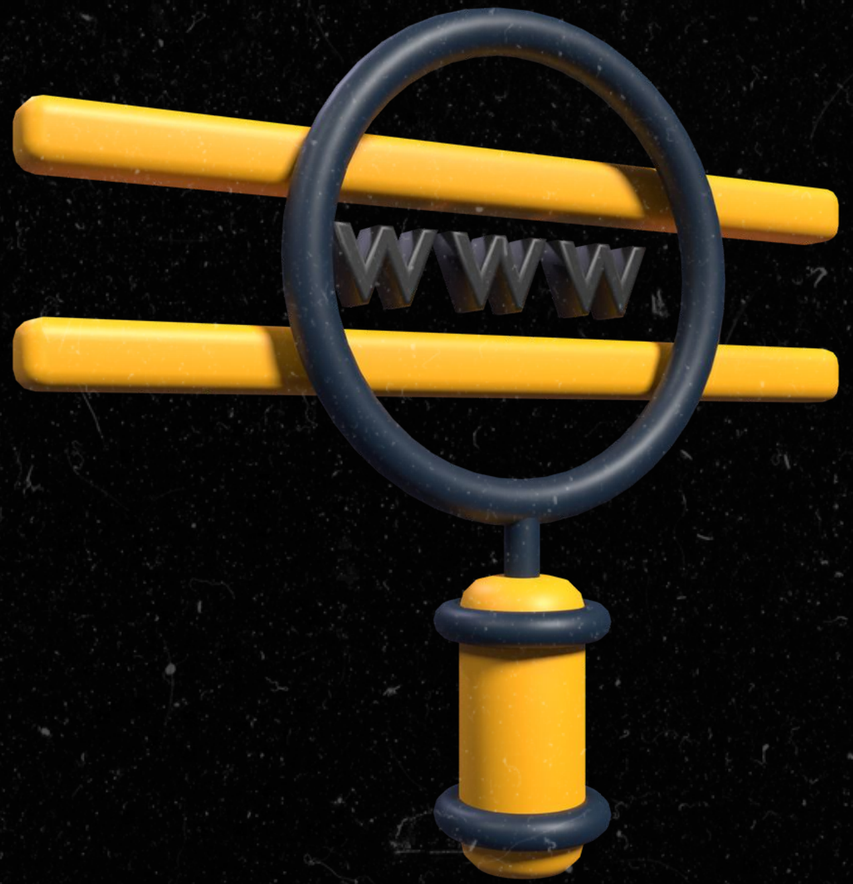
<Company> <Lesson>

Sistema para Restaurante Universitário

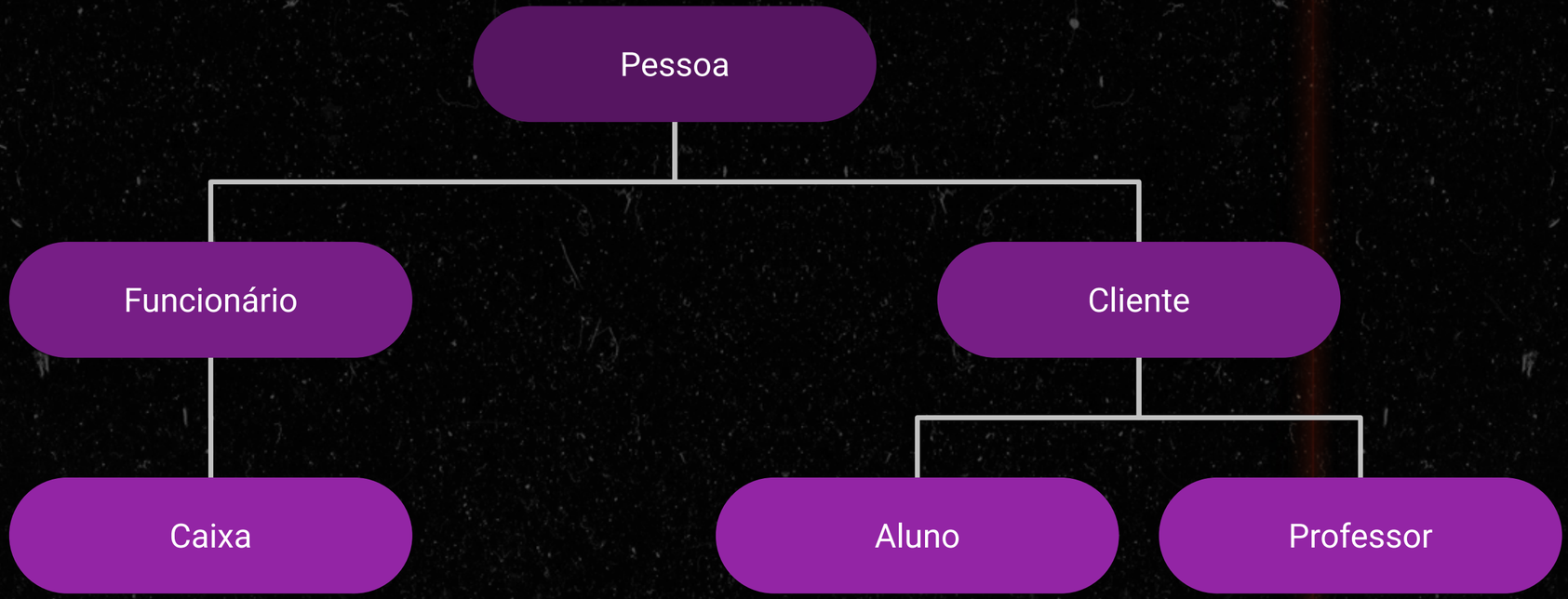
Ana Laura Soares Mello,
Larissa Roncali Faria,
João Pedro Braga da Silva,
Matheus Magalhães Caldeira Brant

