# Sistema de Integralização Remoto


| ***Nomes*** | ***RAs*** |
|---|---|
| *João Renato D. do Sacramento* | *061740* |
| *Thiago Pastore* | *082925* |
| *Jonathan Nunes Boilesen* | *083682* |

# Índice

# Introdução

Neste documento apresentaremos o projeto, modelagem UML, descrição da implementação e testes realizados, além de um pequeno tutorial para obter o código e instalar o mesmo.

# Descrição do Problema

Este projeto visa solucionar o problema de integralização enfrentado pela Diretoria Acadêmica da Universidade Estadual de Campinas (DAC - UNICAMP). Cada aluno da UNICAMP possui um histórico que contém todas as disciplinas cursadas pelo mesmo nesta instituição. Tais disciplinas são listadas anualmente nos catálogos de graduação. Este catálogo contém todas as disciplinas que um aluno de determinado curso deve cursar para conseguir seu diploma.

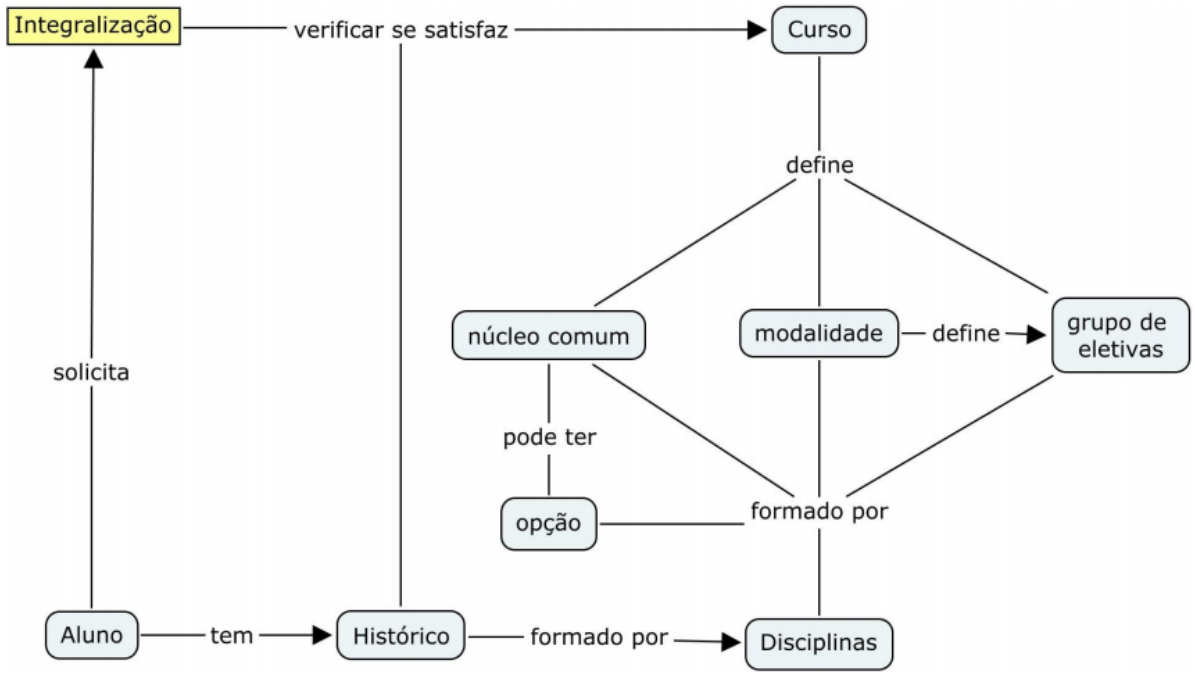
As disciplinas podem ser divididas em conjuntos[1], não disjuntos sendo que alguns podem ser vazios de acordo com cada curso, tais como: obrigatórias do núcleo comum, eletivas do núcleo comum, obrigatórias da modalidade, eletivas da modalidade, obrigatórias da opção, eletivas da opção e eletivas gerais.

