UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GBC064 - Engenharia de Software

Projeto da Disciplina (4ª Entrega)

Gabriel Riquieri Campos - 11911BCC030
João Batista De Sousa Paula - 11911BCC008
João Antônio Nardini - 11811BCC028
Nathan Augusto - 11811BCC046
Cleiton Kennedy de Morais Filho - 11021EMT018

1) Revisão da Aplicação da Metodologia Scrum

Foi utilizada a metodologia Scrum para o desenvolvimento do projeto. A escolha por utilizar essa metodologia veio do anseio de proximidade com o cliente, correção de rumo com o mínimo de dano e retrabalho possível ao longo do projeto e a possibilidade de existirem entregas parciais.

Membros assumiram papéis da metodologia, sendo o Gabriel Riquieri scrum master e product owner, enquanto João Batista, João Nardini, Nathan Augusto e Cleyton Filho assumiram o papel de equipe de desenvolvimento. As sprints foram definidas com um período de duas semanas, dez dias úteis. Foi utilizado o Trello para manutenção do backlog.

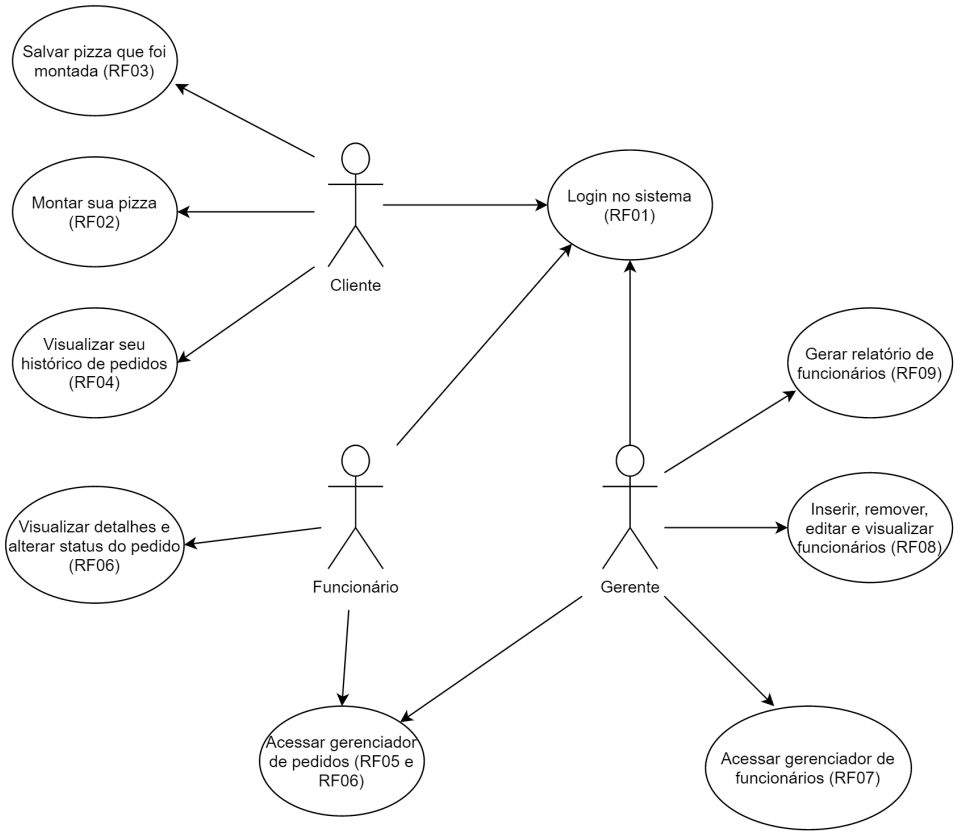
Ademais, foram cumpridas as cerimônias propostas pela metodologia. Ao início de cada sprint foi realizada a planning, reunião a qual o product owner administrava o backlog e o backlog da sprint com base em prioridade. Por motivos acadêmicos a daily foi realizada apenas uma vez por semana e eram apresentados de maneira simples o andamento das tarefas. Ao final da sprint foram realizadas as review, reunião a qual as tarefas da sprint eram avaliadas e dadas como concluídas ou não.

