

# Fase 1 - Programação

A equipe deverá desenvolver um site que servirá de interface para controle dos gabaritos da OCI.

O site deverá integrar com um banco de dados SQL para armazenar os participantes, provas, escolas, e as leituras de gabaritos; incluindo notas.

O site também deverá realizar a leitura de um gabarito automaticamente. Para isso, será disponibilizado para as equipes uma biblioteca em C++ que recebe a imagem de um gabarito (path, arquivo, ou raw data) e retorna um struct com a leitura do gabarito. A equipe não precisará fazer o próprio algoritmo de leitura da imagem do gabarito.

Além disso, serão disponibilizados arquivos para testes: tabelas de informações de provas e participantes, e um .zip com imagens de gabaritos com informações correspondentes à tabela.

O back-end do site deverá rodar no sistema operacional Linux (Pode ser um container Docker).

As ferramentas e frameworks são de livre escolha da equipe.

## Features desejadas

- ☐ Elaboração e organização de um banco de dados para armazenamento de participantes, provas, e leituras de gabaritos (com suas respectivas notas).
- ☐ Front-end intuitivo e organizado:
  - ☐ Leitura de gabaritos:
    - ☐ Input de imagens por parte do usuário
    - ☐ Mostrar as informações da leitura do gabarito na página
    - ☐ Permitir edição da leitura antes do salvamento no banco de dados
  - ☐ Leitura de tabelas do banco de dados
  - ☐ Adição/Edição/Deleção de registros em tabelas do banco de dados
- ☐ Back-end eficiente e organizado:
  - ☐ Obrigatoriamente compatível com Linux.
  - ☐ Processamento das imagens do usuário no back-end.
    - ☐ Integração do back-end com a biblioteca disponibilizada.
    - ☐ Tratar e retornar uma nota de acordo com o ID da prova e os pesos das questões
  - ☐ Processamento das operações de adição/edição/deleção no back-end
- ☐ Documentação do projeto
- ☐ Desenvolvimento do projeto em um repositório do GitHub.

# Opcionais e diferenciais:

- Criatividade! Sinta-se livre para inventar ou adicionar o que quiser!
- Uma interface de usuário bonitinha ;)
- Boas práticas de UX e interação humano-computador
- Leitura de múltiplas imagens de uma vez
- Código bem organizado e documentado
- Código eficiente e elegante

## A Biblioteca

A biblioteca é feita em C++, compilada para plataforma Linux, e tem duas funções para leitura da imagem de um gabarito. A documentação básica é incluída no header.

A biblioteca (**libleitor.so**) depende de duas outras bibliotecas também incluídas no drive ([raylib](#) e [zxing](#)).

```
typedef struct {
    int erro;           // Código de erro da leitura.
                        // - 0: Não houve erro.
                        // - 1: Erro de leitura do código Aztec
                        // - 2: Imprecisão ou erro na identificação da área
de leitura
                        // - 3: Erro fatal durante a leitura.
    int id_prova;       // ID da prova (-1 se não foi possível identificar)
    int id_participante; // ID do participante (-1 se não foi possível
identificar)
    char* leitura;      // String com a leitura do gabarito
                        // - 0: questão em branco
                        // - ?: questão com mais de um item marcado
                        // - a, b, c, d, e, ...: o item marcado na questão
} Reading;

// EXEMPLO:
// erro: 0
// id_aluno: 10211291
// id_prova: 15
// leitura: abbaccdeea0ed?dd0ba

// Recebe: o path de uma imagem no sistema
// Retorna um struct Reading com a leitura realizada
Reading read_image_path(const char* path);

// Recebe:
// 1. extensão da imagem (ex: .png),
// 2. um array de chars correspondente aos dados (bytes) do arquivo
// 3. o tamanho do array em bytes
```

```
// Retorna um struct Reading com a leitura realizada  
Reading read_image_data(const char* file_type, const unsigned char*  
file_data, int file_data_size);
```

Arquivos disponíveis no [google drive](#)