UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE QUÍMICA

GUSTAVO VALENCIO TOFOLO

Simuladores baseados em vídeo usando recursos da plataforma Google

São Paulo
2021

SUMÁRIO

1 INTRODUÇÃO

O ensino experimental desempenha um papel central na educação, pois, além de corroborar com o material teórico, dá cerne ao processo ativo de construção do conhecimento (1). Contudo, ele nem sempre é viável, por motivos como: custo, tempo, periculosidade, problemas com descarte, entre outros. Por outro lado, a introdução de recursos educacionais baseados em tecnologia digital possibilita a realização de simulações de experimentos.

Atividades envolvendo estes recursos não contemplam objetivos de aprendizagem relacionados ao ambiente físico do laboratório, como a observação fenomenológica e o desenvolvimento de habilidades e procedimentos técnicos. Contudo, quando inseridas em propostas de caráter mais investigativo, as simulações de experimentos podem tratar bem de pontos como a formulação de hipóteses e a atribuição de sentido aos dados experimentais.

Dentre as inúmeras propostas de simuladores de experimentos, destaca-se para os fins desta pesquisa aqueles baseados em vídeos interativos. Esta abordagem inovadora substitui representações tridimensionais pictóricas digitais por registros em vídeo de experimentos reais. Nestes simuladores, os estudantes podem de fato realizar medidas de leitura de escalas e observar fenômenos capturados por eles, ou por seus docentes. Assim, combinados a uma planilha, estas gravações compõem um recurso para aquisição e tratamento de dados.

A criação destes simuladores pode ser feita de maneira simples, através da inserção de vídeos em planilhas do Excel, somada a tratamentos posteriores, por exemplo. No entanto, há questões ligadas à disseminação destes recursos que devem ser consideradas, como: necessidade de conhecimentos consideráveis em outras ferramentas – tanto do professor, quanto do aluno –, difícil escalabilidade e difusão, não funcionamento em diferentes dispositivos, fluxo não dinâmico de dados, entre outros.

Felizmente, o Google, que vem se destacando no meio educacional durante a pandemia devido ao uso corriqueiro de recursos como Classroom e Meet, fornece recursos que, quando integrados, podem driblar todos os obstáculos citados até então. Nesse contexto, o presente projeto visa desenvolver, através desta integração, uma forma simplificada de criar e distribuir tais simuladores.

2 OBJETIVOS

- Integrar recursos do Google, como: Youtube, Sheets, Forms e Classroom;
- Através deste ambiente integrado, desenvolver uma ferramenta que permita a criação e compartilhamento de simuladores baseados em vídeos;
- Fazer deste um ambiente democrático e didático, que possa ser acessado em diferentes dispositivos – celulares ou computadores, independente do sistema operacional – e entregue a alunos do sistema público de ensino, por exemplo.

3 METODOLOGIA

As aplicações aqui desenvolvidas derivam de trabalhos elaborados por Danilo Macedo (2) e Lígia Bozzi (3), em suas respectivas dissertações de mestrado, orientados, também, pelo professor Guilherme Marson. Em ambas as teses, foi defendido o uso da simulação baseada em hipervídeos. No entanto, as pesquisas foram direcionadas a temáticas pontuais de ensino. Logo, essa abordagem não compreende, inteiramente, os objetivos desse projeto, visto que o mesmo é mais abrangente e generalista.

Para que haja uma adaptação ao atual contexto, deve-se refletir sobre os princípios que fundamentam a nova ideia, e definir requisitos e formas de avanço baseados nesses valores. A fim de enxugar esta seção, três pilares foram estipulados, sendo eles: finalidade, público-alvo e difusão.

3.1 Pilares estratégicos

3.1.1 Finalidade

Estabeleceu-se como finalidade o simples acesso, criação e divulgação destes simuladores, no qual não há amarras quanto a sua temática, nem dificuldade de escalabilidade e compartilhamento.

3.1.1.1 Ferramentas [1]

"Acesso, criação e divulgação" é, na visão dos desenvolvedores, sinônimo de aplicação web e, para a criação da mesma, foram utilizadas duas ferramentas distintas, sendo elas: Apps Script (4), para a *v1*, e Django (5), para a *v2*. Elas, por sua vez, lidam com a lógica do lado do servidor, ou seja, com o back-end.