Com isso, observamos que no geral a organização por cards no Trello funcionou muito bem. A qualquer momento algum membro poderia entrar e ter uma perspectiva geral do que estava sendo feito. O uso de checklists dentro de cada card foi útil para uma visão mais detalhada sobre o andamento daquela tarefa. Na grande maioria das cerimônias todos os membros conseguiram comparecer. Também podemos visualizar as entregas parciais como as apresentações feitas ao professor e a turma ao longo da disciplina.

2) Revisão dos Artefatos Produzidos

2.1)Diagrama de Requisitos

Os requisitos de nossa aplicação foram colocados de acordo com o diagrama a seguir:



Foi criado 3 áreas separadas para cada “tipo” de pessoa utilizando a aplicação:

Cliente: Pessoa a qual realiza o pedido, afim de comprar o produto oferecido pela pizzaria

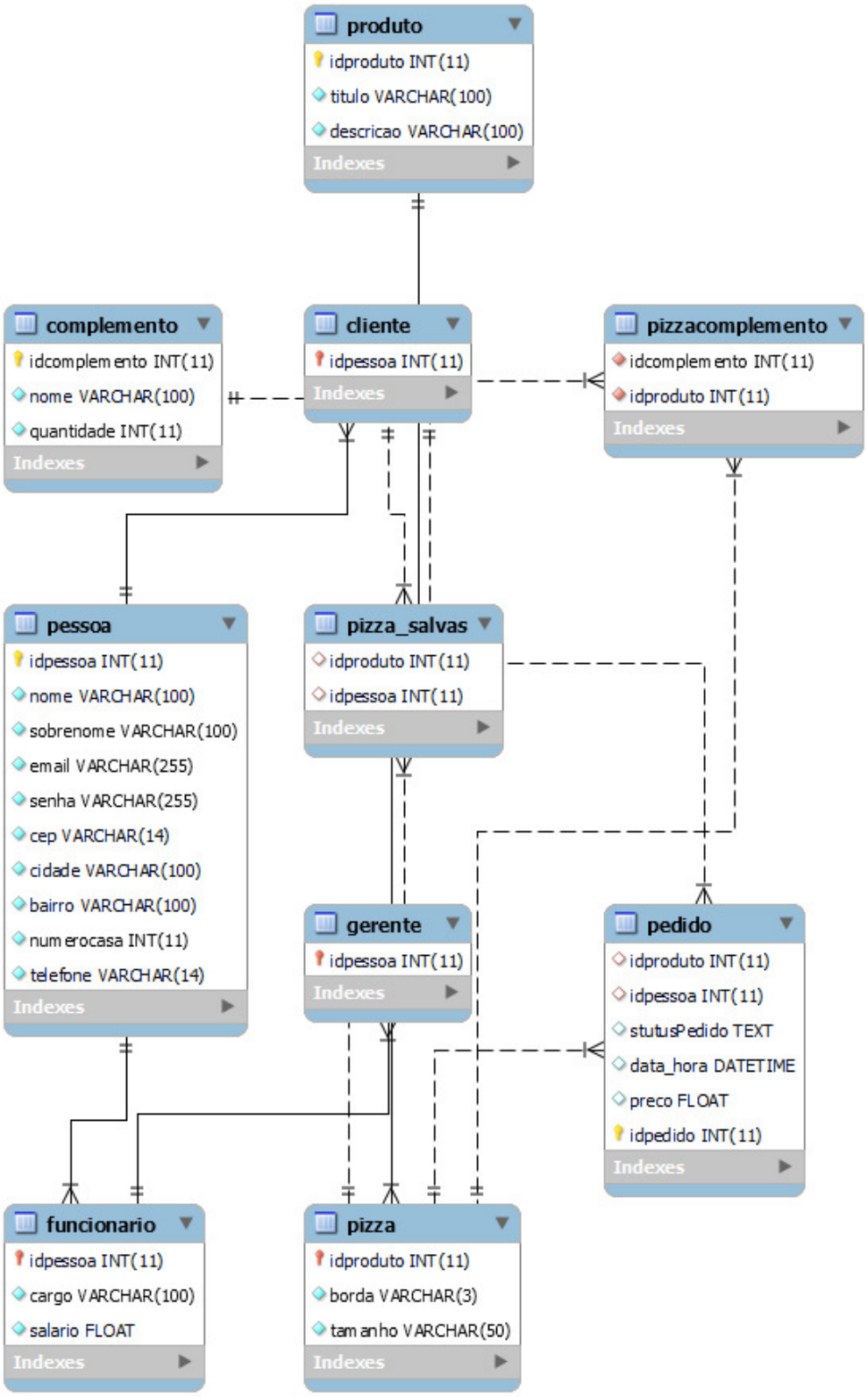
Funcionário: Pessoa responsável por realizar a checagem de status e outras coisas relacionadas ao pedido do cliente.