Sumário

{01} Motivação e Modelagem

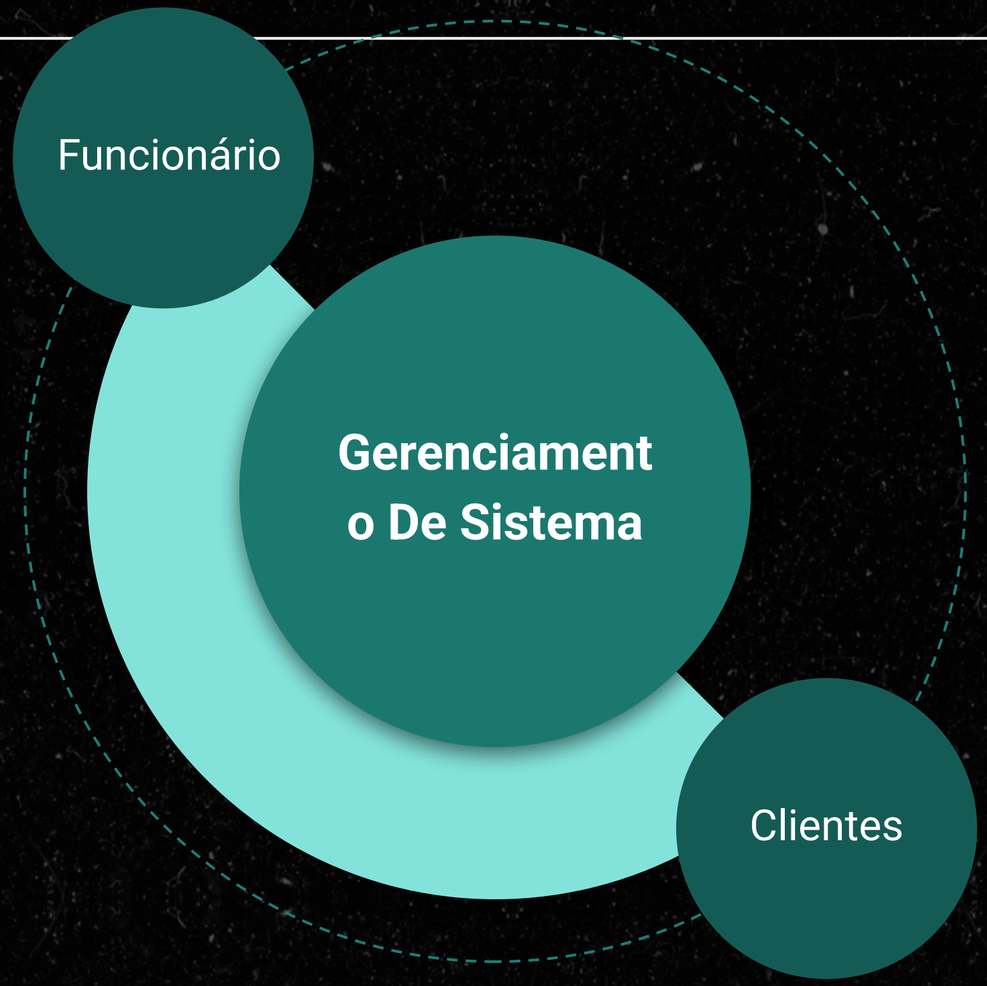


Com o crescimento das filas e o estresse associado, nossa proposta é criar um sistema que torna o pagamento mais rápido e eficiente.



Herança entre Classes

<Company> <Lesson>



Composição entre Classes

02 Código e Aplicações

Classe Cliente/Doxygen

```cpp
/**
 * @brief Registra um acesso (refeição), atualizando a data correspondente.
 * @param tipoDeRefeicao Um char indicando o tipo de refeição ('a' para almoço, 'j' para jantar).
 * @return Retorna um booleano indicando se a operação foi bem-sucedida.
 */
bool registrarAcesso(char tipoDeRefeicao);
```

Classe Funcionário/Excessões

```
void Funcionario::liberarRefeicao(Cliente* cliente) {
    try {
        if (cliente->getSaldo() < cliente->getValorRefeicao()) {
            throw ExcecaoSaldoInsuficiente();
        }

        // A lógica de verificação de data esta encapsulada dentro de cliente->registrarAcesso().
        bool sucesso = cliente->registrarAcesso(GerenciamentoDeSistema::_tipoDeRefeicao);

        if (sucesso) {
            std::cout << "Refeição liberada com sucesso pelo funcionário." << std::endl;
        }

    } catch (const ExcecaoSaldoInsuficiente& e) {
        std::cerr << e.what() << std::endl;
    }
}
```

Classe Aluno/Testes de Unidade

```cpp
/**
 * @class Aluno
 * @brief Representa um cliente do tipo Aluno, que possui um nível FUMP e um curso.
 * * Esta classe herda de Cliente e ajusta o valor da refeição com base no
 * nível socioeconômico do aluno (nível FUMP).
 */

class Aluno : public Cliente {