Dentre os papéis do back-end, têm-se como principal a integração das bases de dados com a aplicação. Feito isso, torna-se prático o ato de criar, editar e remover objetos, assim como, relacioná-los. Para tal, os dados foram manipulados a partir dos seguintes recursos: planilha Sheets, para a *v1*, e SQLite3 (6) / SGBD PostgreSQL (7) para a *v2*.

3.1.2 Público-alvo

A aplicação é destinada a alunos e professores e, para ambos os grupos, os requisitos de uso devem ser mínimos, pois, estando em um ambiente de amplo acesso, a facilidade e clareza durante seu uso devem ser assimiladas tanto por usuários experientes, quanto por

[1] Os recursos utilizados durante o período de produção foram variados, tendo resultado em duas versões distintas, chamadas, provisoriamente, de *v1* e *v2*.

leigos. Dessa forma, o serviço pode ser facilmente distribuído em diferentes nichos educacionais.

3.1.2.1 Ferramentas

Com foco no usuário comum, construiu-se o front-end da aplicação. Ele é a "cara" do que será entregue, ou seja, o design das interfaces, formulários e outros elementos visuais. Para isso, três siglas foram estudadas, sendo elas: HTML (HyperText Markup Language), CSS (Cascading Style Sheets) e JS (JavaScript), fundamentais para a criação de páginas web.

Somado a isso, o framework CSS, Bootstrap (8), o software Bootstrap Studio (9), e o pacote python-django, Widget Tweaks (10), foram utilizados, a fim de entregar um visual profissional, amigável e responsivo.

3.1.3 Difusão

Com a plataforma na web, professores capacitados – através de um curso de extensão, por exemplo – podem criar seus simuladores e, através de URLs únicas, encaminhá-los a seus alunos. Fora isso, podem também os dispor ao público em uma aba semelhante a um blog, na qual experimentos servem de inspiração e recurso para a comunidade científica.

3.1.3.1 Ferramentas

A plataforma de nuvem Heroku (11) já hospeda ativamente o sistema (12), por meio de um plano gratuito que não suporta um fluxo de dados robusto. Por isso, elementos como vídeos não são armazenados, mas sim, suas URLs, que são posteriormente controladas através de códigos de terceiros, como a Youtube Player API (13), quando renderizados.

Já para lidar com a página de blog, o pacote python-django, Django Taggit (14), e o framework JS, Tagify (15), levaram a implementação de um sistema de marcadores refinado, que enriquece a navegação do usuário entre os simuladores públicos.

Por fim, a reunião dessas tecnologias ocorreu no ambiente de desenvolvimento Pycharm (16), na *v2*, e no próprio editor de scripts do Google, na *v1*.

4 RESULTADOS E DISCUSSÃO

4.1 Primeira versão

Inicialmente, o projeto se limitava ao desenvolvimento de uma planilha Sheets com inserção manual de vídeos, que, posteriormente, seria replicável através de configurações de parâmetro no próprio arquivo. Note que isso atenderia, de forma simples, os objetivos propostos. Mas, os obstáculos foram muitos.

O primeiro passo era simples: inserir um vídeo do Youtube nessa planilha. No entanto, as duas soluções para esta situação esbarraram em problemas maiores, já que a inserção direta não é, no momento desta escrita, possível. São elas:

1. Criar um Slides (que permite inserção de vídeo) e adicioná-lo como um objeto Desenho dentro do Sheets;
    - Problema: na fase de replicação, traria dificuldades na parte de configuração dos parâmetros devido a conversa entre muitos mecanismos.
2. Através do Sheets API da Google, utilizar Apps Script para criar menus extras na interface padrão do Sheets e, nos itens desse menu, renderizar código HTML;
    - Problema: os menus criados não são visíveis em dispositivos móveis.

Frente a isso, a saída seria a criação de um aplicativo web simples, com um back-end controlado pelo próprio Apps Script, que forneceria um sistema básico para controle de requisições e respostas web. Assim, qualquer dispositivo com acesso à internet poderia acessá-lo e o Google continuaria sendo uma ferramenta exclusiva.