Gerente: Pessoa responsável por gerir as condições da aplicação em si (Visualizar tanto a área dos funcionários quanto a dos clientes) .

2.2)Banco de dados

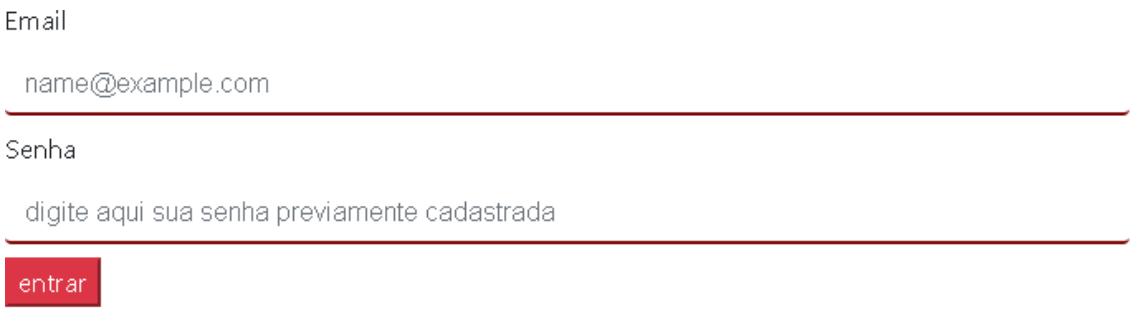
O banco de dados da aplicação conta com as entidades “*Pessoa*”, “*Cliente*”, “*Funcionário*”, “*Gerente*”, “*Produto*”, “*Pizza*”, “*Complemento*” e “*Pedido*”, sendo as 4 primeiras utilizadas para identificação pessoal da

pessoa a qual estamos referindo, e as demais para identificação do pedido em questão.



2.3)Navegabilidade

Para uma melhor experiência do cliente, é necessário que este realize o login (ou cadastre-se, caso não tenha um cadastro prévio) para visualizar todas as funcionalidades disponíveis.

