UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE CIÊNCIAS APLICADAS A EDUCAÇÃO

DEPARTAMENTO DE CIÊNCIAS EXATAS

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

EasyQuiz: Um serviço para dar suporte à criação de exercícios com feedback automático

Autor: Digenaldo de Brito Rangel Neto

Orientadora: Dra. Ayla Débora Dantas de Souza Rebouças

RIO TINTO - PB

2015

Autor: Digenaldo de Brito Rangel Neto

EasyQuiz: Um serviço para dar suporte à criação de exercícios com feedback automático

Trabalho de conclusão de curso apresentado para obtenção do título de Bacharel à banca examinadora no Curso de Bacharelado em Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAE), Campus IV da Universidade Federal da Paraíba.

Orientadora: Dra. Ayla Débora Dantas de Souza Rebouças.

RIO TINTO - PB

2015

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca da UFPB

R196e Rangel Brito, Digenaldo de Brito.

EasyQuiz: um serviço para dar suporte à criação de exercícios com feedback automático. / Digenaldo de Brito Rangel Brito. – Rio Tinto: [s.n.], 2015. 48f. : il.-

Orientador (a): Prof. Msc. Ayla Débora Dantas de Souza Rebouças.

Monografia (Graduação) – UFPB/CCAE.

1. Software - desenvolvimento. 2. Informática na educação. 3. Sistemas de informação.

UFPB/BS-CCAE CDU: 004.4:37(043.2)

Autor: Digenaldo de Brito Rangel Neto

EasyQuiz: Um serviço para dar suporte à criação de exercícios com feedback automático

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal da Paraíba, Campus IV, como parte dos requisitos necessários para obtenção do grau de BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Assinatura do autor:_____

APROVADO POR:

_____

Orientadora: Dra. Ayla Débora Dantas de Souza Rebouças

Universidade Federal da Paraíba – Campus IV

_____

Prof. Msc. Yuri Malheiros

Universidade Federal da Paraíba – Campus IV

_____

Profa. Msc. Jarbele Cassia da Silva

Universidade Federal da Paraíba – Campus IV

RIO TINTO - PB

2015

“O sacrifício é o intervalo entre o seu objetivo e sua glória”

AGRADECIMENTOS

A Deus por ter me dado saúde e força nesta jornada.

A meus pais Ronaldo Rangel, Josineide Rangel, pelo amor, incentivo, apoio incondicional e por nunca me deixarem faltar nada. Vocês são da mais extrema importância para minha formação como pessoa, estudante e profissional.

A minha irmã Geoselis Rangel que me deu apoio nas horas difíceis, de desânimo e cansaço.

A minha orientadora Ayla Rebouças, que com seu vasto conhecimento me auxiliou e me orientou com muita paciência em todas as etapas deste trabalho.

Aos meus amigos com os quais compartilhei e adquiri conhecimentos e amizade durante boa parte da minha graduação dividindo moradia na cidade de Rio Tinto – PB: Kelson Victor, Danilo Formiga, Pablo Lima, Raphael Diniz, Rennan Felizardo, Lucas Cantarelli, José Paulo, Rafael Farias, entre outros companheiros.

A Aline Nery (in memoriam) por todo incentivo, apoio incondicional, dedicação e por todos os bons momentos em que vivi ao seu lado.

A Clara Nery por todo incentivo, apoio e orientação nas etapas da minha graduação, e a quem serei eternamente grato.

A todos os outros familiares e amigos, não mencionados, que de alguma forma tiveram sua participação ao longo desta caminhada.

RESUMO

Este trabalho propõe uma solução tecnológica para o problema da disponibilização e compartilhamento online de questões de diversos tipos e com feedback automático. Tal solução apresenta como principal características uso de *web services* para criação de questões e para o seu compartilhamento. A proposta é facilitar o cadastro e compartilhamento de questões cadastradas entre diferentes usuários que tenham acesso ao sistema, além de facilitar o uso de questões cadastradas por diferentes tipos de aplicativos e serviços.

Palavras chave: Web services, REST, SOAP, informática na educação.

ABSTRACT

This paper proposes a technological solution to the problem of making available and sharing online exercises of various types and which present automatic feedback. Such a solution has as main features the use of web services for creating questions and for sharing them. The proposal is to facilitate the registration and sharing exercises between different registered users who have access to the system, and also to aid in the process of making available these exercises to be used by different services and applications.

Keywords: Web services, REST, SOAP, informatics in education.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

TI: Tecnologia da Informação

AVA: Ambientes Virtuais de Aprendizagem

XML: *eXtensible Markup Language*

REST: *Representational State Transfer*

HTTP: *Hypertext Transfer Protocol*

SOA: *Service Oriented Architecture*

MTV: *Model Template View*

DRY : *Don't Repeat Yourself*

ORM: *Object Relacional Mapping*

API: *Application Programming Interface*

MER: Modelo de Entidade e Relacionamento

IDE: Ambiente Integrado de Desenvolvimento

UML: *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Segundo Bates (2003), a informática na educação tornou-se algo necessário para o avanço do ensino e aprendizagem entre alunos e professores. A tecnologia auxilia no desenvolvimento e elaboração de conteúdos para o progresso da educação em diversos setores. Com isso, as possibilidades de interações entre alunos e professores com a tecnologia da informação são diversas, facilitando assim a forma como as informações são trocadas.

Como parte das atividades dos professores está o processo de elaborar questões e disponibilização para seus alunos para que possam exercitar conteúdos vistos e para que o professor possa avaliar seu desempenho. Visto a necessidade do problema, surgiu a ideia da um serviço, onde se possam criar exercícios de perguntas de múltipla escolha e os alunos possam facilmente baixar tais exercícios e respondê-los por meio de dispositivos móveis. Segundo Soffa Marilice Mugnaini (2009), uma importante preocupação hoje em dia é a utilização da tecnologia da informação para expandir a forma como os professores buscam passar o conteúdo abordado em salas de aulas para os seus alunos. Também podemos levar em consideração algo bastante interessante, que é a troca de informação entre professores e alunos, para o compartilhamento de conteúdo e exercícios já criados, sem a necessidade de recriá-los. O aplicativo mostra-se inovador perante os demais com os quais foi comparado, por além de cadastrar questões ter a possibilidade de compartilhamento dos dados tanto para acesso interno pelos os usuários quanto ao acesso externo por outras aplicações tanto desktop quanto *mobile.* Isso tudo acontece devido a criação de um web service que foi um ponto chave para deixar um grande diferencial nesta aplicação e que possibilita o compartilhamento dos dados do sistema.

1.1 OBJETIVOS GERAIS

O presente trabalho tem por objetivo apresentar e fazer uma proposta de *web service* para suportar a criação e a prática de exercícios de múltipla escolha onde constam perguntas e respostas corretas (gabaritos), facilitando a elaboração de provas, questionários e qualquer tipo de exercício baseado em questões, que possam ser disponibilizados por sistemas web ou por aplicativos para dispositivos móveis em diferentes plataformas.

Ou seja, o trabalho propõe uma solução tecnológica para o problema de gerenciar e compartilhar questões entre usuários, de tal forma que contemple e facilite a geração e compartilhamento de questões já criadas relacionadas a determinados conteúdos.

1.2 OBJETIVOS ESPECÍFICOS

- Criar um serviço que permita o cadastro de questões e o acesso via aplicativo ou serviço web às questões criadas;
- Analisar as vantagens e desvantagens do serviço oferecido comparado a outros serviços semelhantes existentes;
- Fornecer uma API REST para disponibilizar os dados cadastrados.

1.3 METODOLOGIA

O primeiro passo na condução desse trabalho foi a análise de tecnologias que poderiam ser empregadas no desenvolvimento do serviço proposto. Posteriormente, foi feita uma observação sobre a necessidade de sistemas para apoiar a geração de exercícios com base em questões cadastradas por professores, o que levou ao projeto inicial do serviço.

O próximo passo da metodologia adotada foi o projeto do serviço a ser implementado, que deveria poder ser acessado por aplicativos para dispositivos móveis a partir dos quais o usuário responderia questões e também deveria suportar o acesso por um sistema para cadastro de questões usando os recursos tecnológicos que fossem mais adequados para a solução proposta.

O sistema foi sendo desenvolvido com todas as especificações definidas, a fim de garantir a efetividade da aplicação.

Por fim, o serviço foi avaliado considerando os objetivos propostos, visando especialmente observar sua utilidade e se poderia ser acessado por diferentes aplicativos/serviços. Além disso, o serviço foi comparado com alternativas semelhantes.

1.4 ESTRUTURA DO TRABALHO

O trabalho está estruturado da seguinte forma: no Capítulo 2 é apresentada a fundamentação teórica dos temas que serviram de estudo para a temática; já no Capítulo 3 é apresentada a arquitetura do serviço proposto, as suas funcionalidades e alguns exemplos de seu uso. Por fim, no Capítulo 4, são apresentadas as conclusões deste trabalho e propostas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentadas as fundamentações teóricas de todos os temas que foram estudados e que serviram como base para essa temática, entre esses assuntos estão: Informática na Educação, *SOAP*, *WEB SERVICE*, *REST*.

2.1 INFORMÁTICA NA EDUCAÇÃO

Com o avanço tecnológico, houve uma mudança no cotidiano da população mundial fazendo com que as formas de ensinar diversos conteúdos tivesse de mudar.

Novas tecnologias vêm desafiando o processo de ensino e aprendizagem e a maneira como esse processo é gerido. A Tecnologia da Informação (TI) vem para apoiar esse processo, permitindo agilidade de comunicação e o acesso rápido a uma grande quantidade de informações. É importante destacar que o processo de aquisição de conhecimento passou a ser feito de diversas formas: a partir das casas dos usuários, de seu trabalho ou de ambientes de ensino.

Há uma grande variedade de formas de explorar a TI no processo de ensino aprendizagem, como abordagens multimídia para educação, técnicas de aprendizagem colaborativas e o ensino a distancia.

Segundo Marta Campos (2003), a integração da tecnologia da informação no ensino hoje é uma questão crucial na garantia da qualidade do sistema educacional. Uma das razões importantes para integração TI e ensino é que se prevê que todas as áreas de estudos no futuro se tornarão dependentes dela. Outra razão é que se tem observando que a TI deverá ser utilizada no ensino ajudando a melhorar sua qualidade e a tornar o processo de ensino-aprendizagem eficiente e eficaz.