Porém, a máxima alcançada na primeira versão foi o desenvolvimento de procedimentos de autenticação rudimentares que usavam um Sheets como “banco de dados”. A dificuldade do avanço se deu, principalmente, pelo fato do Apps Script não ser dedicado ao desenvolvimento web, o que justifica sua ineficiência quando aplicado a projetos escaláveis. Tópicos fundamentais como cache, sessões, segurança e tratamento de dados se mostraram cansativos e, em muitas vezes, impraticáveis. Além do mais, muitos dos recursos que poderiam auxiliar a enfrentar estas questões só funcionavam em plataforma em nuvens específicas que requerem configuração manual ou paga.

Ademais, os vídeos do Youtube conseguiram ser manipulados através da Youtube Player API. Assim como, as inúmeras falhas trouxeram um entender mais sólido do funcionamento da web, que foram essenciais para os sucessos segunda versão, que contou com o auxílio de uma ferramenta potente e dedicada: o framework python, Django.

4.2 Segunda versão

https://ic-deploy.herokuapp.com/account/login/

4.2.1 Componentes e recursos desenvolvidos

Defronte ao ocorrido na *v1*, foi decidido recorrer a alternativas consolidadas, que serão aprofundadas dentro de dois grandes grupos, sendo eles: back-end e front-end.

4.2.1.1 Back-end

Como previamente dito, o back-end é a programação que cuida dos processos internos da aplicação. Atualmente, existem diferentes ferramentas que buscam automatizar a sobrecarga relacionada a esse desenvolvimento, como os frameworks web.

Dentre os disponíveis, um se encaixou perfeitamente às necessidades do projeto: o Django. Sua autodescrição, encontrada em seu site, justifica o porquê de seu uso:

> Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Sem a necessidade de "reinventar a roda", mostrou-se cabível seu estudo.

4.2.1.1.1 Arquitetura do Django

Um projeto Django é construído a partir da interação entre diferentes arquivos .py (extensão para arquivos de código python). Eles, que podem ser Models, Views ou Templates, nomeiam esta estrutura como MVT. Nela, todas essas entidades são dependentes umas das outras, seguindo a lógica do fluxograma a seguir:

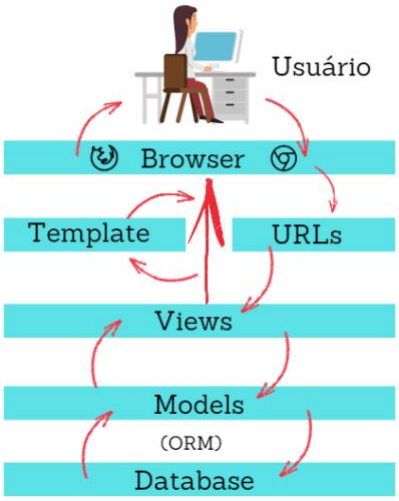

Figura 1 - Fluxograma que descreve o funcionamento do Django (17)

Resumidamente, temos que o usuário, através de um navegador, acessa uma URL. Essa requisição é tratada pela View, que lida com a lógica da aplicação que, se necessário, acessa os Models. Eles são representações gerais dos modelos presentes no banco de dados, que auxiliam as Views nas consultas. Quando a View pega o que precisa, ela retorna isso ao usuário no formato de Template, ou seja, como elemento visual (front-end).

Bastou adequar isso ao contexto do projeto.