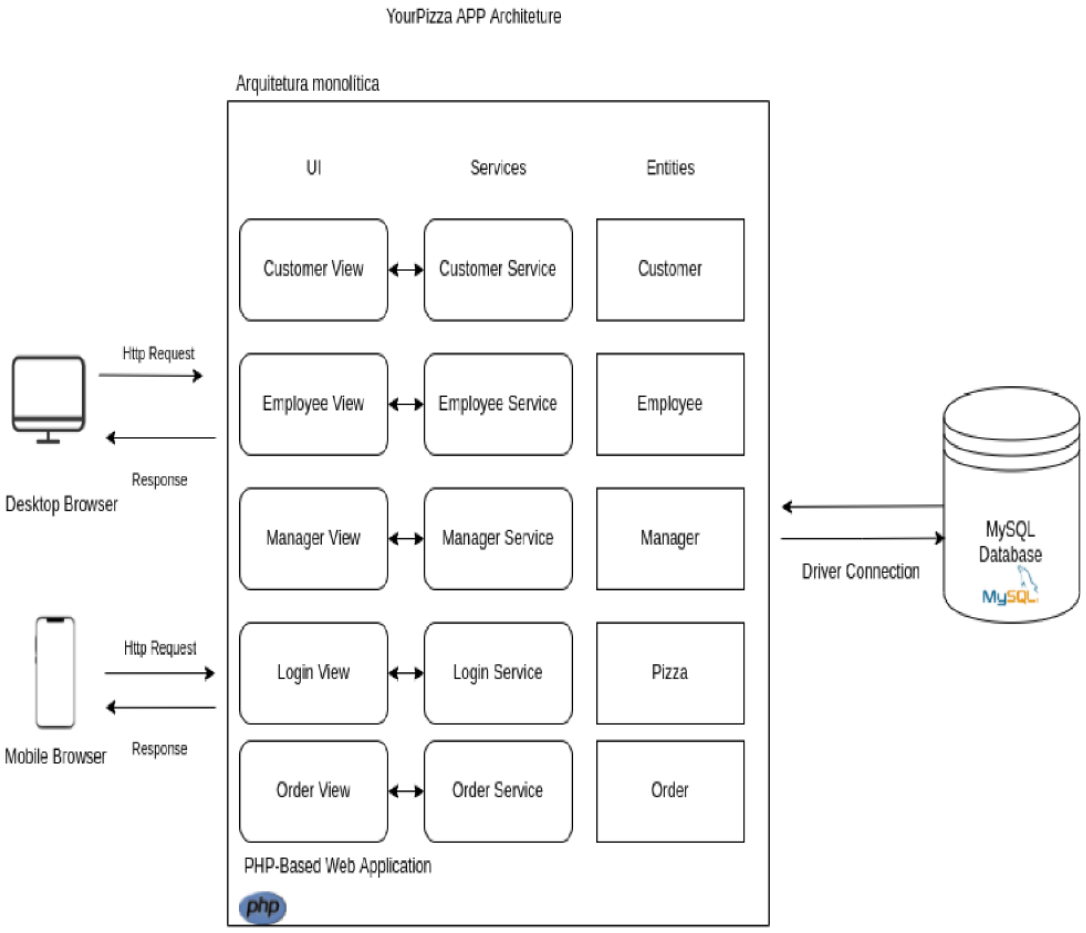


A navegabilidade da aplicação se baseia inteiramente nas abas disponíveis no cabeçalho da página.

2.4)Arquitetura

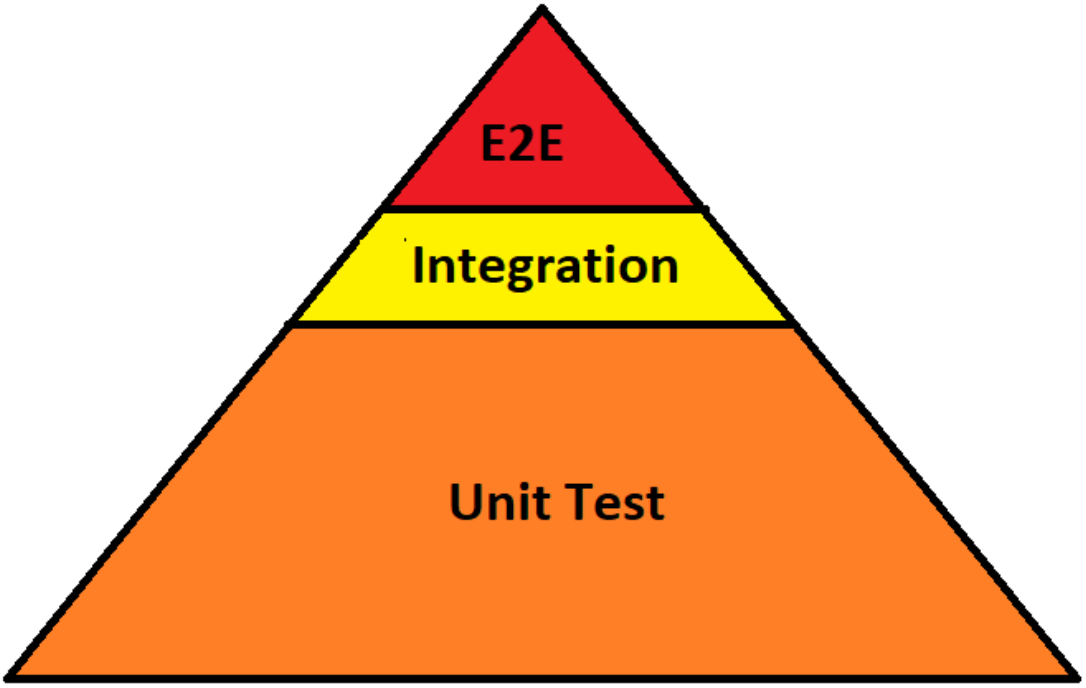
Nosso projeto consiste em uma aplicação que irá atender uma única pizzaria e seus clientes. Portanto, o investimento inicial deve ser baixo para que o investidor não sofra prejuízos relevantes e a escalabilidade não é tão importante, visto que, o público alvo é limitado por região e poder aquisitivo.