O problema da integralização consiste em, dados um histórico e um catálogo de graduação, averiguar se determinado histórico cumpre o catálogo respectivo. Caso ainda existam disciplinas pendentes, o sistema deve alocar tais disciplinas de maneira que o aluno esteja formado no menor tempo possível. Computacionalmente um algoritmo de busca exaustiva tem complexidade exponencial fazendo com que tal implementação não seja viável uma vez que a UNICAMP possui mais de 10 mil alunos e deve rodar a integralização diariamente.

Desta maneira, o sistema projetado deve encontrar uma solução diferente (por exemplo: técnicas de programação dinâmica) para o problema, além de fazer uso de recursos diversos (computação distribuída) para torná-lo viável e mais eficiente.

### Diagrama Conceitual[2]

[1] Vanini, Fernando - http://www.ic.unicamp.br/~vanini/mc857/Projeto.pdf
[2] - Vanini, Fernando - http://www.ic.unicamp.br/~vanini/mc857/Projeto.pdf


Integralização
verificar se satisfaz
Curso
define
solicita
núcleo comum
modalidade
define
grupo de eletivas
pode ter
opção
formado por
Aluno
tem
Histórico
formado por
Disciplinas

# Descrição do(s) Ator(es)

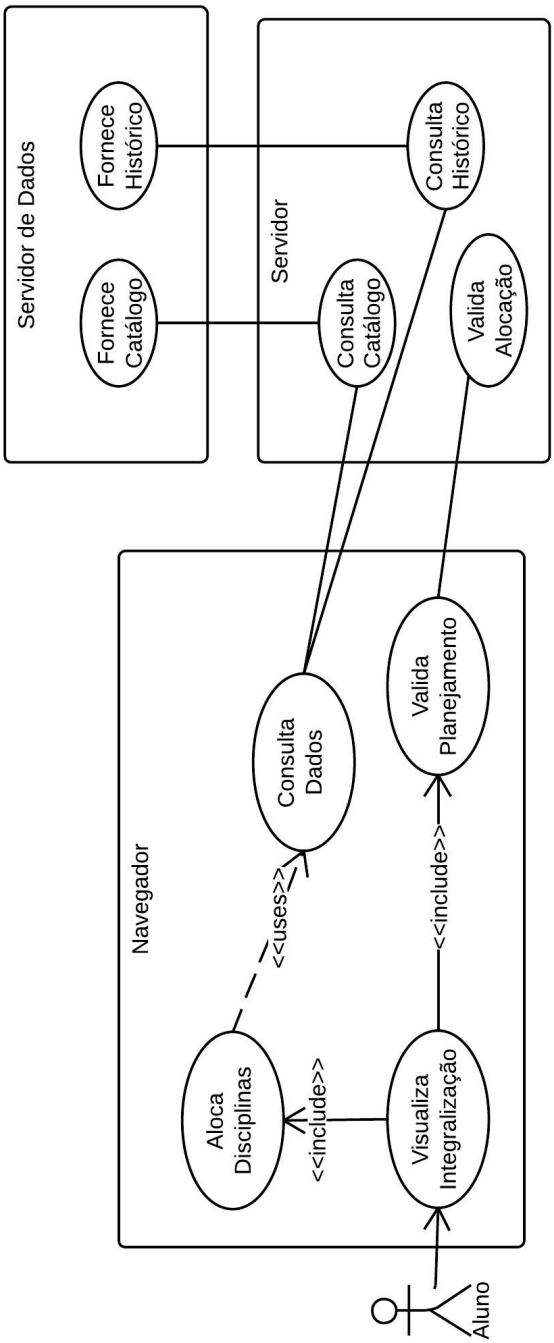
Nesta seção descreveremos o(s) ator(es) do sistema em desenvolvimento. Tal(is) ator(es) é(são) responsável(is) pela interação com o sistema. Devido as características do problemas apenas 1 ator se mostrou necessário até agora. Segue abaixo a descrição deste ator.

| | |
|---|---|
| **Ator** | Aluno |
| **Descrição** | Usuário já cadastrado e logado no sistema da DAC que possui um curso de graduação ou pós-graduação em andamento. |