private:
    int _nivelFump;   ///< Nível socioeconômico do aluno, usado para definir o preço da refeição.
    std::string _curso; ///< O curso no qual o aluno está matriculado.
```

Classe Aluno/Testes de Unidade

```cpp
TEST_CASE("Criação de Aluno com nível FUMP válido e curso definido") {
    Aluno a("João", "12345678900", 3, "Engenharia Elétrica");

    CHECK(a.getNome() == "João");
    CHECK(a.getCpf() == "12345678900");
    CHECK(a.getCurso() == "Engenharia Elétrica");
    CHECK(a.getNivelFump() == 3);
    CHECK(a.getValorRefeicao() == doctest::Approx(1.0f));
}
```

Main.cpp/Interface com o usuário

```
switch (opcao) {
    case 1: {
        cout << "\n>> Selecione o tipo de usuário para cadastro:\n";
        cout << " [1] Aluno\n";
        cout << " [2] Professor\n";
        cout << " [3] Visitante\n";
        cout << " [4] Funcionário\n";
        cout << " [5] Caixa\n";
        cout << " [6] AdmFump\n";
        cout << "-------------------------------------\n";
        cout << "Tipo: ";
        cin >> opcao_cadastro;
        cout <<"\n"<<endl;
        cout <<"\n"<<endl;
```

{Conclusão}

Esperamos uma significativa redução nas filas nos Restaurantes Universitários, uma vez que o tempo de espera para pagamentos será drasticamente reduzido.

{  }{Funcionários}

A implementação do sistema reduz a necessidade de troco, aliviando a carga de trabalho dos funcionários da FUMP.

{  }{Estudantes}

A agilidade no pagamento melhora o fluxo de clientes no restaurante, proporcionando aos estudantes mais tempo para aproveitar o ambiente e o lazer universitário.