4.2.1.1.2 Models

A modelagem desenvolvida é simples, visto que são dois os objetos principais da aplicação: os usuários, e os simuladores. Eles, representados por tabelas, são caracterizados por atributos: campos que descrevem o que deve ser armazenado. Os principais são:

Conta	**Simulador**
ID: chamado de chave primária, é um valor único que identifica a entidade.	ID.
E-mail: campo que será utilizado, juntamente com a senha, na autenticação.	Título: necessário para identificação.
Senha: necessário para autenticação.	Conceitos requeridos: base teórica ideal para bom uso do simulador.
Primeiro nome: necessário para identificação.	Conceitos mínimos: base teórica mínima.
Último nome: necessário para identificação.	Dimensões da tabela: variáveis de estudo que darão nome às colunas da tabela auxiliar ao experimento.
Nome da instituição de ensino do usuário (aluno, professor ou associado): necessário para identificação.	Link do Youtube: link do vídeo do experimento.
Mensagem de solicitação: texto fornecido pelo usuário no momento do registro. Baseado nele, o administrador pode, ou não, aceitar a solicitação de uso.	Link do Forms: link do formulário que terá questões relacionadas a simulação.
Situação da conta: status da autorização. “Autorizado”, se aceito pelo administrador, “não autorizado”, se não, “pendente”, se em espera. Necessário para autenticação.	Token: série de caracteres aleatórios que estarão nas URLs dos simuladores a fim de restringir o acesso dos simuladores privados.
	Privado: define se o simulador será de acesso público ou não.
	ID de conta: chamada de chave estrangeira, relaciona a conta ao simulador. Define um relacionamento de um-para-muitos, ou seja, cada conta pode ter N simuladores.

Tabela 1 – Atributos das tabelas detalhados

Dito isso, é praticável, através de um banco de dados e uma linguagem de consulta, gerar e manipular essas tabelas. Felizmente, com o Django, é possível definir esta modelagem através de classes em python, no qual ele faz o trabalho de construir o mapeamento restante. Assim, nenhuma linha de código SQL (linguagem de consulta oficial) é necessária.

Ainda mais – sem a necessidade de qualquer configuração –, todo novo projeto é acompanhado de um banco de dados SQLite3, que, além de ser leve, é, por muito pouco, preenchido por esse mapeamento.

E, a fim de deixar a plataforma mais robusta, este banco pode ser trocado por outro, como o sistema gerenciador de banco de dados PostgreSQL. Este, que também tem compatibilidade com o Django, herda o positivo da ferramenta anterior e entrega mais, pelo preço de ser mais complexo.

Feito isso, cabe às Views conversar – com base no que foi definido nas Models – com o PostgreSQL. Este, deve manipular as informações da base de dados.

4.2.1.1.3 Views

Dentre as desenvolvidas a fim de atender às exigências do projeto, se mostrou como mais urgente as de autenticação, visto que qualquer implementação posterior teria como requisito um sistema de usuários. Para isso, soluções para login, logout, mudança de senha, redefinição de senha e seus pormenores teriam de ser desenvolvidos do zero.

Porém, isso não foi necessário, já que o Django é embutido com um framework de autenticação pronto que cria estruturas capazes de lidar com usuários, permissões, grupos e sessões, o que inclui uma lista de Views base de fácil configuração.

Uma vez que os usuários podem acessar o aplicativo de forma segura e eficiente, eles precisam se registrar, e ter o que fazer neste ambiente. Novamente, o Django agiliza este processo. Para isso, ele fornece cinco Views baseadas em classe genéricas, sendo elas: ListView, DetailView, CreateView, UpdateView e DeleteView. A partir de conceitos da Orientação a Objetos, é possível estender essas classes, alterando atributos e métodos a fim de moldá-las ao seu contexto. Como os nomes sugerem, elas listam, detalham, criam, atualizam e excluem dados, respectivamente.

Estas ações simples descrevem toda e qualquer interação que ocorre com o banco de dados. Quanto a seus usos frente a aplicação, acompanhe:

View Genérica	**Utilização**
ListView	1. Dashboard do usuário. Deve listar os simuladores pertencentes a conta. O filtro ocorre por meio das chaves primárias e estrangeiras; 2. Comunidade. Deve listar qualquer simulador não privado, independente do usuário. O filtro ocorre por meio do atributo “privado”, que podem ser refinados através de campos de busca.