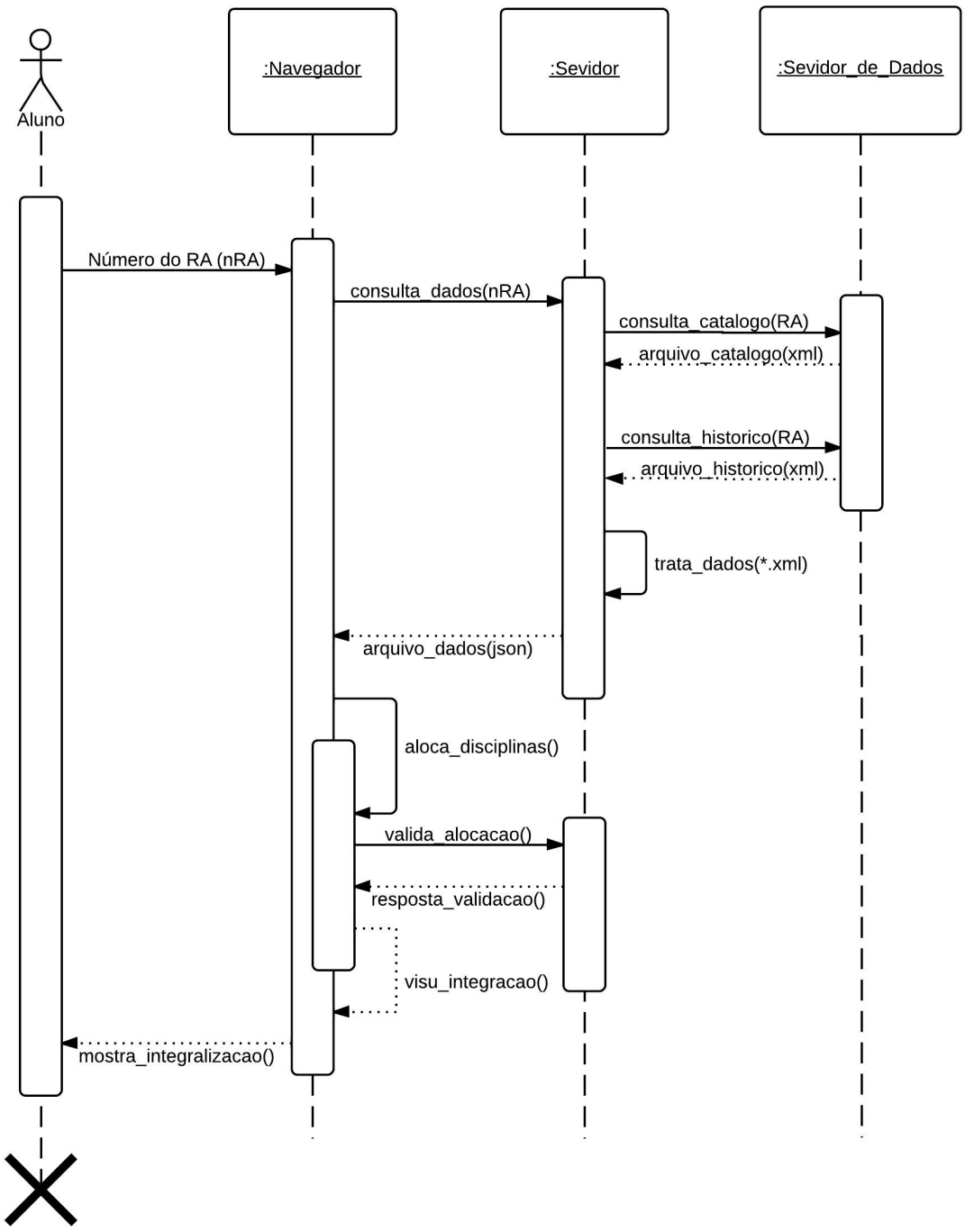
# Descrição do(s) Caso(s) de Uso

| | |
|---|---|
| **Nome** | Visualizar Integralização |
| **Descrição** | O ator deseja ver sua integralização e, caso seja necessário, a projeção de disciplinas a serem cursadas. |
| **Precondição** | Estar “logado” no site da DAC. |
| **Fluxo Básico** | 1. O usuário fornece seu número de registro acadêmico (RA);<br>2. O sistema busca o catálogo e o histórico de disciplinas cursadas no servidor;<br>3. O sistema elimina as disciplinas cursadas e aloca as disciplinas faltantes;<br>4. O sistema valida com o servidor a alocação feita;<br>5. O sistema mostra ao usuário sua integralização. |
| **Fluxos Alternativos** | No passo 2: se o sistema não consegue buscar o catálogo ou o histórico do usuário uma mensagem de erro é mostrada ao usuário.<br><br>No passo 4: caso o sistema perca a conexão com o servidor durante a validação, a integralização é mostrada ao usuário com uma mensagem de erro na validação. |