Uma das formas de se explorar a TI na educação é por meio dos ambientes virtuais de aprendizagem (AVAs). Segundo Dillenbourg (2002), um ambiente virtual de aprendizagem (AVA) é um sistema computacional web criado para gerenciamento de conteúdo com foco no ensino e aprendizagem à distância utilizando a Internet. Os AVAs não se restringem apenas a troca de informação de dados, mas são também uma forma interativa e organizada para o gerenciamento de conteúdo.

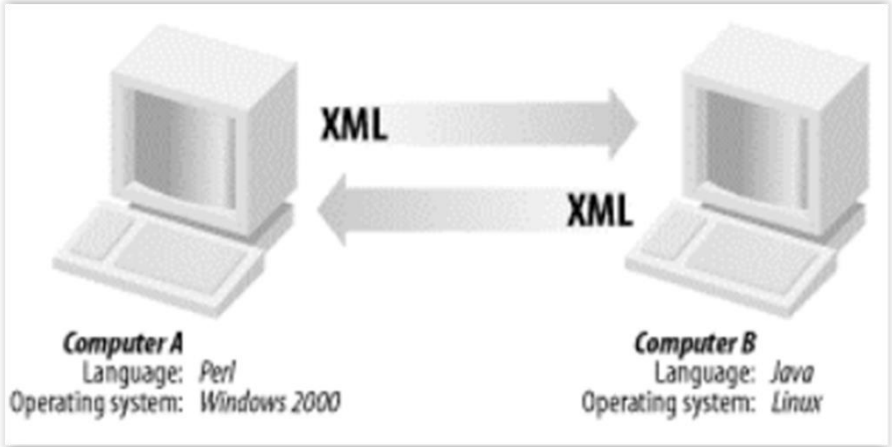
Um AVA é diferente de um site web educacional, embora alguns autores da atualidade tenham usado o termo para se referir também a sites com paginas web estáticas. Além disso, Segundo Dillenbourg (2002), um AVA não precisa obrigatoriamente explorar tecnologias de realidade virtual 3D ou outros recursos avançados, podendo apresentar interfaces bem menos sofisticadas, como as baseadas em texto puro.

Um exemplo de AVA muito conhecido é o Moodle. Segundo Moodle Pty (2001), o Moodle é classificado como um sistema de gerenciamento de aprendizagem de código aberto, distribuído sob GPL (General Public License). Baseado em princípios pedagógicos, Moodle é uma plataforma de ensino a distância, que nos dias atuais tornou-se uma forma de ensino extra em diversas instituições de ensino. Com recursos de administração personalizáveis, é usado para cursos privados e públicos com o intuito de alcançar metas de aprendizagem. O Moodle foi criado por Martin Dougiamas, para ajudar a professores na criação de cursos online com foco na interação e colaboração de alunos para uma evolução contínua de aprendizagem. Os AVAs tem suporte à criação de exercícios, todos integrados à plataforma, mas seria de grande relevância ter uma solução independente, para explorar não só o ensino a distância, mas também o ensino presencial e desenvolvida também de forma a se integrar com AVAs. Uma forma de alcançar esseobjetivo é a utilização de tecnologias independentes de plataforma, como por exemplo, *web services*, para agilizar a integração das funções com outros serviços.

2.2 WEB SERVICE

Segundo Cerami (2002), um *web service* é qualquer serviço que esteja disponível na Internet, que utiliza um sistema de troca de mensagens padronizadas por XML (*eXtensible Markup Language*) ou por qualquer padrão de comunicação adequado, e que não está vinculado a qualquer sistema operacional ou linguagem de programação. A Figura 1 mostra um funcionamento básico de um *web service.*

Figura 1: Funcionamento básico de um *web service.*

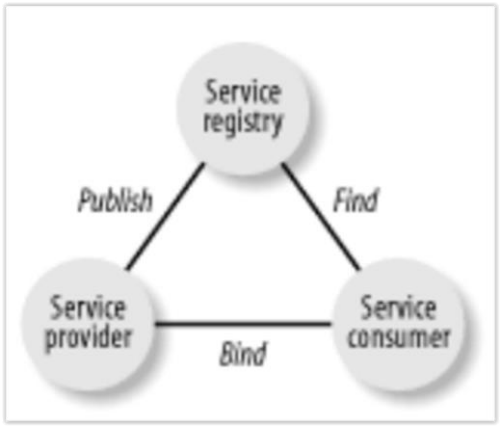


(Fonte: Figura retirada de Cerami, 2002)

Por causa da abstração fornecida pelas interfaces baseadas em padrões, não importa se os serviços de aplicativos são escritos em Java e o navegador escrito em C++, ou um serviço de aplicativos sendo executado no Unix, enquanto o navegador está rodando no Windows. *Web services* permitem a interoperabilidade entre plataformas de maneira que o tipo de plataforma seja irrelevante (Tidwell et al, 2001).

Existem diversas maneiras de observar a arquitetura de um web service. Uma delas seria a consistência em analisar cada função que envolve um serviço. Outra é analisar a pilha existente de protocolos com que se pode interagir na comunicação deste serviço. A Figura 2 mostra um modelo arquitetural de um *web service*.

Figura 2: Arquitetura de um web service.



(Fonte: Figura retirada de Doug Tidwell et al, 2001)

Como visto na figura anterior, observa-se um exemplo de um ciclo de operações que são realizadas durante o funcionamento de um *web service.* O *service provider* publica uma descrição dos serviços ofertados através do *service registry*. O *service costumer* procura o *service register* para encontrar um serviço que atenda as suas necessidades. Vale salientar que o *service provider* poder ser representado por uma pessoa ou programa de computador.

Por fim, a ligação entre as entidades refere-se ao *service consumer,* que se utiliza do serviço oferecido por um *service provider,* requisitando procedimentos no momento em que se deseja realizar certa operação.

2.3 SOA

Segundo Rotem-Gal-Oz (2012), SOA (Service Oriented Architecture) é uma arquitetura para sistemas baseados em interações de componentes, autônomos e de baixo acoplamento chamados de serviços. Cada serviço expõe processos e práticas por meio de contratos, que caracterizam mensagens que podem ser enviadas a endereços denominados servidores. O comportamento de um serviço é regido por políticas que são externas ao próprio serviço. Os contratos e as mensagens são usados por componentes externos chamados consumidores de serviços.

Segundo JOSUTTIS, M. N (2007), SOA tem o propósito de tratar os requisitos de baixo acoplamento, desenvolvimento baseado em padrões, computação distribuída independente de protocolo, mapeamento dos sistemas de informação da organização para todos os seus fluxos de processos de negócios, integração de aplicações, gerência de transações, políticas de segurança e coexistência de sistemas em múltiplas plataformas e também sistemas legados (Papazoglou et al, 2007).

A partir do modelo de arquitetura da SOA, tem-se uma visão de um paradigma para elaboração de um sistema distribuído de maneira que todas as suas funções estejam engajadas em um conjunto de serviços, de forma que possam ser utilizadas por outros serviços ou aplicações.

2.4 REST

Segundo Fielding (2000), REST (Representational State Transfer) é uma abstração da arquitetura da *World Wide Web*. De forma precisa, REST é um estilo arquitetônico que consiste em um conjunto coordenado de restrições aplicadas a componentes, conectores e

elementos de dados, dentro de um sistema computacional distribuído. REST ignora os detalhes da implementação do componente e sintaxe do protocolo, a fim de concentrar-se em suas funcionalidades.

Segundo Ferreira C (2006), SOAP é uma alternativa para sistemas com padrões rígidos e ambientes de alta complexidade. Diversas ferramentas corporativas tiram vantagem do padrão e possibilitam filtragem, enfileiramento, classificação e redirecionamento das mensagens trocadas entre os sistemas. Praticamente todas as plataformas e linguagens modernas suportam REST e a solução final é muito mais simples do que o equivalente em SOAP.

Além disso, integrações com alto volume de requisições são inviáveis em SOAP. REST é capaz de atender volume e complexidade sem dificuldades, exigindo apenas um mínimo de experiência do desenvolvedor para estabelecer e reforçar os padrões adequados.

Ao contrário do modelo distribuído (Fielding, citado por CHIN e CHANSON, 1991), em que todos os dados são encapsulados pelos componentes de processamento, o estado dos dados de uma arquitetura computacional é um aspecto fundamental do REST. A justificativa para este aspecto pode ser vista nos elementos da computação distribuída onde os dados são armazenados em um serviço que será utilizado na maioria dos casos por outras pessoas.

Isso é diferente de muitos outros paradigmas de processamento distribuídos (Fielding, citado por ANDREWS, 1991 e FUGGETTA et al, 1998), onde é possível e mais eficiente, mover o "agente de processamento" (por exemplo, código de programação móvel, processos armazenados, expressões de busca, etc.) para os dados, em vez de deslocá-los para o processador.

Segundo WEBBER, S. I (2010), REST possui recursos, que podem ser utilizados por meio de um identificador global, fazendo com que os componentes físicos de interação como, por exemplo, cliente-servidor se comuniquem através de um protocolo padrão HTTP (Hypertext Transfer Protocol) trocando representações de arquivos ou elementos que são recebidos ou enviados.O protocolo HTTP possui sete operações: os métodos GET, POST, PUT, DELETE, HEAD, OPTIONS e TRACE. Duas delas (GET e POST) já são bem conhecidas e utilizadas: GET quando clicamos em links ou digitamos o endereço no navegador, e POST quando preenchemos formulários de cadastro. Cada método tem uma semântica diferente e juntando o método à URI (Uniform Resource Identifiers) que representa todas as ações de um sistema. As semânticas principais são:

GET - recupera informações sobre o recurso identificado pela URI. Ex: listar produtos, visualizar o produto 45. Uma requisição GET não deve modificar nenhum recurso do seu

sistema, ou seja, não deve ter nenhum efeito colateral, você apenas recupera informações do sistema.

POST - adiciona informações usando o recurso da URI passada. Ex: adicionar um produto. Pode adicionar informações a um recurso ou criar um novo recurso.

PUT - adiciona (ou modifica) um recurso na URI passada. Ex: atualizar um produto. A diferença fundamental entre um PUT e um POST é que no POST a URI significa o lugar que vai tratar a informação, e no PUT significa o lugar em que a informação será armazenada.

DELETE - remove o recurso representado pela URI passada. Ex: remover um produto.

HEAD, OPTIONS e TRACE - recuperam metadados da URI passada. Respectivamente o Header, quais métodos são possíveis e informações de debug.