DetailView	1. Simulação. Deve dar ao template os atributos do simulador acessado, para que o vídeo do Youtube possa ser renderizado, por exemplo.
CreateView	1. Criação de simulador. Deve fornecer ao Template um formulário com campos apoiados aos atributos do Model do simulador. Se preenchido corretamente, o adiciona à base de dados; 2. Atualização de conta. Semelhante a anterior, mas baseada no Model da conta.
UpdateView	1. Atualização do simulador. Semelhante a View de criação. Porém, ao invés de criar, ele atualiza uma entidade já existente; 2. Atualização da conta. Semelhante a anterior, mas baseada no Model da conta; 3. Atualização do Token. Semelhante a anterior, mas baseado só no atributo token, que é aleatório e caracteriza as URLs privadas.
DeleteView	1. Exclusão de simulador. Deve fornecer ao Template um botão que exclui a entidade em questão.

Tabela 2 – Views baseadas em classe genéricas e suas potenciais utilizações

Para que isso funcione corretamente, restrições importantes foram definidas no código, sendo elas: usuários não autenticados têm acesso limitado, pois não podem, por exemplo, ter um dashboard; usuários autenticados não podem alterar ou deletar objetos que pertencem a outro usuário. Ademais, páginas estáticas não precisam de qualquer tratamento.

Além disso, o Django cria, por padrão, uma área de administração completa e automática, que lê metadados para fornecer uma interface rápida e centrada nos Models, onde usuários confiáveis podem gerenciar simuladores, variáveis de impacto e requisições de uso. E, a fim de melhorar ainda mais esta seção, foi configurado um servidor SMTP simples, que interage diretamente com o usuário via e-mail em situações como sua autorização, por exemplo.

Finalmente, cabe aos Templates apresentar o conteúdo e interagir com o usuário.

4.2.1.2 Front-end

Frente a combinação “usuário comum + plataforma de aprendizado”, foi construída uma plataforma simples e minimalista, com ênfase no uso fácil e intuitivo, e textos precisos.

4.2.1.2.1 Templates

Seguindo esta lógica, a fonte ABeeZee, auto-descrita como amigável e voltada ao aprendizado, se mostrou adequada. O mesmo pode ser dito sobre a adoção de uma paleta de cores pastel, tidas geralmente como “suaves”.

Para concretizar isso, foi preciso, através da linguagem HTML, construir a estrutura de cada página. É ela que, por meio de tags, diz ao navegador o que deve ser renderizado. No entanto, o resultado é simplório, já que nenhuma customização é possível. Isso se resolveu com a adoção do CSS, um mecanismo que, através de identificadores, leva estilo aos elementos. Por fim, qualquer funcionamento mais dedicado foi implementado com o JavaScript.

Porém, a escrita dos recursos visuais, se não acompanhada de um framework a parte pode ser extensa, já que tópicos como o layout do site em diferentes telas devem ser definidos aqui. Assim, foi escolhida para uso o Bootstrap, framework CSS que contém inúmeros arranjos prontos para uso. Ele, que funciona num sistema de colunas e linhas, as rearranja automaticamente através de quando a largura ou altura da tela se alteram. Dessa forma, só foi necessário construir os layouts, e realizar alterações simples sobre os arranjos. E, para facilitar isso, o software Bootstrap Studio foi utilizado, que forneceu recursos de drag-and-drop elegantes, e um output automático do código que representava o que estava sendo construído.

Dito isso, estruturas que fossem gerais a todas as páginas puderam ser criadas, como a barra de navegação, e o rodapé. E, para que eles não tenham de ser repetidos a cada página, o Django, que também fornece facilitadores para esta etapa, permite que um elemento seja definido como a base do projeto, enquanto outros, são acoplados a este.

O que era acoplado era definido pelas variáveis de contexto. Elas, são variáveis entregues pelas Views que podem ser renderizadas ao usuário. Numa CreateView, por exemplo, este container central era preenchido pelo formulário que, baseado no Model, já tinha os campos adequados. Ademais, a customização deste específico elemento foi feito através do pacote python-django Widget Tweaks, visto que o Django nativo não suportava esta tarefa.

Finalmente, para aumentar o dinamismo da navegação na página comunidade, um sistema de marcadores foi implementado. Ele, que permeia back-end e front-end, foi implementado através de dois componentes externos: o pacote python-django, Django Taggit, e a framework JS, Tagify. A primeira, por fácil configuração, construiu um Model de Tags que poderia ser levado às Views pelas variáveis de contexto. Assim, cabia fazer o tratamento destes valores nos Templates, que foram administrados pela segunda.