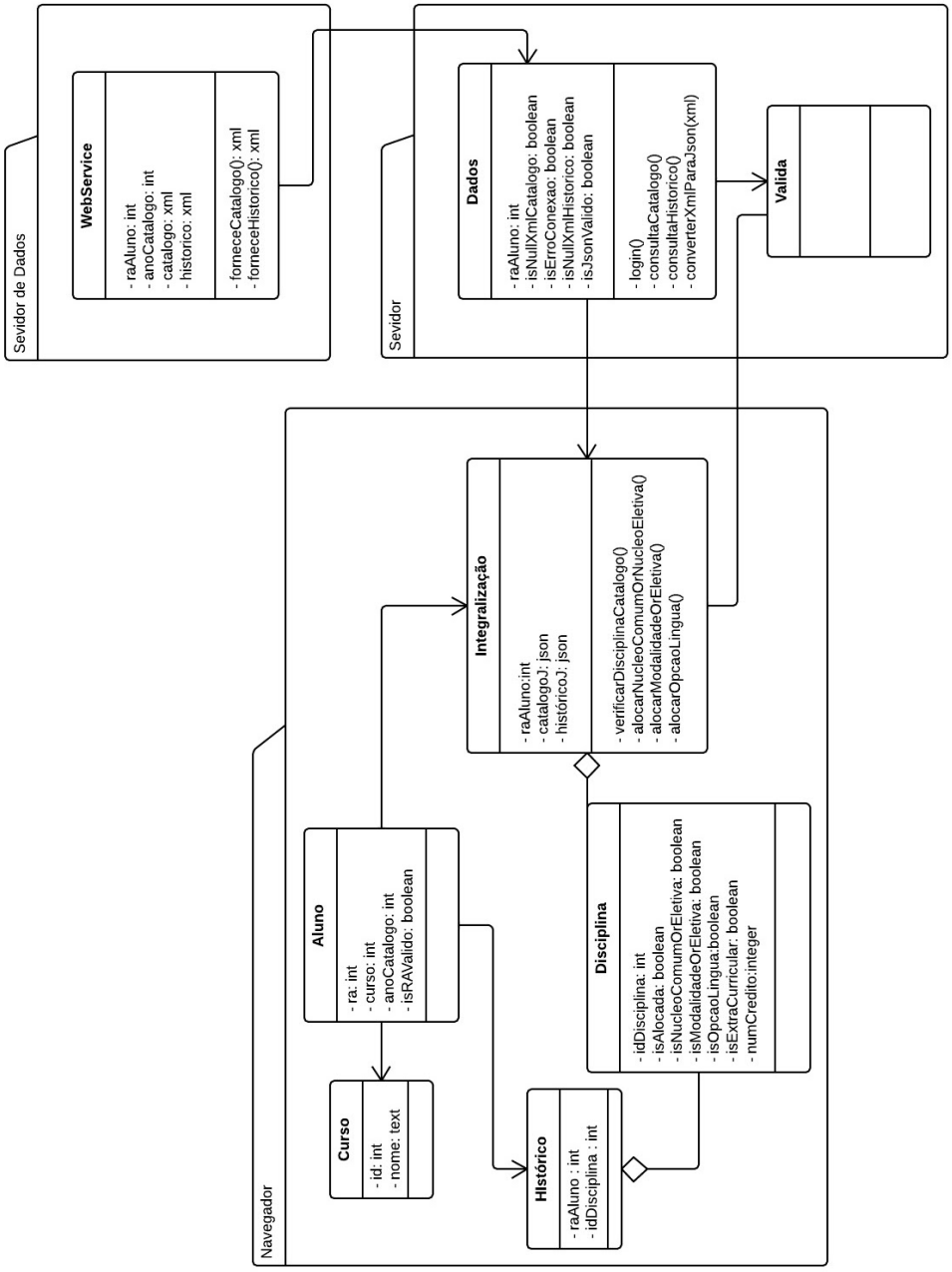
# Diagrama de Caso(s) de Uso



# Diagrama de Sequências



# Diagrama de Classes



# Descrição da solução implementada

Nossa solução contempla um pequeno servidor PHP que busca no servidor principal os dados em formato xml e faz um “parsing” para uma estrutura json que é enviado ao cliente. No cliente, através de javascript, utilizamos busca exaustiva juntamente com programação dinâmica, conforme sugerido pelo professor nos slides de descrição do projeto, para chegar na solução final. Primeiramente eliminamos todas as disciplinas obrigatória seja da cadeia principal ou de uma modalidade e depois geramos todas as combinações possíveis para alocar as disciplinas eletivas nos diversos grupos possíveis. Ao final, é exibida na tela do navegador uma descrição das matérias que ainda devem ser cursadas ou uma mensagem de congratulações.

# Testes Realizados

## Metodologia

Devido ao atraso que enfrentamos durante a etapa de implementação, optamos por realizar testes funcionais com o objetivo de assegurar o correto funcionamento do software em situações mais próximas a realidade. O teste foi dividido em 3 fases, todas utilizando os dados fornecidos em conjunto com o servidor PYTHON. Nas fases 1 e 2 utilizamos todos os históricos existentes, ou seja, forma realizados, no mínimo, 41 testes. Na primeira fase, nosso objetivo foi encontrar problemas graves que levassem o software a um estado de incosistência (travamento ou fechamento inesperado). Na segunda fase, nosso objetivo era validar se o software apresentava erros óbvios nas funções que lidavam com a integralização. Na última fase selecionamos os históricos que apresentaram problemas nas etapas iniciais, juntamente com históricos que fossem representativos (apresentassem as mesmas características de um determinado conjunto de históricos) para validar se a solução proposta era correta. Nos baseamos nos resultados esperados fornecidos.