2.5 PYTHON

Segundo Python.org (1991), Python é uma linguagem de programação de alto nível, usada por muitos programadores de todo mundo. Esta linguagem tem a capacidade de suportar múltiplos paradigmas de programação, incluindo programação imperativa, funcional, orientada a objetos, e estilos processuais. A linguagem possui um sistema de tipagem dinâmica, que faz com que na declaração de variáveis não se exijam tipos de dados. Os tipos a utilizar são atribuídos de maneira dinâmica para cada variável, e podem ser alterados durante a compilação ou execução do programa.

Segundo Moraes, M. B. D (2006), uma das grandes vantagens de criar aplicativos utilizando Python é o fato dos seus comandos serem executados de forma imediata, sem a obrigação de compilar ou vinculação com outras ferramentas. Além disso, para utilizar um script desenvolvido no Linux, Windows ou Mac, seria necessário somente copiar o programa para a plataforma destino.

Python foca no código legível ao invés da maneira de como ele é interpretado pela máquina. O empenho do desenvolvedor tem maior enfoque que a parte computacional, buscando diminuir o primeiro, mesmo que seja necessário aumentar o segundo. A linguagem possibilita a combinação de sintaxe com módulos e *frameworks* criados por terceiros ou com recursos de sua biblioteca padrão.

Figura 3: exemplo de código em Python

```
    """ Serves a file statically """
    if mime_type is None:
        mime_type = mimetypes.guess_type(filename)[0]
    web.header('Content-Type', '%s' % mime_type)
    stat = os.stat(filename)
    web.header('Content-Length', '%s' % stat.st_size)
    web.header('Last-Modified', '%s' %
                web.http.lastmodified(
                    datetime.datetime.fromtimestamp(stat.st_mtime)
                )
    )
    return wsgiref.util.FileWrapper(open(filename, 'rb'), 16384)


def list_files(directory):
    """ List files in a directory. Returns a list """
    try:
        dirlist = [(filename,
                    os.path.isdir(os.path.join(directory, filename)))
                    for filename in os.listdir(directory)]
    except:
        #this directory is empty
        dirlist = []
                                                    49,1          96%
```

2.6 DJANGO

Segundo o seu site oficial[1], Django é um *framework* para desenvolvimento web, codificado na linguagem Python, que utiliza o padrão de projeto *Model Template View* (MTV). Sua criação se deu a partir de um sistema para gerenciar um site jornalístico, que por fim tornou-se um projeto de código aberto que recebe contribuições de empresas e desenvolvedores de todo o mundo. Django opera sobre o principio *Don't Repeat Yourself* (DRY), que permite ao programador reaproveitar todo o seu código que já foi feito, evitando repetição.

Esse *framework* tem a existência de um padrão em três camadas MTV. Na primeira parte são os Models, onde as classes são criadas e objetos criados para depois serem armazenados no banco. Na segunda camada é a de Template, onde fica toda a parte visual, ou seja, a parte do html do projeto onde haverá a interatividade com o usuário. A última camada é a View, onde poderão ser criadas instancias de classes, classes da lógica da aplicação e métodos serão montados para depois serem chamados nas páginas que montam as URLs. Desta forma, cada método será implementado para uma página especifica. Os principais recursos do Django são:

- Mapeamento Objeto-Relacional (ORM): esse mapeamento significa que a partir dos atributos determinados em cada classe Python, o banco de dados poderá ser modelado sem a necessidade de usar o banco de forma direta. Os campos e tabelas são criados e manuseados pelas classes.
- Interface Administrativa: O Django possui por padrão uma interface administrativa, montada a partir dos objetos especificados nas classes facilitando a maneira como a aplicação possa permitir os acessos restritos a usuários.
- URLs: no Django há a possibilidade de personalizar a maneira de como você deseja que as urls de acesso as funcionalidades sejam montadas. Com a utilização de expressões regulares para o gerenciamento de números e letras deixando todo o esquema de urls bastante simples.
- Formulários: formulários podem ser gerados a partir dos objetos do banco de dados.
- Internacionalização: suporte a diversos idiomas facilmente configuráveis para toda aplicação.

Figura 4: exemplo de tela do Django admin

O *framework* Django na criação de uma aplicação web simplifica e agiliza o seu desenvolvimento. A manutenibilidade é um fator crucial para quem deseja elaborar e desenvolver uma aplicação em curto prazo, e sua ideia de não repetir códigos facilita sua manutenção fazendo com que blocos de códigos sejam reutilizados por outras partes da aplicação.

[1] https://www.djangoproject.com/

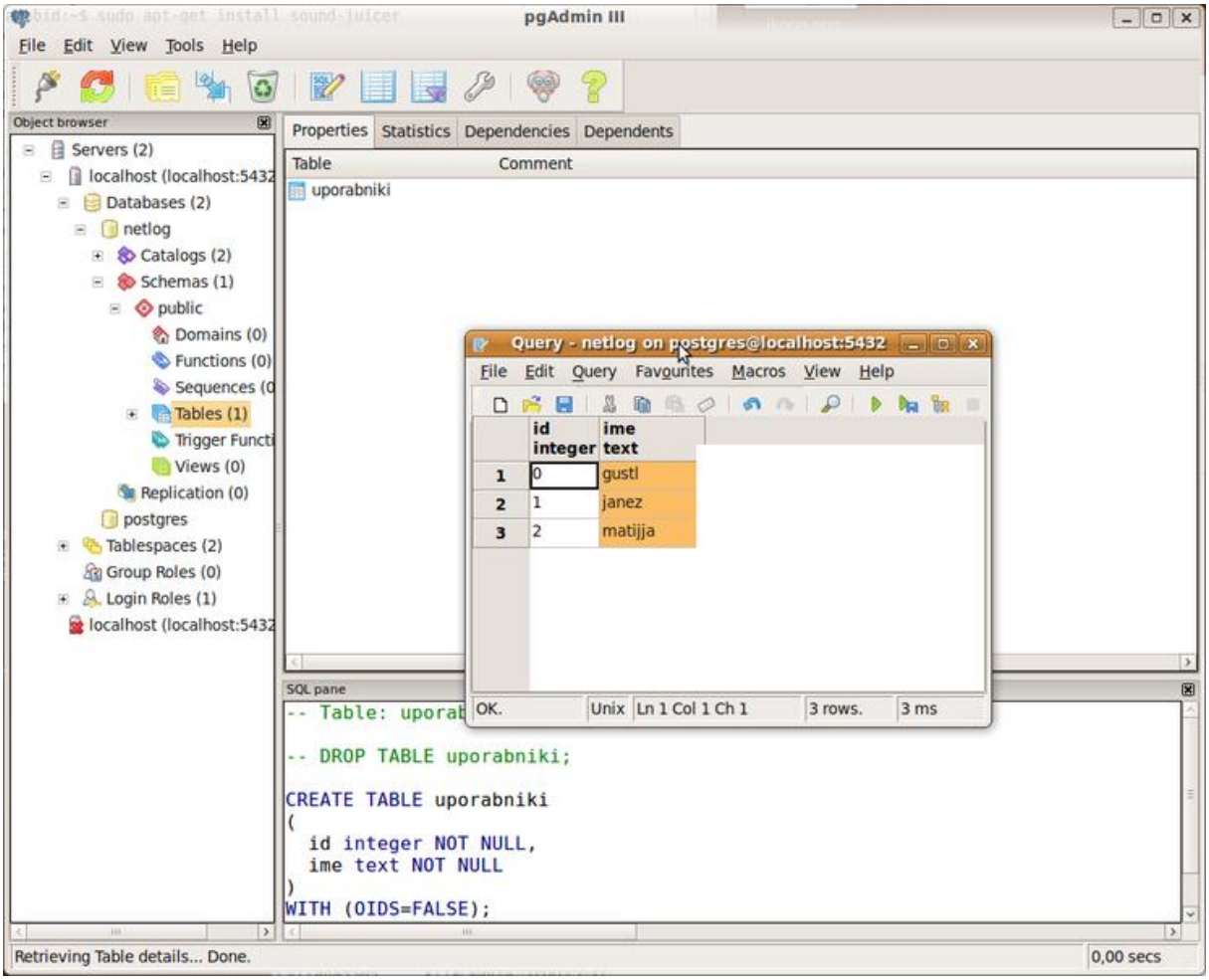
2.7 POSTGRESQL

Segundo o site oficial do Postgres[2], o Postgresql é um banco de dados objeto-relacional de código aberto extremamente robusto, estável e está em desenvolvimento desde 1997, além de ser totalmente flexível e rico em recursos. É considerado objeto-relacional por implementar, além das características de um SGBD (Sistema Gerenciador de Banco de Dados) relacional, algumas características de bancos orientados a objetos, como herança e tipos personalizados. O Postgresql tem como principais características:

- *Logging* de transações;
- *Commit/Rollback/Checkpoints*;
- *Triggers/Stored Procedures*;
- *Constraints/Foreign Keys*;
- Recuperação automática após crash de sistema;
- MVCC (Controle de Concorrência de Multi-versão). Neste mecanismo, processos de leitura não bloqueiam processos de escrita e vice-versa, reduzindo drasticamente a contenção entre transações concorrentes e paralisação parcial ou completa;
- Múltiplos tipos de índice: suporta vários tipos de índices, permitindo a escolha do índice com mais eficiência para cada aplicação.
- Índices em cluster: cada tabela pode suportar um índice em *cluster*. Este índice classifica fisicamente os dados, na mesma sequencia como especificada pelo índice. Um índice de cluster permite a maior velocidade possível na recuperação de dados melhorando o desempenho geral do banco de dados;
- Tamanho ilimitado de registro: não impõe limites no tamanho de armazenamento dos tipos de dados.