Segundo a extensão Chrome, GitHub Gloc, o repositório da *v2* conta com 12 mil linhas de código (18), contra 1.2 mil na *v1* (19). Estes valores justificam a escala do relatório e a sua abordagem simplificada. Para não estender ainda mais, as imagens das telas também estão nestes repositórios.

4.2.2 Procedimentos operacionais para usar a plataforma

Segue fluxogramas de navegação da plataforma:

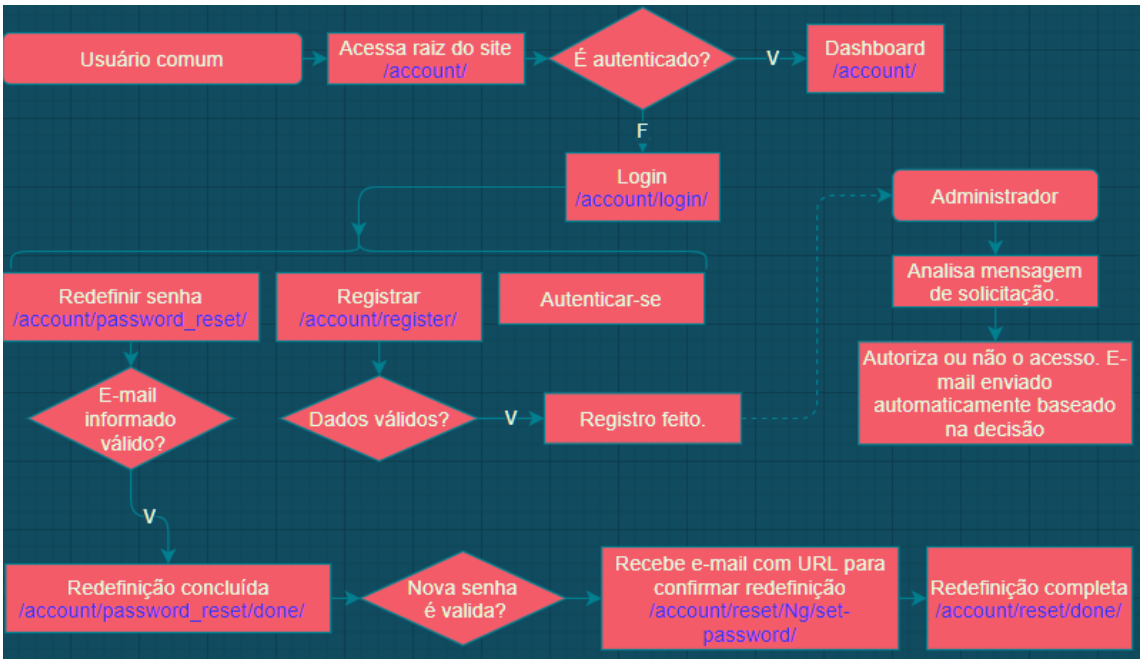

Figura 2 - Fluxograma que descreve a autenticação

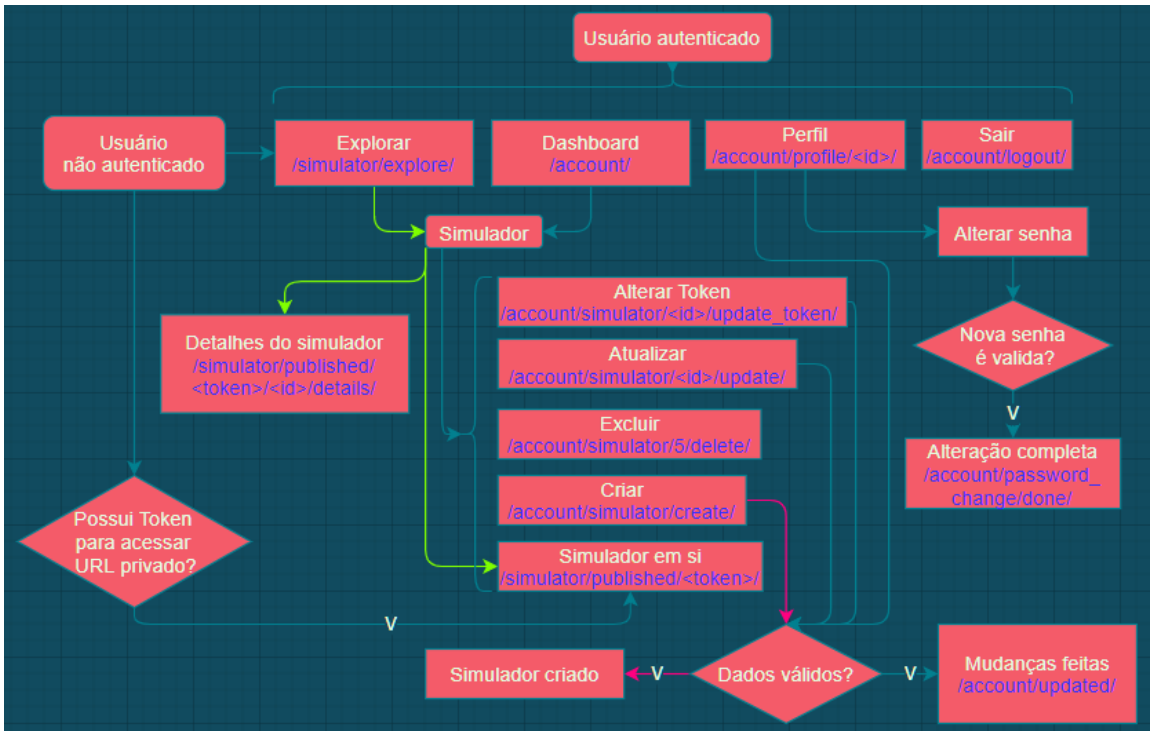

Figura 3 - Fluxograma que descreve a plataforma em geral

5 CONCLUSÃO E PERSPECTIVAS FUTURAS

Vê-se que, junto a ferramentas modernas, é possível construir uma plataforma completa que tem potencial para disseminar os simuladores baseados em vídeo adentre a comunidade científica. No entanto, enquanto muito sobre a estrutura, segurança e autenticação foi discutido, pouco foi feito quanto a como esta simulação aparecerá ao usuário. Ademais, isto será tratado em um relatório futuro, que será desenvolvido na disciplina Introdução à Tecnologia ou à Pesquisa Científica II.

Ainda assim, os objetivos propostos provaram ser alcançáveis, já que, a aplicação já pode ser acessada por qualquer dispositivo que tenha acesso à Internet, e tudo por meio de uma interação usuário-servidor simples e descomplicada. Logo, com uma base sólida e eficiente, novos recursos podem ser acoplados ao