Documentação Técnica - MyLogin System

Client-Side vs Server-Side

Client-Side (Frontend)

O código executado no navegador do usuário inclui:

1. **Validações em JavaScript**:

```javascript
// Exemplo de validação client-side
function validatePassword(password) {
  const hasUpperCase = /[A-Z]/.test(password);
  const hasNumbers = /\d/.test(password);
  // Validação imediata sem requisição ao servidor
  return hasUpperCase && hasNumbers;
}
```

2. **Feedback Visual**:

```javascript
// Atualização da interface em tempo real
function showError(fieldId, error) {
  const errorDiv = document.getElementById(fieldId + "-error");
  errorDiv.textContent = error;
```

}

```

3. **Manipulação do DOM**:

```javascript
// Interação com elementos HTML
document.getElementById("loginForm").addEventListener("submit", function (e) {
  if (!validateForm()) {
    e.preventDefault(); // Impede envio do formulário se inválido
  }
});
```

Server-Side (Backend)

O código executado no servidor inclui:

1. **Validações em PHP**:

```php
// Exemplo de validação server-side
function validarSenha($senha) {
    if (strlen($senha) < 8) {
        return "Senha muito curta";
    }
    // Validação final com acesso ao banco de dados
```
```

    return true;
}
```

2. **Acesso ao Banco de Dados**:

```php
// Exemplo de query segura com prepared statement
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = ?");
$stmt->execute([$username]);
```

3. **Gerenciamento de Sessão**:

```php
// Controle de autenticação
session_start();
$_SESSION['user_id'] = $user['id'];
```

Estruturas de Controle

1. Condicionais

```php
// Exemplo de if/else para validação
if (empty($username)) {

```
    $erro = "Campo obrigatório";
} else if (strlen($username) < 3) {
    $erro = "Muito curto";
} else {
    // Processamento
}
```

2. Loops

```php
// Exemplo de foreach para processar dados
foreach ($validations as $field => $rules) {
    foreach ($rules as $rule) {
        // Validação de cada campo
    }
}
```

3. Switch

```php
// Exemplo de switch para tipos de erro
switch ($errorType) {
    case 'auth':
        $message = "Erro de autenticação";
        break;
```

```
    case 'validation':
        $message = "Erro de validação";
        break;
    default:
        $message = "Erro desconhecido";
}
```

Segurança

1. Proteção contra SQL Injection

- Uso de PDO com prepared statements
- Parâmetros sempre vinculados
- Nunca concatenação direta de strings SQL

2. Proteção XSS

- Escape de saída com htmlspecialchars()
- Validação de entrada
- Headers de segurança

3. CSRF Protection

- Tokens únicos por sessão
- Validação em cada requisição POST
- Renovação após uso

4. Senha

- Hash seguro com password_hash()
- Sal único por senha
- Nunca armazenamento em texto plano

Manipulação de Dados

1. Arrays

```php
// Exemplo de manipulação de array
$userdata = [
    'username' => $username,
    'email' => $email,
    'created_at' => date('Y-m-d H:i:s')
];
```

2. Banco de Dados

```php
// Exemplo de CRUD
function createUser($data) {
    $sql = "INSERT INTO users (username, email) VALUES (?, ?)";
    return $pdo->prepare($sql)->execute([$data['username'], $data['email']]);
```

}
```

## Sessões e Cookies

### 1. Sessão

```php
// Início seguro de sessão
session_start();
session_regenerate_id(true);
```

2. Cookies

```php
// Cookie seguro
setcookie('remember_me', $token, [
    'expires' => time() + 30*24*60*60,
    'path' => '/',
    'secure' => true,
    'httponly' => true,
    'samesite' => 'Strict'
]);
```

Tratamento de Erros
```

1. Try-Catch

```php
try {
    // Operação arriscada
    $result = operacaoPerigosa();
} catch (Exception $e) {
    // Log do erro
    logError($e->getMessage());
    // Mensagem amigável
    return "Ocorreu um erro. Tente novamente.";
}
```

2. Logs

```php
function logError($message) {
    $date = date('Y-m-d H:i:s');
    $log = "[$date] $message\n";
    file_put_contents('error.log', $log, FILE_APPEND);
}
```

Boas Práticas

1. **Separação de Responsabilidades**

   - Código PHP separado do HTML
   - JavaScript em arquivos separados
   - CSS em arquivos separados

2. **Comentários e Documentação**

   - Comentários explicativos
   - PHPDoc para funções
   - Documentação técnica atualizada

3. **Código Limpo**

   - Nomes descritivos
   - Funções pequenas e focadas
   - Evitar repetição de código

4. **Performance**
   - Queries otimizadas
   - Cache quando apropriado
   - Minimização de requisições