[2] http://www.postgresql.org/

Figura 5: Interface gráfica do Postgres no Pgadmin

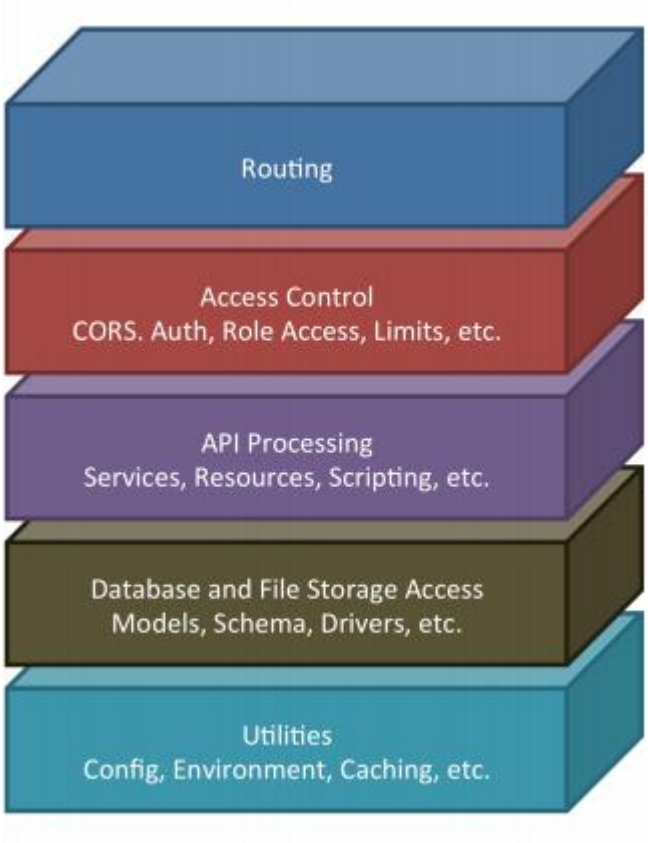


Um dos principais motivos para a utilização do Postgres é por ele oferecer baixo custo total de propriedade e ao mesmo tempo fornecer alto desempenho, confiabilidade e escalabilidade.

2.8 DREAMFACTORY

Segundo o site oficial do Dreamfactory[3], o Dreamfacotry é uma API (Application Programming Interface) de código aberto que fornece serviços REST para a construção de aplicações computacionais. Trata-se de uma aplicação que é executada em um servidor web onde é possível configurar qualquer tipo de banco de dados e fornecer diretamente todo banco mapeado com rotas especificas através de uma API REST. O Dreamfactory é dividido na lógica operacional como mostra a Figura 6.

Figura 6: Funcionamento operacional do Dreamfactory



(Fonte: Figura retirada de dreamfactory.org, 2015)

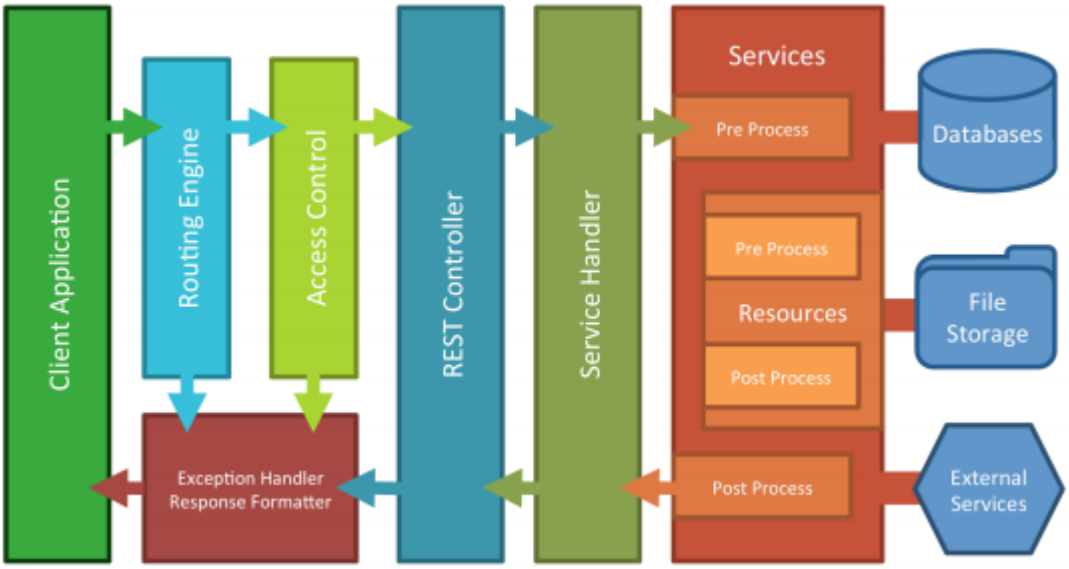
O Dreamfactory apresenta como características principais:

- Autenticação de usuários: possui por configuração, módulo para autenticação de usuários via REST.
- Banco de dados SQL: integração com banco de dados SQL (MySQL, SQL Server, PostgresSQL, Oracle, IBM DB2, SQLite.

- Banco de dados *NoSQL*: integração com banco de dados *NoSQL (AWS DynamoDB, Azure Tables, CounchDB, MongoDB, SalesforceDB*.
- Armazenamento de arquivos: configuração para armazenamento de arquivos.
- Envio de e-mails para usuários já configurados.
- Fornece a api REST do banco de dados configurado remotamente.

A Figura 7 apresenta de forma simplificada o funcionamento do Dreamfactory:

Figura 7: Funcionamento do Dreamfactory



(Fonte: Figura retirada de dreamfactory.org, 2015)

Por ser simples de usar, fácil configuração, personalizável, dinamicamente expansível, código aberto e utilizado por mais de duzentos mil programadores por todo o mundo, o Dreamfactory foi uma ferramenta crucial para a criação do serviço da aplicação proposta neste trabalho.

[3] https://www.dreamfactory.com/

3 FERRAMENTAS RELACIONADAS

Para poder avaliar o serviço proposto nesse trabalho, foi realizada uma pesquisa bibliográfica a procura de possíveis ferramentas que tivessem funções que correspondessem ao serviço proposto. Isto foi realizado com a finalidade de colher, e consequentemente justificar, a viabilidade da proposta e a implementação de um novo aplicativo, com funções diferenciadas e que se adequem às necessidades da solução proposta.

Na pesquisa, algumas ferramentas semelhantes foram encontradas e são descritas antes de exibir e comparar algumas de suas características acentuadas no contexto deste trabalho, sendo elas:

- QuizCreator(QuizCreator, 2015) : é uma ferramenta que permite aos educadores a criação de questionário ou provas com diversos formatos, podendo formular suas perguntas com imagens, flash, áudio e vídeo além de suporte a símbolos matemáticos. Também é permitido o acompanhamento dos resultados assim como a criação de relatórios.
- Educandus(Educandus, 2015): é composto por um banco de questões, possui um gerador de provas, além de ser uma ferramenta de monitoramento de aprendizagem, onde é possível inserir questões, elaborar provas e simulados, selecionando questões específicas ou de forma randômica, podendo monitorar a aprendizagem do aluno por meio de gráficos e tabelas. O sistema ainda possui outras funcionalidades como área administrativa, onde é possível realizar cadastro de quaisquer perfis de usuário. Há ainda um gerenciador de conteúdo considerado peça fundamental na criação da programação de cursos a distância, um gerenciador de aulas, onde é possível criar páginas com conteúdos multimídia, textos e exercícios personalizados.
- iTeste(MICHAL TOMLEIN, 2015): é um sistema para produzir testes. O sistema é composto por dois programas: o iTesteServer, que é composto por um banco de dados de questões e um servidor de exames e o iTesteCliente, que é utilizado pelos alunos para resolver os testes. O sistema torna fáceis as atividades de criar e organizar o banco de questões, configurar o servidor e a impressora, assim como conectar um computador cliente para cada aluno, que gera um teste de acordo com as configurações do seu servidor. O sistema é multiplataforma e oferece várias opções de língua, dentre elas o português.

- Educador.net(Educador.net, 2015): software gratuito desenvolvido para auxiliar o professor nas suas tarefas diárias, otimizando o trabalho de controle das suas atividades e agilizando-as. Um dos destaques do programa é um banco de questões e provas, que permitem arquivar as questões por tema e nível de dificuldade, facilitando na elaboração de provas. Mas ainda podem-se destacar características como cálculo da média da turma e gráficos de avaliação.
- SGP(STARLINE, 2015) – Sistema de Gestão de Provas: é um sistema onde a criação das provas é realizada a partir do banco de questões, onde é possível filtrá-las por nível ou tipo de conteúdo. Uma prova pode ser montada com questões escolhidas de forma randômica, além de garantir um formato padrão para as provas. Quando se faz a solicitação de uma prova ao sistema, o mesmo pergunta qual o formato das provas, ou seja, se serão realizadas na versão eletrônica ou impressa. O sistema ainda oferece uma opção de relatórios e gráficos que auxiliam na verificação do desempenho dos alunos entre várias outras funcionalidades que ajudam o professor no dia-a-dia.

Depois de uma análise das ferramentas citadas, foi possível fazer uma descrição de suas características principais de maneira comparativa, e que está mostrada na Tabela 1.

A partir da pesquisa efetuada, viu-se que todos esses aplicativos mostraram como características básicas a criação, edição e alteração de dados para geração de provas, assim como correção das mesmas que é realizada pelo próprio aplicativo. Algumas das ferramentas apresentadas possuem perfil direcionado para o ensino a distância, como é o caso do QuizCreator, e outros. Em algumas situações, por se tratar de aplicativo proprietário, não foi possível o acesso total a todas as funcionalidades, dificultando a argumentação quanto à sua utilização.

Contudo, os aplicativos apresentados na Tabela 1 têm sua eficiência voltada ao cadastro, criação e correção de questões. A grande desvantagem notada na utilização desses aplicativos foi o fato da obrigatoriedade da instalação da ferramenta em uma máquina específica, pondo limite em sua instalação, algumas com custos de manutenção muito elevado e muitas delas tendo a necessidade de aplicação em laboratórios para a realização das provas dificultando a mobilidade para os professores e alunos.

	QuizCreator	Educandus	iTeste	Educador.net	SGP
Aplicação Web		X		x	
Aplicação Desktop	x	X	x	x	x
Correção por computador	x	X	x	x	x
Avaliação por Computador	x		x	x	x
Armazenamento em banco de dados	x	X	x	x	x
Armazenamento em Arquivo	x				
Licença Proprietária	x	X			x
Licença Livre			x	x	

Tabela 1: Características dos aplicativos utilizados para comparativo

Com relação aos dados levantados na pesquisa vale destacar as características do serviço proposto e que se espera que possa ser usado por diferentes aplicativos em diferentes plataformas. O resumo das características analisadas nas ferramentas pesquisadas está destacado na Tabela 1. O serviço idealizado fornece aos usuários a possibilidade do armazenamento de questões em categorias tendo como principal destaque a disponibilização dos dados através de um serviço web, que será capaz de ser acessado por outros aplicativos fazendo com que haja um desacoplamento dos dados e o principal diferencial é deixar essas informações de forma compartilhada e multiplataforma sendo acessado por qualquer tipo de aplicativo.

Todos os softwares da pesquisa que se encontram na Tabela 1 possuem o cadastro de questões e armazenamento dos dados, mas nenhum dos aplicativos contempla, respectivamente, a característica de disponibilizar os dados para outros aplicativos. Baseado nessa análise, pode-se afirmar que é válido o desenvolvimento e implementação de um serviço que possa ser independente de plataforma e permitir a comunicação com diferentes tipos de dispositivo.