## Resultados: Fase 1

- **Casos testados:** todos os históricos disponíveis;
- **Falhas encontradas:** foram exibidos erros de conexão com o servidor quando o curso e o ra dos alunos não existia no servidor.
- **Soluções:** adicionamos tratamento as respostas do servidor.

## Resultados: Fase 2

- **Casos testados:** todos os históricos disponíveis;
- **Falhas encontradas e Soluções:** mais de 25 erros foram encontrados e o código foi alterado para a solução dos mesmos, após apresentação inicial para o professor encontramos mais uma série de condições que foram corrigidas (implementação da busca exaustiva e programação dinâmica).

### Resultados: Fase 3

- **Casos testados:** históricos problemáticos e representativos, segue uma lista com alguns deles: 020991, 039765, 041220, 048221, 090200;
- **Falhas encontradas e Soluções:** alguns erros foram encontrados e o código foi alterado para a solução dos mesmos.

## Instalação

### Obtendo o código-fonte

Primeiramente devemos realizar o download do software que está disponível em um repositório público no site: https://github.com/jboilesen/mc857. Para baixá-lo podemos entrar no site, através de um navegador, e clicar no botão “Download ZIP” no canto esquerdo inferior. Caso o software git esteja instalado em seu computador basta clonar o repositório com o endereço: https://github.com/jboilesen/mc857.git.

### Instalando o software

Descompacte o arquivo zip na pasta atual (pule esta etapa se estiver usando o git), e copie o conteúdo da subpasta MC857 para o seu servidor HTML (por exemplo: /opt/lampp/htdocs em ambientes GNU/Linux com lampp). Inicialize o servidor python contido na subpasta: server/servidorDeDadosMC857.py. Abra um navegador e vá para a url http://localhost/mc857/MC857/ e pronto.

# Bibliografia

1. Sommerville,  Ian - Engenharia de Software - Addison Wesley, 2003
2. Vanini, Fernando - http://www.ic.unicamp.br/~vanini/mc857/Projeto.pdf
3. Object Management Group - http://www.uml.org/