sistema existente e, com os Models devidamente construídos, será fácil manipular vídeos e formulários pelas variáveis de contexto, dentro dos Templates. Além disso, o blog pode ser implementado, com a adição de número de visualizações, favoritos, compartilhamento direto em redes sociais, embeds para outras páginas web e afins.

O que foi até então construído já é o suficiente para ser apresentado a diferentes instituições, a fim de conseguir apoio, divulgação e/ou vínculo. Como possibilidade, tem-se o portal Química Nova Interativa (20), que difunde, dentre tantas coisas, novos métodos ou técnicas voltadas à educação. Além disso, o acesso à plataforma poderia ser liberado a professores do ensino médio, capacitados através de um curso de extensão, para que dessa forma, pudessem com autoria, usar a ferramenta em suas atividades pedagógicas.

Por fim, reitera-se o potencial educacional do projeto em questão. Se bem utilizado, pode lidar com qualquer tópico científico analítico, no qual a imaginação do criador é o principal limitador. Quando difundido, levará aos usuários um potencial ativo de construção de conhecimento, no qual tanto professores quanto alunos podem criar e compartilhar suas simulações, gravações e resultados que, aos poucos, preencheram um acervo científico vasto e replicável.

REFERÊNCIAS

1. **Silva, Edson da.** *A importância das atividades experimentais na educação.* 2017. p. 47.