4 DESENVOLVIMENTO DO EASY QUIZ

Este capítulo detalha o desenvolvimento do EasyQuiz e é dividido em quatro seções. A Seção 4.1 trata do modelo de desenvolvimento utilizado e descreve as etapas realizadas: a análise de domínio, a elicitação de requisitos e a definição de casos de uso. A Seção 4.2 apresenta uma abstração da arquitetura concebida para a solução proposta e descreve os seus componentes. A Seção 4.3 descreve o diagrama do banco de dados e, por fim, a Seção 4.4 apresenta um protótipo utilizável da solução proposta, incluindo descrição de codificação e telas de uso.

4.1 ENGENHARIA DE REQUISITOS

"A Engenharia de Requisitos corresponde à atividade de entendimento das necessidades do usuário no contexto do problema a ser resolvido, bem como das limitações impostas na solução" [Sommerville 2003 apud Neto 2008]. A Engenharia de Requisitos nos oferece maneiras bem decididas para que a análise do problema seja feita de forma que o desenvolvedor com a ideia da aplicação possa auxiliá-lo durante todo processo de desenvolvimento da mesma. Três processos da Engenharia de Requisitos são aplicados na criação deste trabalho: a análise de domínio, a elicitação de requisitos e a concepção de casos de uso.

4.1.1 ANÁLISE DE DOMÍNIO

Após a verificação de uma revisão bibliográfica foi possível observar e relatar funcionalidades particulares de cada sistema existente relacionado ao Easy Quiz, assim como explanado na Tabela 1, no Capítulo 2. A revisão bibliográfica teve papel essencial para se notar que todos os sistemas avaliados possuíam características iguais quando foi observado o meio utilizado para gerir as questões de todos os sistemas. Fazendo uso dessa observação, foi possível propor uma abordagem diferenciada das outras ferramentas, visto que além do cadastro de questões, uma funcionalidade disponibilizada por todos os sistemas citados, há a possibilidade de disponibilizar as questões cadastradas através da *web service* para que outros aplicativos tenham acesso a elas. Além disso, existe a possibilidade de acessar o aplicativo

por dispositivo móvel por ele possuir uma interface responsiva, ou seja, a visualização da aplicação é adaptável a diversos dispositivos.

4.1.2 ELICITAÇÃO DOS REQUISITOS

A partir da análise de domínio do sistema foram concretizados alguns processos para se determinar os requisitos do sistema, assim como a fixação dos casos de uso, sendo abordados logo a seguir.

4.1.3 REQUISITOS DO SISTEMA

As definições dos requisitos de um sistema não é uma tarefa tão simples de ser realizada, pois é através da especificação dos requisitos que serão elaboradas as funções que o sistema deverá possuir em determinadas ocasiões, propriedades e restrições do sistema e regras que garantem alguns serviços do sistema, assim como a sua funcionalidade. Os requisitos foram estabelecidos junto à orientadora deste trabalho, que atuou como stakeholder do projeto. Descrevem-se a seguir os requisitos funcionais, os não funcionais e os de domínio.

Requisitos funcionais	**Requisitos não funcionais**	**Requisitos de Domínio**
Fornecer a funcionalidade de cadastrar, editar e excluir uma conta de usuário.	Desenvolver a aplicação na plataforma web.	Só poderão ter acesso ao serviço os usuários cadastrados.
Fornecer a funcionalidade de cadastrar, editar e excluir uma categoria de questão.		Só poderá existir uma conta de usuário por email.
Fornecer a funcionalidade de cadastrar, editar e excluir uma questão.		A autenticação só é realizada uma única vez por usuário.
Fornecer acesso ao serviço de criação de questões, e pesquisa de questões por categoria.		
Fornecer uma url para acesso às questões cadastradas no serviço.		

Tabela 2: Requisitos do sistema.

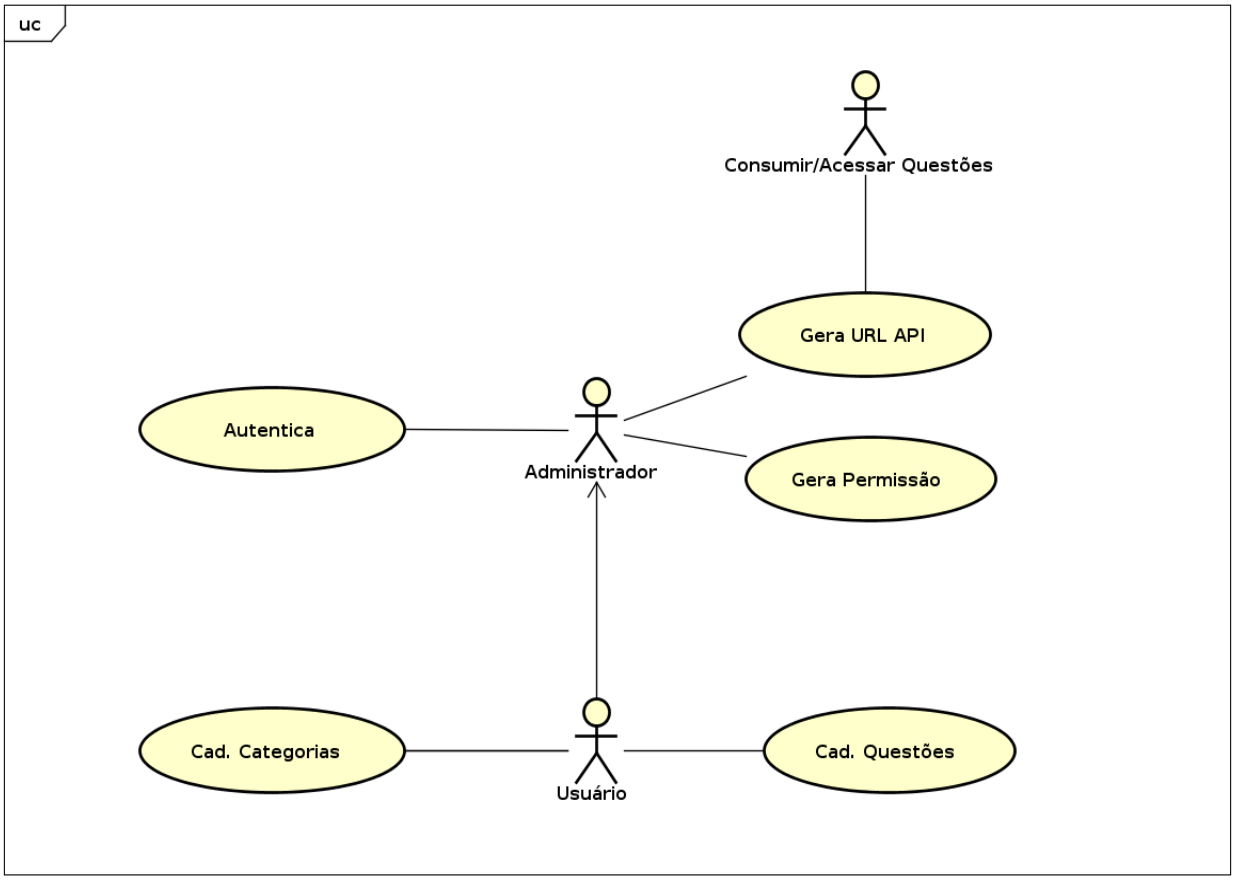
4.1.4 CASOS DE USO

A função dos diagramas de caso de uso é tentar formar uma comunicação entre os usuários e o programador, com o objetivo de fazer com que ambos os lados possam compreender o problema e suas necessidades da mesma forma. Foram identificados alguns diferentes perfis de utilizadores do sistema, dentre eles:

- Usuário – utilizador que cadastra as questões relacionadas a sua área de atuação ou formação.
- Administrador – indivíduo que manipula os acessos necessários para outros aplicativos terem acessos aos dados.
- Aplicativos externos – aplicativos que desejam ter acesso às questões cadastradas pelos usuários.

A Figura 8 expõe o diagrama de casos de uso para o EasyQuiz, o qual apresenta os distintos perfis de atores do sistema e ações que tais atores podem realizar, como as várias tarefas de cadastro, edição e exclusão de questões e acessos de outros aplicativos.

