Gerenciamento de espaço físico

Documento de arquitetura

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Motivo

Este documento descreve a arquitetura e implementação do sistema, apresentando os motivos, filosofia, e justificativas sobre as decisões arquiteturais.

# Objetivos arquiteturais e filosofia

O sistema será guiado pela performance, pois é essencial que o mesmo seja capaz de retornar uma boa disposição de turmas e salas em pouco tempo. Não é necessário gerar a melhor disposição possível, mas deve-se manter-se em um nível aceitável pelo cliente. O sistema deve ser flexível e sua arquitetura deve ser pouco acoplada, de forma que seja possível realizar manutenções ao longo do tempo, adicionando novas funcionalidades e integrações com outros sistemas internos do cliente.

# Suposições e dependências

A principal funcionalidade do sistema é a geração de disposição de turmas e salas. O algoritmo responsável pela geração deve ser flexível, de forma que se possa ser alterado a qualquer momento sem interferir no restante do sistema, visto que o time não possui experiência em soluções deste tipo. O sistema deve possuir suporte *offline* para funcionar a qualquer momento, visto que o sistema é essencial para a organização das questões acadêmicas do cliente. O sistema também deve ser capaz de ser instalado em qualquer servidor desejado pelo cliente, portanto, deve possuir o máximo de configuração via código para reduzir a quantidade de configurações manuais e facilitar as instalações nos servidores do cliente.

# Requisitos arquiteturais significantes

1. UC001 - Cadastrar dados acadêmicos
2. UC002 - Cadastrar espaço físico
3. UC008 - Consultar alocação de uma sala
4. UC016 – Relatório de alterações
5. UC012 - Solicitar alteração
6. UC005 - Gerar alocações

# Decisões, limitações e justificativas

* Separar o algoritmo de geração de disposição de turmas e salas em um projeto próprio (como uma biblioteca), para reduzir o acoplamento do código e facilitar a alteração (caso necessária);
* Não referenciar as classes do algoritmo diretamente; criar interfaces para tais classes de forma que, caso sejam alteradas, não afete o restante do sistema;
* Em caso de alterações manuais na disposição de turmas e salas, armazenar as mudanças apenas quando o usuário solicitar, e não em tempo real enquanto ele realiza as alterações;
* Armazenar um histórico das disposições geradas de forma que seja possível realizar um *rollback* para uma disposição anterior para caso o usuário deseje desfazer seus últimos ajustes. O histórico também é importante para caso o usuário queira retornar à disposição gerada por nosso algoritmo, de forma que não seja necessário executá-lo novamente;
* Implementar o sistema com a linguagem PHP orientado a objetos, pois o modelo lógico será baseado em objetos e os desenvolvedores possuem experiência nesta linguagem, assim como o *framework* Laravel;
* O banco de dados será MariaDB, devida a experiência do time;
* O sistema será modelado com o padrão de camadas, devida a decisão do cliente;
* Utilizar OpenUP por restrição do trabalho;
* Arquitetura cliente-servidor distribuido em camadas;
* Sistema executado em rede;

# Mecanismos arquiteturais

# Persistência

Necessária para armazenar os dados e apresentá-los ao usuário. Deve-se possuir autorização para acessá-la.

# Solução (código)

Necessária para iniciar a codificação. Deve possuir controle de versionamento e todos os membros do time devem poder acessá-la.

# Abstrações chaves

* *Repositórios*, responsável por gerenciar os dados do sistema;
* *Model-view-controller*, responsável por organizar a solução em um padrão flexível;
* *Biblioteca de geração de disposição física*, responsável por disponibilizar o (s) algoritmo (s) de geração de disposição física de forma não acoplada;
* *Aluno*, representando um aluno matriculado no curso, que fará parte de uma turma;
* *Grade* de alocação, que representa a grade integradora de toda a distribuição de salas, turmas e disciplinas;
* *Sala*, representando o espaço físico disponível na universidade;
* *Turma*, representando um conjunto de alunos organizados por um identificador numérico, sobre a qual deve estar conectada a uma ou mais disciplinas;
* *Funcionário*, representando algum funcionário da universidade que tenha acesso ao sistema; este mesmo é derivado para professor e funcionário da secretaria, onde o primeiro tem acesso à visualização de dados e requisição de alterações, enquanto o segundo tem acesso à visualização e manipulação de dados.

# *Framework* arquitetural

A arquitetura se baseará no padrão de camadas com MVC(*Model-view-controller*). O framework utilizado para a camada lógica (domínio) será Laravel.

# *Views* arquiteturais

* **Persistência**

Responsável por administrar o banco de dados, fornecendo acesso ao sistema de forma que seja possível realizar a comunicação entre o sistema e o banco de dados.

* **Domínio**

O domínio representa a lógica do negócio, sendo responsável por chamadas à camada de persistência, conversão e tratamento de dados.

* **Apresentação**

A camada de apresentação deve possuir toda a estrutura *web* do sistema (*views*), com *frameworks* *web* que utilizarmos, como *jQuery* e *bootstrap*. Também deve possuir todas as imagens utilizadas no sistema, como logos e ícones. As classes *controller* da camada de apresentação são responsáveis por realizar a chamada dos serviços disponíveis no domínio, de forma que o usuário possa realizar as ações desejadas.