. O comando `SELECT pg_start_backup('backup_timestamp');`, onde backup_timestamp é um marcador qualquer inicia o processo de *backup*. Neste momento pode-se usar qualquer ferramenta para fazer o *backup* convencional. Quando terminado o *backup* executa-se o comando `SELECT pg_stop_backup();` para que o banco sai do modo *backup*.

Desta maneira o processo de *backup* é finalizado.

3. Recuperação

Caso algo inesperado ocorra com o banco de dados como a remoção de um banco de dados por um usuário malicioso por exemplo, é possível recuperar o banco para qualquer estado depois do *backup* base, como já foi dito na seção 2. Os passos a seguir representam o fluxo de recuperação.

1. Parar a conexão com o banco, caso esteja ativa;
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.
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;

Para o arquivo de recuperação é imprescindível o seguinte comando: `restore_command = 'cp /mnt/server/archivedir/%f %p'`. Ele irá carregar os logs arquivados na pasta de dados.

4. Autorização

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 para manipular qualquer recurso que seja o pertença ou a um de seus estuadantes. 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 SELECT, UPDATE, DELETE 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 SELECT, UPDATE, DELETE 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.

5. Auditoria

O modelo central da plataforma são os recursos. Estudantes e professores visitam essa página para colaborarem entre si com artigos, livros, conferências, etc. Para que usuários acompanhem as modificações foi desenvolvida uma estratégia de auditoria.

Foi criada uma tabela chamada audits, que armazena: o identificador e o tipo do objeto modificado, o identificador do usuário, um array com as modificações comparadas com a última versão, a ação executada e o endereço remoto que realizou a ação.

A página de auditoria fornece então a união de todas as versões de um determinado recurso e de seus objetos dependentes (citações e notas) ordenadas pela versão mais recente.

6. 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.

7. 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 apenas de suporte para o desenvolvimento.

Para o lançamento da versão 3.0 foram feitas três pequenas iterações chamadas de versões 2.3, 2.5 e 2.8. As iterações estão esclarecidas à seguir.

Versão 2.3

Nesta versão foi finalizada a programação de recursos para a plataforma, como a recomendação de referências e a página de estudantes. Foram também implementadas as técnicas de *backup* e recuperação.

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
Implementação da auditoria e incorporação de todos os outros módulos. Como artefatos dessa iteração constam o presente artigo mais uma apresentação de slides.

A versão 3.0 não foi concluída por completo, pois ainda é necessário desenvolver a auditoria para modelos que seguem o relacionamento muitos para muitos. Independente disso, a aplicação possui um conjunto de funcionalidades importantes e já pode ser considerada usável.