Figura 8: Diagrama de caso de uso do aplicativo EasyQuiz

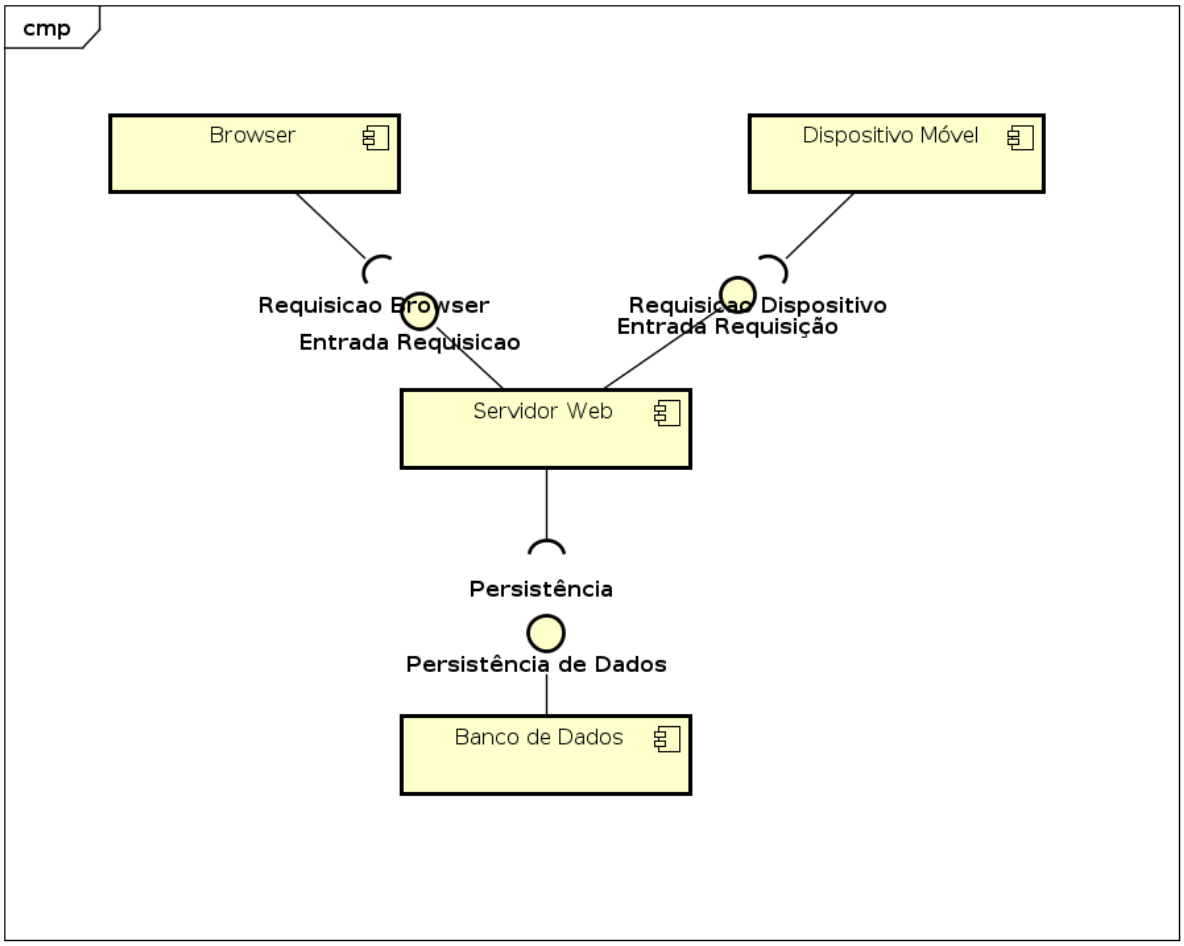


4.2 ARQUITETURA DO SISTEMA

O sistema foi criado a partir de uma arquitetura hierárquica de três camadas. Uma delas é a camada cliente, onde é oferecida a área de atuação do usuário, ou seja, a interface gráfica onde o usuário desempenha suas funções no sistema, que pode ser instanciada em um navegador web a partir de um desktop ou através de um dispositivo móvel. A segunda camada é onde se encontra a parte de maior processamento lógico da aplicação, representada por um servidor de aplicação web, que é o responsável pelas regras de negócio da aplicação e do serviço. A terceira camada é a camada de acesso a dados, onde se encontra o servidor de banco de dados, responsável pela persistência, tratamento e gerenciamento dos dados da aplicação.

Uma abstração da arquitetura descrita está representada pela Figura 9, exibida abaixo:

Figura 9: Diagrama de Componentes do EasyQuiz



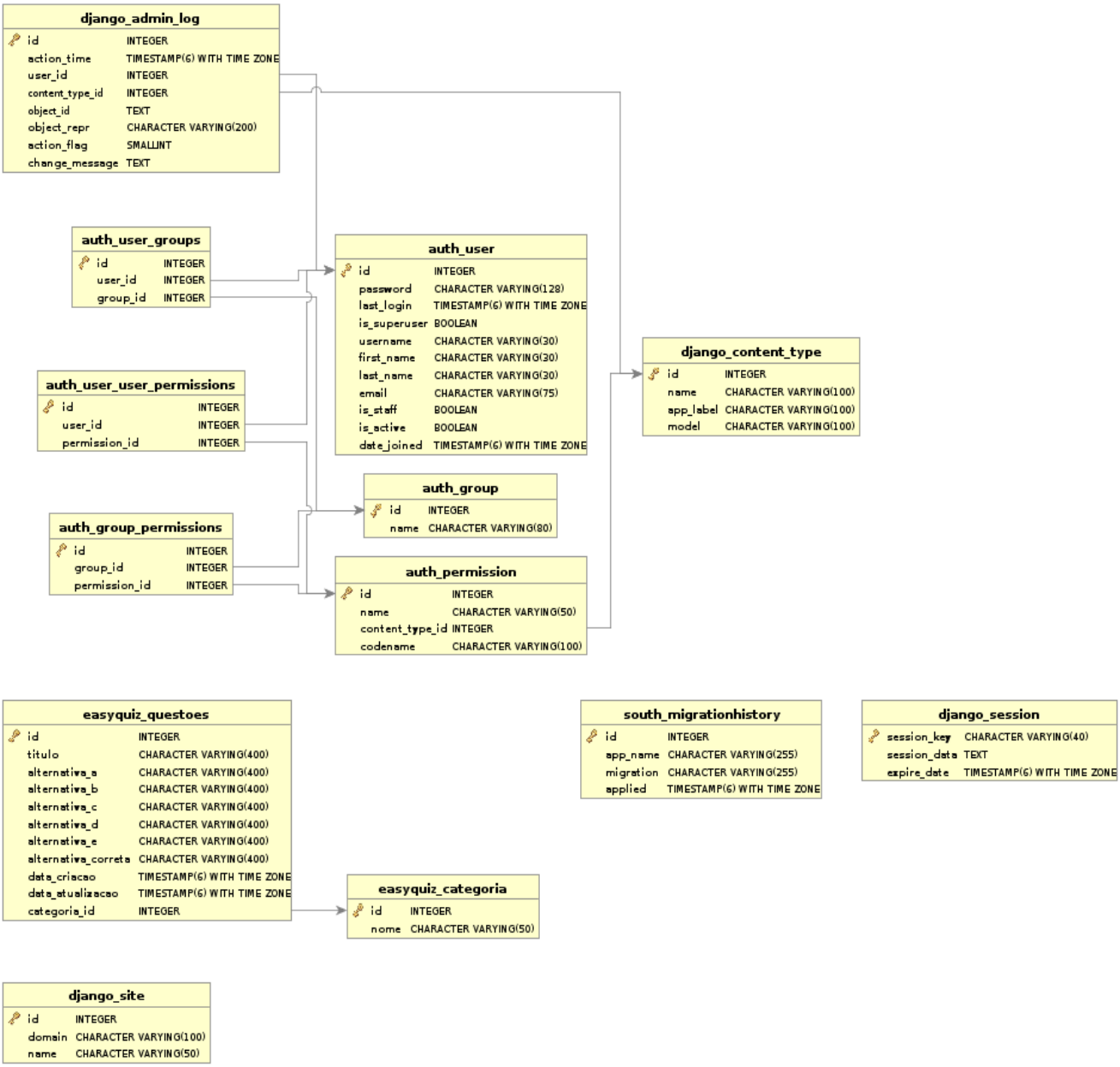
4.3 DIAGRAMA DO BANCO DE DADOS (MER)

O modelo de entidade e relacionamento do banco de dados tem a utilidade de apoiar o desenvolvimento do banco, descrevendo as ideias e os dados pertencentes ao domínio do problema. Nesse sistema a utilização do MER (Modelo de Entidade e Relacionamento) facilitou todas as especificações da estrutura lógica pensada para criação do banco de dados.

O banco de dados é peça fundamental do sistema, pois o serviço que o mesmo proporciona é de extrema importância para o funcionamento do sistema. Sem a utilização de um banco de dados seria muito inviável oferecer funções da aplicação de forma que ocorresse persistência, integridade e coesão dos dados.

A partir do planejamento e o desenvolvimento do modelo entidade e relacionamento, foi realizada a criação da base de dados. Na Figura 10 é mostrado o estado atual da estrutura do banco de dados desta aplicação.

Figura 10: Diagrama Modelo de Entidade e Relacionamento



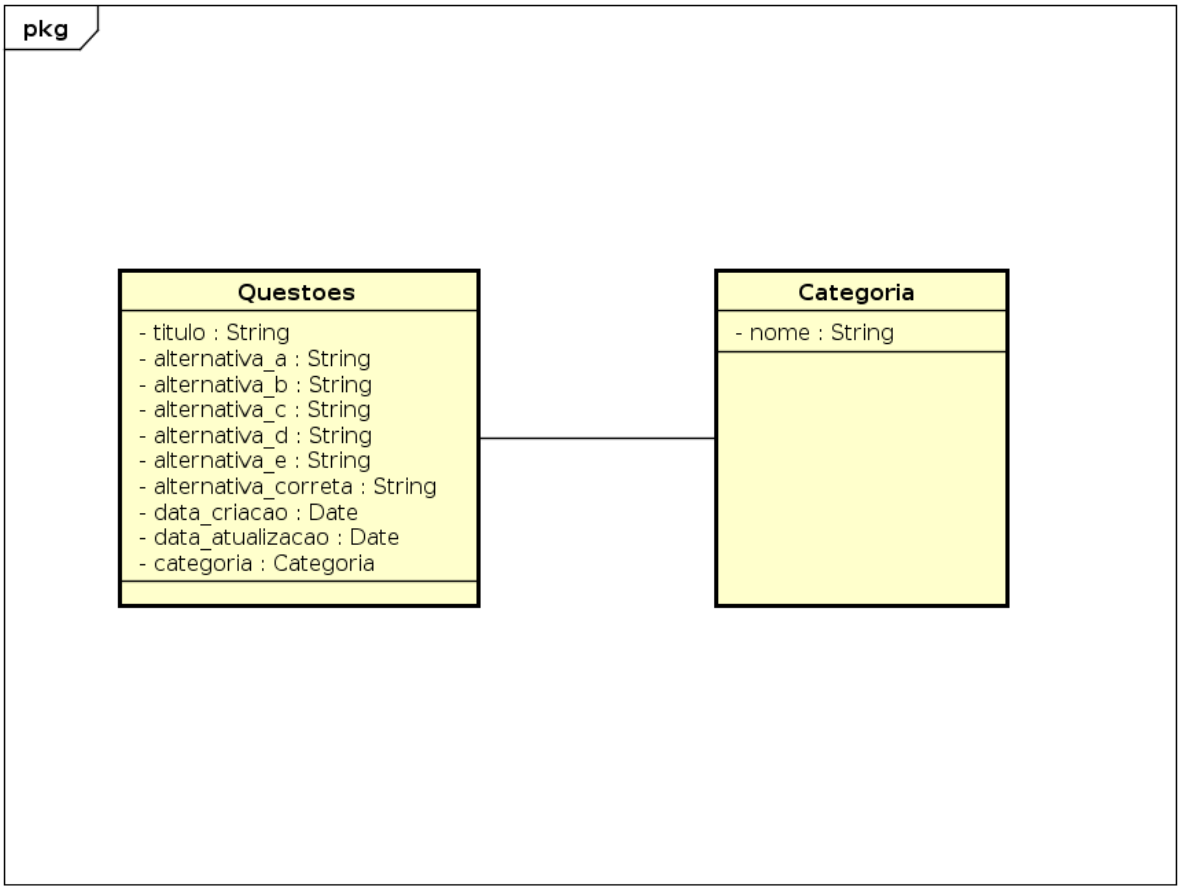
4.4 IMPLEMENTAÇÃO DO PROTÓTIPO

Para realizar uma prova de conceito quanto à solução proposta, mostrando um serviço independente de plataforma, foi implementado um protótipo usando as tecnologias descritas no Capítulo 2.

