Instituto de Computação
Departamento de Ciência da Computação
Universidade Federal da Bahia (UFBA) - Salvador, BA - Brasil

MATA62 - Engenharia de Software I
Prof. Eduardo Almeida

Alunos grupo 1: Igor Sobral (igor.sobral@ufba.br); João Lucas Lima de Melo (joaollm@ufba.br); Matheus Guimarães (guimaraes.matheus@ufba.br); Natan Moura (natan.moura@ufba.br)

- Sala Virtual do Grupo 1
- Especificações do trabalho parte 1 (pdf)
- Pasta Drive do Grupo
- Demais grupos da Disciplina
- Fase inicial do documento (Draft 1)
- Doc para Entrega da Parte 1
- Apresentação

Projeto Incremental de Engenharia de Software

1 - Identificar o conjunto de características arquiteturais da aplicação

- Auditabilidade
- Segurança
- Disponibilidade
- Autenticação
- Autorização
- Legalidade
- Privacidade
- Escalabilidade
- Elasticidade
- Acessibilidade
- Performance
- Recuperabilidade
- Suportabilidade
- Robustez
- Atualizabilidade
- Usabilidade

2 - Definir a priorização do conjunto de características arquiteturais da aplicação com justificativa

Crítico:

- **Auditabilidade**
  - Solicitado como aspecto crítico da aplicação, já que as operações de validação devem ser mantidas para consulta até quarenta anos depois.

- **Segurança**
    - As informações dos diplomas devem ser mantidas de forma segura e com controle de acesso.
- **Disponibilidade**
    - O sistema deve estar disponível de segunda a sexta em horário comercial.
- **Autenticação**
    - Para acessar o sistema, os usuários devem estar logados com credenciais válidas.
- **Autorização**
    - Deve haver diferenciação entre o acesso dos usuários a certas funcionalidades, como especificado.
- **Legalidade**
    - A emissão de diplomas e a confidencialidade das informações são aspectos que devem seguir à risca as leis vigentes.
- **Privacidade**
    - Pela sensibilidade dos dados com os quais a aplicação vai lidar, a privacidade dos mesmos é crítica.

Alta:

- **Escalabilidade**
    - O sistema deve suportar a adição de novos usuários já que estamos falando da inclusão de instituições parceiras.
- **Elasticidade**
    - O sistema deve ser capaz de suportar picos de uso, como por exemplo nos finais de ano, quando os processos de validação de diplomas são mais frequentes.
- **Acessibilidade**
    - A acessibilidade é importante para incluir os usuários com necessidades especiais.

Média:

- **Performance**
    - A performance apresenta média prioridade pois decidimos priorizar a capacidade de inclusão de novos usuários e o suporte a picos de acesso.
- **Recuperabilidade**
    - Como existe um *gap* temporal considerável entre os horários que a aplicação precisa estar disponível, o tempo necessário para recuperar a aplicação pode ser maior.
- **Suportabilidade**
    - O *tracking* de erros e o suporte técnico exigido pela aplicação não é tão crítico, pois espera-se um comportamento estável da plataforma, ainda que se faça necessário prover ferramentas para gerenciamento de erros por gestores do sistema.

Baixa:

- **Robustez**
    - Erros são aceitáveis e não tão críticos, portanto há baixa prioridade em robustez.
- **Atualizabilidade**
    - Permitir atualizações é desejável, porém não tão marcante para o escopo da aplicação.
- **Usabilidade**
    - O uso da plataforma se dá em um contexto fechado, com usuários de propósito específico e de acesso restrito, portanto a usabilidade não se faz aspecto marcante.

3. Identificar os componentes candidatos da arquitetura, como eles se relacionam e como os componentes incorporam as características arquiteturais da aplicação (textual e visual apresentação). É importante registrar e deixar implícitos as decisões tomadas nesse processo.

Componentes identificados:

- Componente para gerência de usuários
- Componente para gerência de cursos
- Componente para gerência de instituições
- Componente para lidar com acessos ao sistema (autorização e validação)
- Componente para gerência de diplomas
- Componente para auditoria do sistema
- Componente para auditoria de diplomas

Definições para a modelagem:

1. **Dirigente institucional** cadastra instituição parceira. **-> componente de gerência de instituições**
2. **Diretor** atualiza os dados da instituição parceira. **-> componente de gerência de instituições**
3. **Diretor** cadastra, exclui e altera usuários da instituição parceira (dirigente institucional, diretor, funcionário) **-> componente de gerência de instituições**
4. **Diretor** consulta usuários que possuem acesso ao sistema **-> componente de gerência de usuários**
5. **Funcionários** da instituição parceira consultam, cadastram, excluem e alteram cursos da instituição parceira. **-> componente de gerência de cursos**
6. **Funcionários** consulta cursos ofertados pela instituição **-> componente de gerência de cursos**
7. **Funcionários** da instituição parceira valida diplomas. **-> componente de gerência de diplomas**
8. **Usuários** realizam login e logout do sistema, além de recuperar senha. **-> componente de acesso ao sistema**
9. **Superintendentes** cadastram, alteram instituições validadoras de diploma. **-> componente de gerência de instituições**
10. **Superintendentes** liberam o acesso de instituições parceiras ao sistema de validação. **-> componente de gerência de instituições**
11. **Superintendentes** consultam instituições parceiras que têm acesso ao sistema de validação.**-> componente de gerência de instituições**
12. **Superintendentes** cadastram, consultam, alteram e excluem usuários para acesso ao sistema de validação, dirigente, superintendente, coordenador CARE e funcionário. **-> componente de gerência de usuários**
13. **Coordenador do CARE** visualiza as ações do superintendente, mas não operacionaliza. **-> componente de auditoria do sistema e de diplomas**

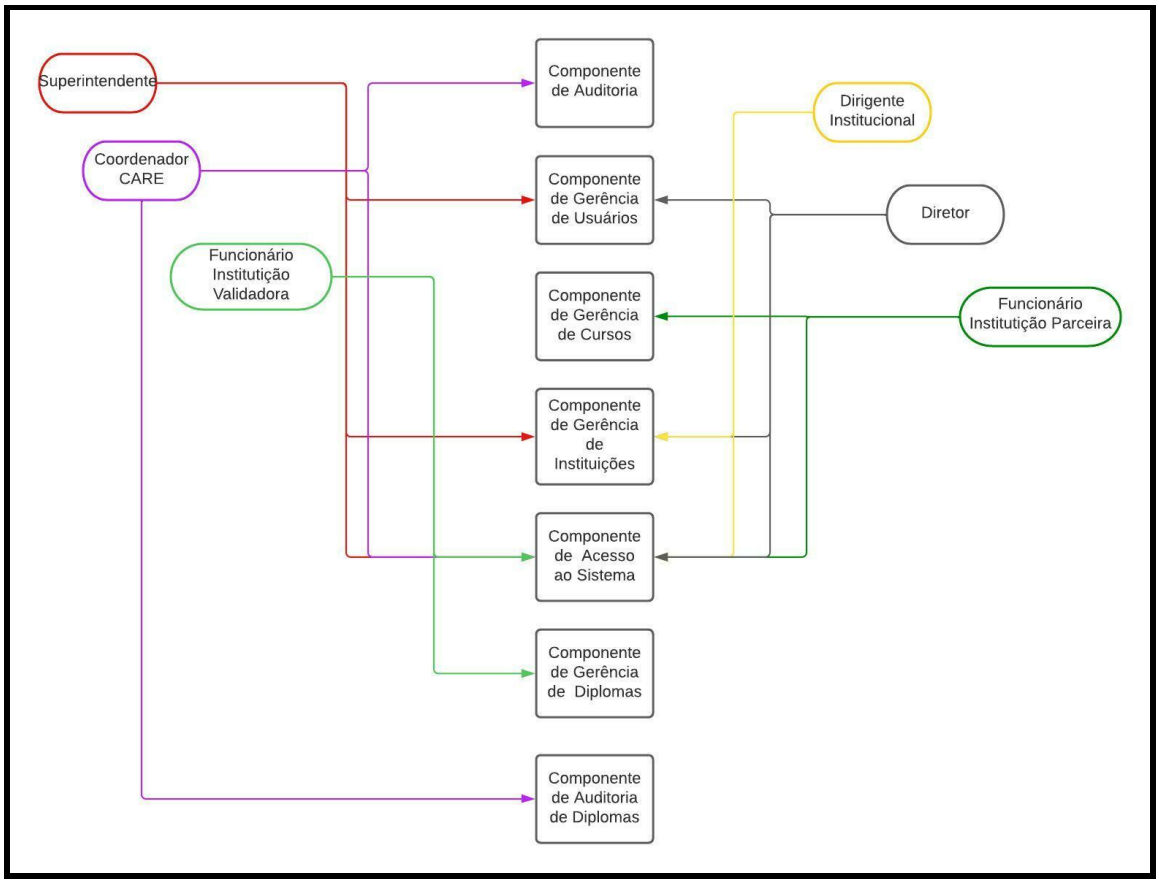

Figura 1

A figura 1 apresenta a relação que identificamos dos usuários do sistema com cada componente, essas relações foram estabelecidas com base nos requisitos da aplicação. A ideia desta representação é simplesmente indicar de quais componentes as ações do usuário seriam mapeadas, como por exemplo, ao gerenciar um usuário o superintendente solicita indiretamente recursos providos pela componente de gerência de usuários.