2. **Macedo, Danilo Fogaça de.** *O uso de simulação baseada em hipervídeo como recurso de ensino e aprendizagem de botânica.* Universidade de São Paulo. São Paulo : s.n., 2018. Dissertação (Mestrado em Ensino de Biologia) - Ensino de Ciências (Física, Química e Biologia).

3. **Bozzi, Lígia D'Ávila.** *Proposta metodológica para construção de simuladores experimentais baseados em hipervídeos.* Instituto de Química, Universidade de São Paulo. São Paulo : s.n., 2018. Dissertação (Mestrado em Química).

4. **Google.** Google Apps Script. *Google Developers.* [Online] [Citado em: 24 de 07 de 2021.] https://developers.google.com/apps-script.

5. **Django Software Foundation.** Django Documentation 3.2. *Django.* [Online] [Citado em: 24 de 07 de 2021.] https://docs.djangoproject.com/en/3.2/.

6. **Hipp, D. Richard.** SQLite Documentation. *SQLite.* [Online] [Citado em: 24 de 07 de 2021.] https://www.sqlite.org/docs.html.

7. **PostgreSQL Global Development Group.** PostgreSQL Documentation. *PostgreSQL.* [Online] [Citado em: 24 de 07 de 2021.] https://www.postgresql.org/docs/.

8. **Mark Otto, Jacob Thornton.** Bootstrap Documentation 4.6. *Bootstrap.* [Online] [Citado em: 24 de 07 de 2021.] https://getbootstrap.com/docs/4.6/getting-started/introduction/.

9. **Bootstrap Studio.** Bootstrap Studio Documentation. *Bootstrap Studio.* [Online] [Citado em: 24 de 07 de 2021.] https://bootstrapstudio.io/docs/.

10. **Korobov, Mikhail.** django-widget-tweaks 1.4.8. *PyPi.* [Online] [Citado em: 24 de 07 de 2021.] https://pypi.org/project/django-widget-tweaks/.

11. **James Lindenbaum, Orion Henry, Adam Wiggins.** Getting Started on Heroku with Python. *Heroku.* [Online] [Citado em: 24 de 07 de 2021.] https://devcenter.heroku.com/articles/getting-started-with-python.

12. **Tofolo, Gustavo Valencio.** App Name - Entrar. *App Name.* [Online] [Citado em: 30 de 07 de 2021.] https://ic-deploy.herokuapp.com/account/login/.

13. **Google.** YouTube Player API Reference for iframe Embeds. *Google Developers.* [Online] [Citado em: 24 de 07 de 2021.] https://developers.google.com/youtube/iframe_api_reference?hl=pt-br.

14. **Jazzband.** jazzband/django-taggit. *Github.* [Online] [Citado em: 24 de 07 de 2021.] https://github.com/jazzband/django-taggit.

15. **yairEO.** yairEO/Tagify. *Github.* [Online] [Citado em: 24 de 07 de 2021.] https://github.com/yairEO/tagify.

16. **Jetbrains.** Saiba mais sobre o PyCharm. *Pycharm.* [Online] [Citado em: 24 de 07 de 2021.] https://www.jetbrains.com/pt-br/pycharm/learn/.

17. **Pinheiro, Fagner.** Entendendo o MTV do Django. *TreinaWeb.* [Online] [Citado em: 28 de 07 de 2021.] https://www.treinaweb.com.br/blog/entendendo-o-mtv-do-django.

18. **Tofolo, Gustavo Valencio.** tofolo17/SIDjango. *Github.* [Online] [Citado em: 30 de 07 de 2021.] https://github.com/tofolo17/SIDjango.

19. —. tofolo17/SIFiles. *Github.* [Online] [Citado em: 30 de 07 de 2021.] https://github.com/tofolo17/SIFiles.

20. **Química Nova Interativa. [Online] [Citado em: 30 de 07 de 2021.] http://qnint.sbq.org.br/novo/.**

21. Melé, Antonio. *Aprenda Django 3 com Exemplos.* s.l. : Novated Editora Ltda., 2020. 978-65-86057-24-9.