Por ser uma aplicação que utiliza bastante dado e formulários, e pela ligação das funcionalidades do sistema com a camada de visualização, o desenvolvimento das versões iniciais deste protótipo utiliza um paradigma de desenvolvimento fortemente centrado em dados, com sequências de códigos orientados a objetos e outras procedurais, mas com foco numa fácil legibilidade.

A aplicação é composta por quatro classes e um controlador de funções. Uma classe é responsável por manter os dados referentes às questões a outra por manter os dados das categorias e as demais classes são responsáveis pela criação das instancias para geração dos formulários no framework. O controlador contém as funções de inserção, alteração e deleção para manipular os dados da aplicação. A Figura 13 apresenta o diagrama UML (*Unified Modeling Language*) das classes Questões e Categorias.

Figura 13: Diagrama de classes da aplicação



Outra funcionalidade bastante importante é o *web service* disponibilizado pela aplicação, possibilitando o acesso aos dados já armazenados por outros aplicativos. A Figura 14 mostra um trecho da geração da url para acesso aos dados das questões já cadastradas.

Figura 14: Trecho da geração da url pelo serviço no Dreamfactory

Request URL

```
http://127.0.0.1:8081/rest/easyquiz/easyquiz_questoes
```

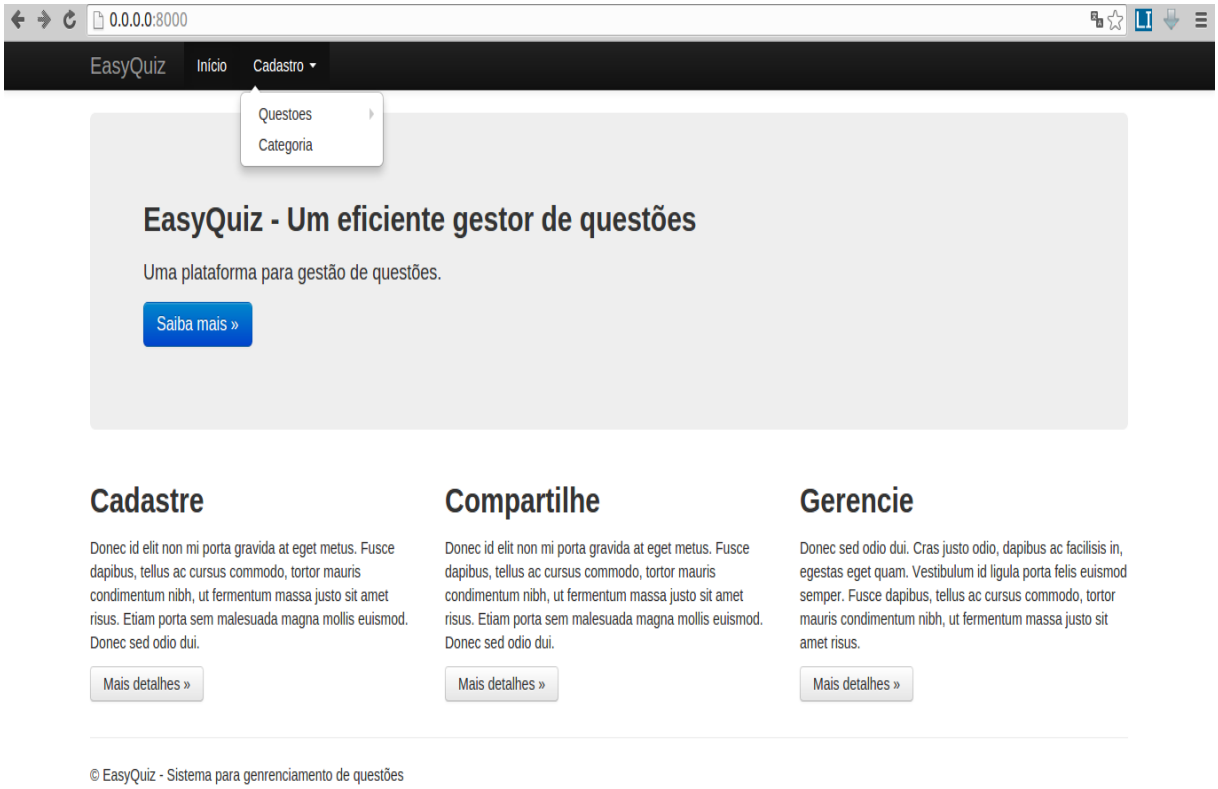
Response Body


```
{
  "record": [
    {
      "id": 3,
      "titulo": "Python é...",
      "alternativa_a": "Linguagem de Programação",
      "alternativa_b": "Animal",
      "alternativa_c": "Inseto",
      "alternativa_d": "Vírus",
      "alternativa_e": "NDA",
      "alternativa_correta": "a)",
      "data_criacao": "2015-12-04 19:55:17.554528-03",
      "data_atualizacao": "2015-12-04 19:55:17.55457-03",
      "categoria_id": 5
    },
    {
      "id": 4,
      "titulo": "PgAdmin é...",
```

4.4.1 TELAS DO UTILIZADOR

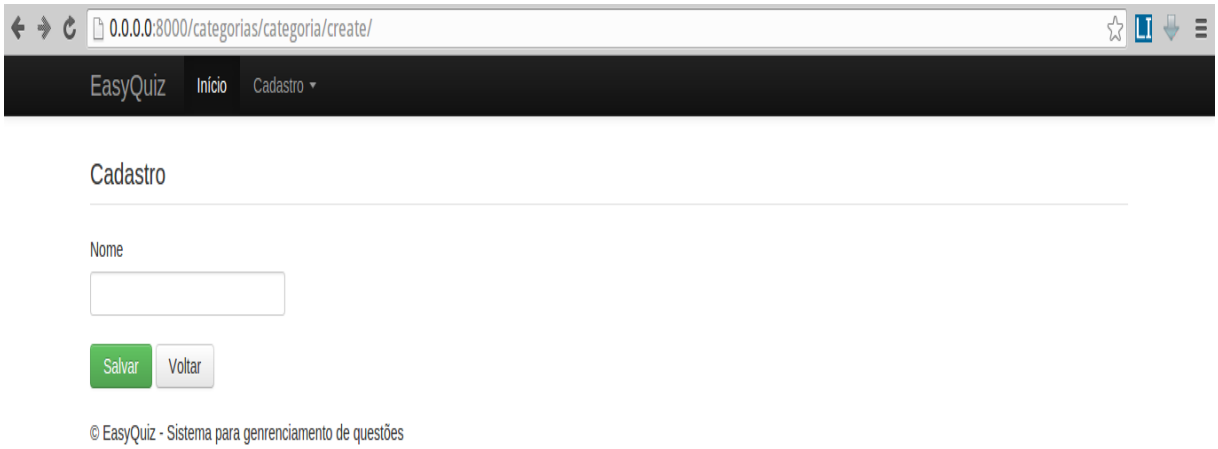
A Figura 15 mostra a tela inicial do sistema e suas funcionalidades principais.

Figura 15: Tela inicial do sistema EasyQuiz



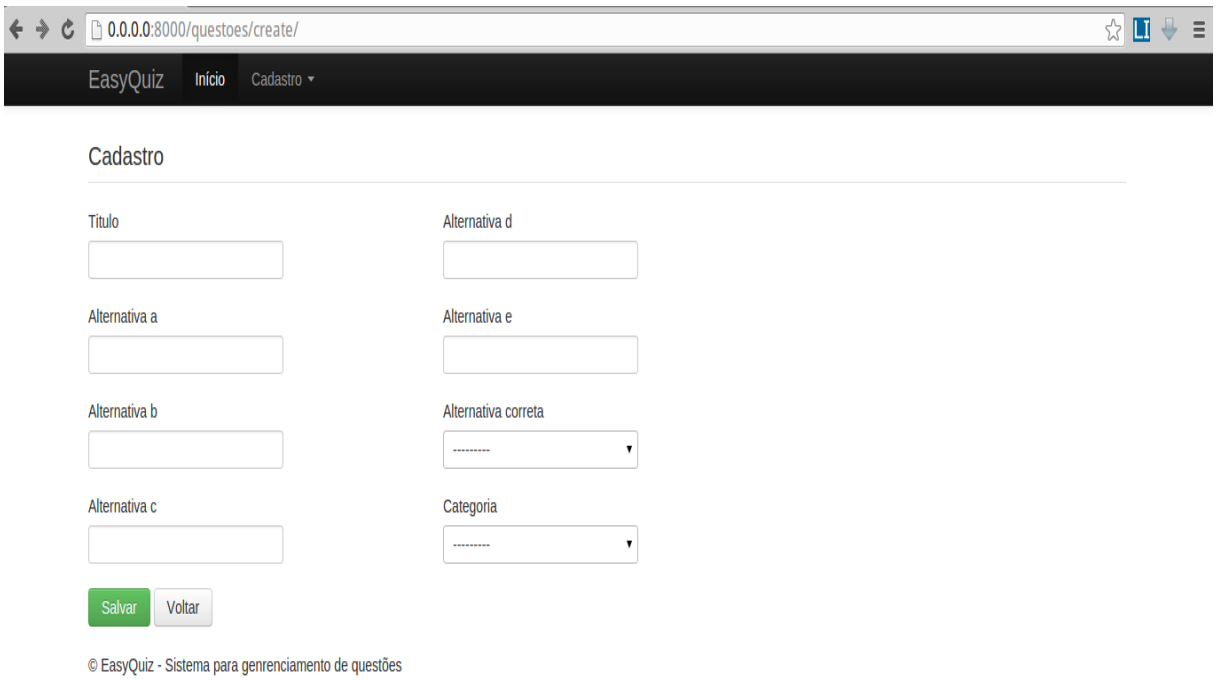
Na Figura 16 é apresentado a forma de como os usuários poderão cadastrar as categorias no sistema.

Figura 16: Tela do cadastro das categorias



Na Figura 17 é ilustrada a tela onde as questões são criadas.