Desta forma, foi escolhida a arquitetura monolítica para desenvolvimento das funcionalidades, visando uma robustez do sistema que permita o baixo investimento e uma manutenção barata



3) Testes Aplicados

Abaixo podemos ver uma pirâmide com os principais testes em um processo de desenvolvimento de software.



End to End: Esses testes são um dos mais adequados para testar um sistema pois ele testa de ponta a ponta o sistema,simula um utilizador real desta forma testou-se front-end,back-end, requisitos,response etc… realmente muito útil, porém caro.

porque caro?

Porque você tem o software praticamente completo para realizá-lo em um container ou um ambiente de teste.

Integration: Esses testes servem para testar a utilização de funcionalidades em conjunto ou seja se está tudo nos conformes

Unit test: estes testes são teste que está responsável por testar um método/função individualmente ou seja um teste algoritmo, é barato.

Exemplo de alguns testes do nosso sistema

```php
<?php
require_once '../classe/Cliente.php';
class TesteCliente{
    public function testeCadastraCliente($nome,$sobrenome,$email,$senha,$cep,$cidade,$bairro,$numerocasa,
    $telefone){···
    }
    public function testeEditaCliente($nome, $sobrenome, $email, $cep, $cidade, $bairro, $numerocasa,
    $telefone,$id){···
    }
    public function testeBuscaEspecifica($id){···
    }


}
?>
```

```php
<?php
require_once '../classe/Complemento.php';
class TesteComplemento{
    public function testeCadastraComplemento($nome,$quantidade){···
    }
    public function testeBusca(){···
    }
    public function testedecrementar_quantidade_conplementos($idComplemento){···
    }
    public function testeBuscaPorId($id){···
    }

}
?>
```

```php
<?php
require_once '../classe/Funcionario.php';
class TesteFuncionario{
    public function testeCadastrar($nome, $sobrenome, $email, $senha, $cep, $cidade, $bairro, $numerocasa, $telefone, $cargo, $salario){ ···
    }
    public function testeBuscaFuncionario(){ ···
    }
    public function testeDeletarFuncionario($id){ ···
    }
    public function testeEditarFucionario($nome, $sobrenome, $email, $cep, $cidade, $bairro, $numerocasa, $telefone, $cargo, $salario,$id){ ···
    }

    public function testeBuscaPorIdFuncionario($id){ ···
    }
}
?>
```

```php
<?php
require_once '../classe/Logar.php';
class TesteLogin{
    public function TesteLoginFuncionario(String $email, string $senha){
        $f=new Logar();
        if($f->login_funcionario( $email,  $senha)){
            return true;

        }else{
            return false;
        }
    }
    public function testeLoginCliente(String $email, string $senha){
        $c=new Logar();
        if($c->login_cliente($email,$senha)){
            return true;
        }else{
            return false;
        }

    }
    public function testeLoginGerente(String $email, string $senha){
        $g=new Logar();
        if($g->login_gerente($email,$senha)){
            return true;

        }else{
            return false;
        }
    }
}
?>
```

phpcs: Request workspace/conf

4) Apresentação do Produto

Trailer:
https://www.youtube.com/watch?v=t0LgzCjoIPw&ab_channel=JBtecnologia

Demonstração Completa:

https://www.youtube.com/watch?v=N7-T4h9V54s&t=0s&ab_channel=JBtecnologia