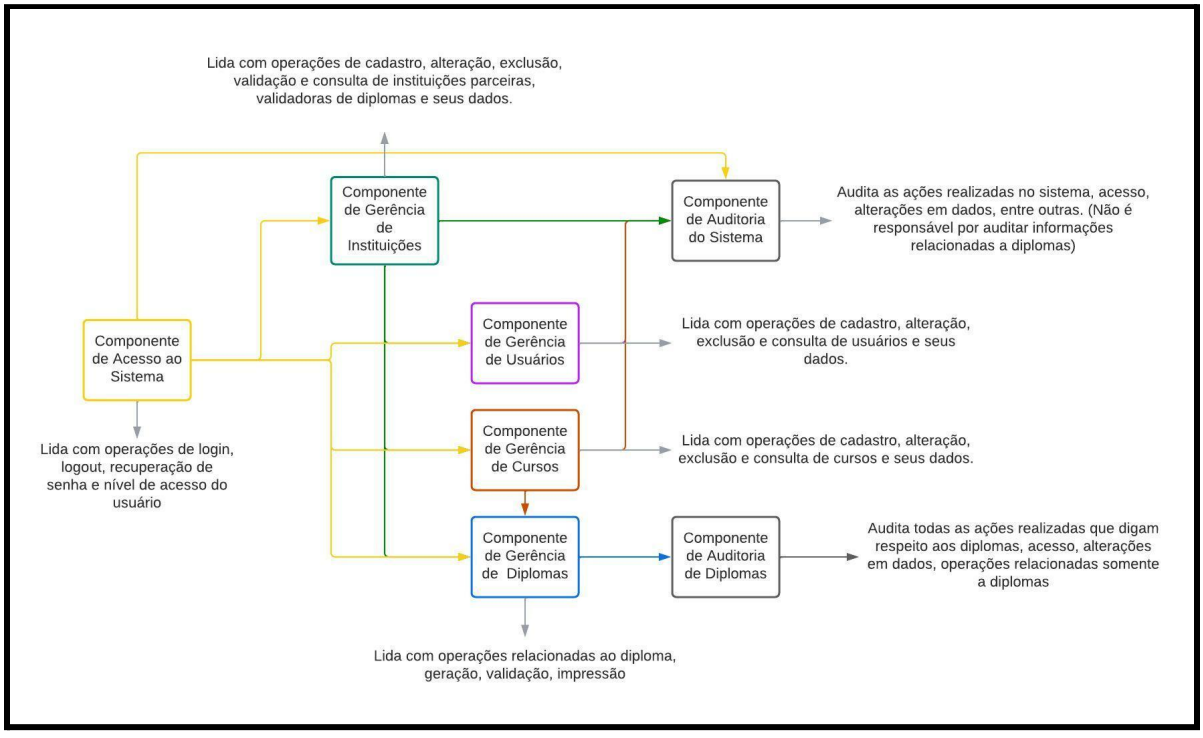

Figura 2

A figura 2 apresenta os componentes do sistema e suas relações. Primeiramente, decidimos agrupar os usuários das instituições e as instituições em componentes únicos, definindo o nível de acesso e a qual instituição se refere por regras de negócio, pois muitas operações são similares e a diferença simplesmente é dada pelo usuário que faz a operação e a qual instituição se refere (incluindo o tipo da instituição).

Os acessos ao sistema e a validação desses acessos são controlados pela componente de acesso ao sistema, que interage com as outras componentes de modo a validar as ações do usuário e permitir as requisições. Esta componente é o reflexo da característica arquitetural de **segurança**, **autenticação**, **autorização**, **privacidade** e **legalidade**.

Como **auditoria** é um aspecto crucial para a aplicação, decidimos criar uma componente que audita os trâmites realizados durante o uso do sistema, como acesso, edição de dados, etc (isso permite *tracking* de erros para a característica de **suportabilidade**) e uma componente de auditoria específica para o processo de validação de diplomas (guardando as informações do processo de validação de diplomas por até 40 anos), por isso estas componentes estão ligadas a todas as outras (se somarmos as conexões).

Como definimos **escalabilidade** e **elasticidade** de usuários como algo importante, decidimos tomar como base o **particionamento de domínio**, pois este pode ser aprimorado futuramente para uma **arquitetura de microserviços**, que acreditamos se encaixar mais com o propósito da nossa aplicação (já que as ações de cada usuário são bem definidas e estas poderiam ser diretamente direcionadas para o serviço referente). Além disso, com a arquitetura de microserviços, se um serviço cair, o sistema continua operante para os outros tipos de serviços, permitindo que a aplicação esteja disponível, mesmo que parcialmente e dando tempo para correção do serviço (característica de **disponibilidade, robustez, recuperabilidade**). A ideia da evolução para a arquitetura de microserviços seria transformar cada componente em um serviço individual, comunicáveis através de mensagens e respeitando as ligações explícitas na figura 2, com isso espera-se melhor **performance** do sistema, já que haverá menos sobrecarga e melhor distribuição das funções.

* Modelagens finais (em: link)