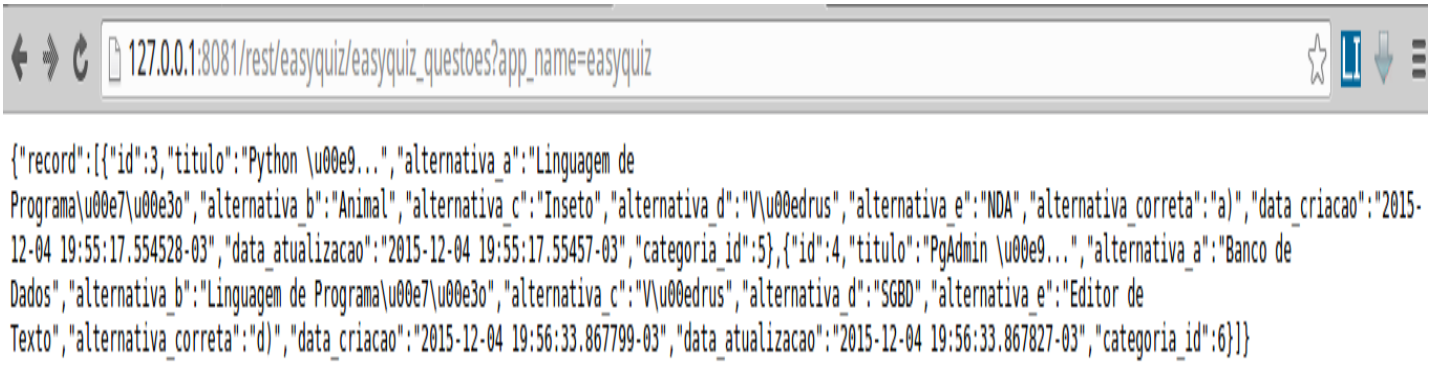
Figura 17: Tela do cadastro de questões do sistema



4.5 WEB SERVICE DO EASYQUIZ

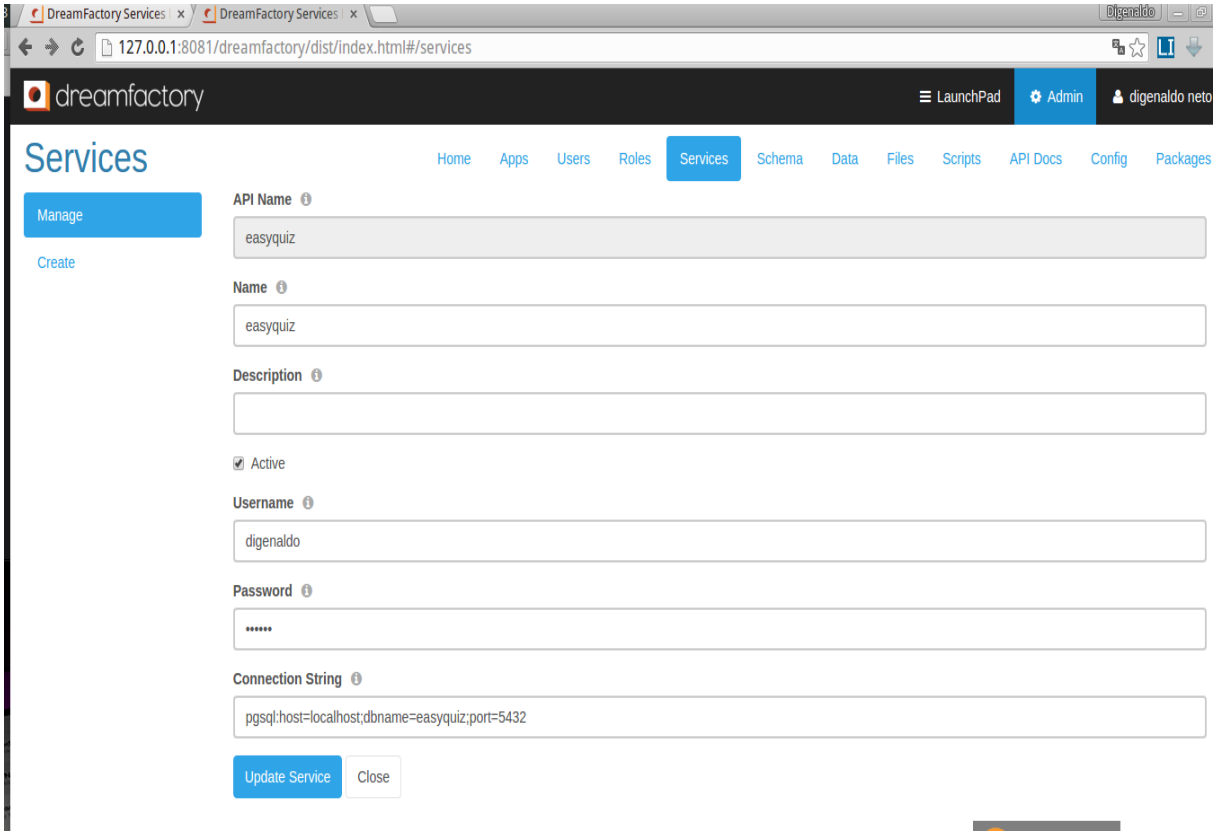
Um dos pontos chaves da aplicação é a oferta do serviço web para que outras aplicações ou usuários possam ter acesso aos dados. Neste protótipo foi implantado o framework Dreamfactory que possibilitou todas as configurações para geração e acesso a API REST que é oferecida pela aplicação. Na Figura 18 é apresentado um trecho do url para acesso aos dados cadastrados.

Figura 18: Tela de dados gerados pelo *web service*



Na Figura 19 se ilustra como é feito o mapeamento do Dreamfactory para o banco de dados. O Dreamfactory é pré-configurado para aceitar as configurações para acesso de diversos bancos de dados.

Figura 19: Tela da configuração para mapeamento do banco de dados



O sistema já possibilita o cadastro de questões de múltipla escolha, e já é possível ter acesso via API REST às questões cadastradas. O serviço ainda precisa se moldar aos outros tipos de questões.

5 CONSIDERAÇÕES FINAIS

Acredita-se que o presente trabalho de conclusão de curso permitiu a exploração e uso de novas tecnologias com todas as suas características e funcionalidades, e o desenvolvimento e documentação de um serviço baseado em necessidades reais de usuários que buscam a criação e disponibilização de questões de maneira prática. Além disso, este TCC permitiu a utilização de diversos conceitos vistos nas disciplinas do curso.

O serviço desenvolvido para cadastro de questões ainda não possui uma versão estável, sendo necessário o melhoramento de algumas funcionalidades como segurança, interface mais atraente e intuitiva, a realização de mais testes, além da implementação de outras funcionalidades que farão com que o sistema possa ser consolidado e possa ter vários utilizadores.

Foi possível concluir a criação de um serviço independente de plataforma e que permite o acesso por outros aplicativos via API REST.

Em relação ao sistema, vale ressaltar que dentro de um curto período de tempo foi possível especificar, desenvolver, testar em ambiente real e concluir que o mesmo atinge ao objetivo principal deste trabalho, que foi o de disponibilizar um serviço web independente de plataforma que gerencia de maneira compartilhada dados de questões e possa auxiliar os usuários na criação e compartilhamento de questões, as quais podem ser exploradas, por exemplo, em provas ou simulados sobre determinados conteúdos.Como trabalhos futuros, uma sugestão é o desenvolvimento de diferentes aplicativos que se comuniquem com o sistema proposto e desenvolvidos em diferentes linguagens e plataformas, como forma de validar a proposta.

6 REFERÊNCIAS BIBLIOGRÁFICAS

JOSUTTIS, M. N. **Soa in pratice The Art of Distributed System Design**. 1.ed. United States of America: Livros Técnicos e Científicos, 2007. 229p.

CERAMI, E. **Web Services Essentials**. 1.ed. United States of America: Livros Técnicos e Científicos, 2002. 304p.

WEBBER, S. I. **REST in Pratice**. 1.ed. United States of America: Livros Técnicos e Científicos, 2010. 415p.

ALLAMARAJU, **RESTFul Web Service Cookbook**. 1.ed. United States of America: Livros Técnicos e Científicos, 2010. 285p.

DOUG, T. J. P. **Programming Web Services with SOAP**. 1.ed. United States of America: Livros Técnicos e Científicos, 2001. 216p.

BATES, A. W. **Effective Teaching with Tecnology in Higher Education: Foundations for Success**. 1.ed. United States of America: Livros Técnicos e Científicos, 2003. 336p.

DJANGO: The Web framework for perfectionists with deadlines. **Django**, 2015. Disponivel em: <https://www.djangoproject.com/>. Acesso em: 1 Novembro 2015.

DREAMFACTORY. **Dreamfactory**, 2015. Disponivel em: <https://www.dreamfactory.com/>. Acesso em: 5 Novembro 2015.

SOFFA, M. M.; TORRES, P. L. **O Processo Ensino-Aprendizagem Mediado Pelas Teconologias da Informação e Comunicação da Informação de Professores On-Line**, Parana, 2009. 12.

PYTHON.ORG. **Python**, 2015. Disponivel em: <https://www.python.org/>. Acesso em: 5 Novembro 2015

DILLENBOURG; JÄRVELÄ; FISCHER. **The Evolution of Research on Computer-Supported Collaborative Learning. In Technology-Enhanced Learning**, Estados Unidos, 2009. 3-19.

MOODLE: Community driven, globally supported. **Moodle**, 2015. Disponivel em: <https://moodle.org>. Acesso em: 10 Outubro 2015.

PostgreSQL, 1997. Disponivel em: <http://www.postgresql.org/>. Acesso em: 10 Novembro 2015.

EDUCANDUS. Disponível em:< http://www.educandus.com.br/produtos/>. Acesso em: 10 nov. 2015.

AIX Sistema. educador.net. Disponível em: <http://www.oeducador.net/>. Acesso em: 10 nov. 2015.

GUEDES, T.A.Gilleanes. UML2: Uma Abordagem Prática. São Paulo. NOVATEC. 2009. 481p.

WONDERSHARE Software Co., Ltd. QuizCreator. Disponível em: <http://www.sameshow.com/quiz-creator.html>. Acesso em: 10 nov. 2015.

MICHAL TOMLEIN. iTest: Computerised examination made easy. Disponível em:< http://itest.sourceforge.net/about.shtml>. Acesso em: 10 nov. 2015.

STARLINE. SGP. Disponível em:<http://www.sistemasmart.com.br/sgp/>. Acesso em: 10 nov. 2015.

Ferreira C. **Comparando Aplicações Web Service**, Paranavai, 2014. 6.

MORAES, M. B. D. **Viabilidade da Linguagem Python no Desenvolvimento de Aplicações de Alto Desempenho**, Limeira, 2006. 4.