Testes unitários

- Cada unidade do sistema é testada individualmente;
- Verifica se a aplicação funciona após sofrer alterações a nível de código;
- Facilita o trabalho de testar as várias unidades, funções ou componentes de um sistema, seja este simples ou complexo;
- Exemplo: Assegurar o tipo de retorno de uma função de um Controller do padrão MVC.

Testes de integração

- Módulos já previamente testados individualmente são agrupados e testados em conjunto;
- Realizados após a fase de testes unitários;
- Analisar a arquitetura do sistema através da presença ou ausência de falhas na integração entre unidades do sistema.

Testes de sistema

- Procuram validar os requisitos do sistema;
- Usualmente, o software é apenas um componente de um sistema maior, pelo que estes testes avaliam o sistema todo e detetam irregularidades entre os componentes do sistema;
- Validam o comportamento dos componentes face aos requisitos do sistema.

Testes de carga

- Verificam os níveis de performance do software;
- Identificam a capacidade operacional máxima das aplicações;
- Determinam se a infraestrutura do sistema está apta a correr a aplicação e o número de utilizadores que suporta em simultâneo;
- Resultam no refinamento de performance, se necessário, para assegurar a estabilidade do software antes de ser disponibilizado publicamente;
- Validam requisitos não funcionais.

Vantagens dos testes unitários

- Identificam código que pode ser reutilizado;
- Aumentam a confiança na manutenção de código;
- Detetam erros na fase inicial do projeto.

Desvantagens dos testes unitários

- Não detetam todos os erros;
- Os problemas não detetados podem ocorrer numa fase de teste de interfaces com o utilizador.

White Box Testing

- Técnica de teste unitário que testa a estrutura interna, design e código do software;
- Realizada por *testers*;
- Vantagens:
    - Otimiza o código;
    - Quando se conhece o código, é mais claro nidificar casos de teste.
- Desvantagens:
    - É necessário um *tester* qualificado, exige maior custo.

    - Não é fácil identificar todos os casos de teste necessários.

Black Box Testing

- Técnica de teste unitário que testa o comportamento externo dos componentes do software, à base de requisitos;
- Realizada por utilizadores;
- Vantagens:
    - O teste agrupa conjuntos de dados de entrada;
    - Não requer conhecimento interno de software.
- Desvantagens:
    - Os casos de teste averiguados podem não ser muito abrangentes;
    - Não deteta erros internos, porque só interessam os dados de entrada e o resultado obtido.

Stubs e Drivers

- Programas fictícios nos testes de integração que facilitam o teste;
- Substituem componentes ausentes no teste;
- Apenas simulam a comunicação de dados com o módulo de chamada durante o teste;
- Um *stub* é chamado pelo módulo que está a ser testado;
- Um *driver* chama o módulo que está a ser testado.

Metodologia incremental

- O teste de integração é feito integrando dois ou mais módulos que estão logicamente relacionados entre si e são testados quanto ao seu funcionamento na aplicação;
- Os outros módulos são integrados de forma incremental até que todos se encontrem integrados e testados com sucesso.

Top Down

- Subcategoria da metodologia incremental na qual o teste de integração ocorre de cima para baixo, segundo o fluxo de conteúdo do sistema de software;
- Os módulos do nível superior são testados primeiro e os módulos do nível inferior são, de seguida, testados e integrados para verificar a funcionalidade do software;
- *Stubs* são utilizados se necessário.

Bottom Up

- Subcategoria da metodologia incremental em que os módulos do nível inferior são testados primeiro;
- O teste dos módulos de níveis superiores é facilitado pelos módulos testados;
- O processo termina quando os módulos de nível superior são testados e integrados;
- São utilizados drivers se necessário.

Sandwich

- Subcategoria da metodologia incremental onde os módulos de nível superior são testados com os módulos de nível inferior;
- Módulos inferior são integrados com módulos superiores e testados como um sistema;
- Trata-se da combinação híbrida do Top Down e Bottom Up;

- Utiliza *stubs* e *drivers*.

Características de um plano de teste de integração

- Abordagens para teste;
- Scopes e itens fora dos escopos;
- Papéis e responsabilidades;
- Pré-requisitos para teste de integração;
- Ambiente de teste;
- Plano de risco e mitigação.

Critérios de entrada e saída

- Testes de integração devem ser realizados quando:
    - Componentes/módulos já foram testados por unidade;
    - Todos os módulos devem ser codificados e integrados;
    - Plano de teste de integração, casos de teste e cenários têm de ser assegurados e documentados;
    - Requer ambiente de teste a ser configurado par ao teste.
- Devem ser dados por concluídos quando:
    - Teste bem-sucedido de aplicação integrada;
    - Casos de teste executados têm de ser documentados;
    - Documentos técnicos a serem submetidos, com notas de lançamento.

Processo de testes de sistema

- São testados o design e o comportamento do sistema, com foco nas expectativas do utilizador;
- Realizado por uma equipa de testes independente para manter a imparcialidade na avaliação do software a nível de requisitos funcionais e não funcionais.

Teste de regressão

- Verificam que não existem defeitos no sistema após conceber uma nova versão do mesmo;

Teste funcional

- Verificam se há funcionalidades em falta no sistema.

Teste de recuperação

- Verificam a capacidade de o sistema recuperar corretamente de uma falha ou *crash*;
- Após uma falha deve ser emitido um *crash log* que informe o que se passou, fechando todos os processos do sistema até que recupere o estado.

Teste de migração

- Verificam se o sistema funciona corretamente após a transição para uma nova infraestrutura.

Teste de usabilidade/acessibilidade

- Verificam que o sistema é apelativo e familiar ao utilizador e corresponde às suas expetativas em termos de acesso, aspeto e navegação.

Teste de software e hardware

- Verificam a compatibilidade do hardware ou software;

- A compatibilidade do sistema fornece flexibilidade.

Vantagens dos testes de carga

- Minimizam os custos, pois são uma forma de mitigação;
- Aumentam a satisfação dos clientes;
- Minimizam a probabilidade do sistema não se encontrar disponível;
- Detetam erros de performance antes de ser levado à produção.

Desvantagens dos testes de carga

- Requerem conhecimento a nível de software;
- As ferramentas são dispendiosas.

Testes de Deployment

- Verificam se a aplicação funciona corretamente em ambiente de produção, sem bugs e problemas de performance.

Módulo

Nome do módulo / Descrição / Prioridade (Must have/Should have/Could have/ Won't have)

Requisitos Funcionais

ID / Módulo a que pertence / Descrição (O sistema deverá permitir que o utilizador…)/ Prioridade

Requisitos de Qualidade

ID/ Categoria / Tipo / Descrição (O sistema deverá funcionar…)/ Prioridade

ISO 9126

- Portabilidade
    - Adaptabilidade (Ex: O sistema deve funcionar em qualquer tipo de sistema operativo);
    - Instabilidade (Ex: As funcionalidades bases do sistema devem de estar disponíveis sem ser necessária configuração previa);
    - Conformidade (Ex: O sistema deve funcionar segundo a regulamentação da portabilidade europeia);
    - Substituibilidade (Ex: O Sistema deve permitir resolver o problema de gestão de equipas de *esports* melhor que as ferramentas atuais).
- Funcionalidade
    - Adequação (Ex: O sistema deve garantir que o utilizador tenha uma ferramenta de calendarização);
    - Precisão (Ex: O sistema deve garantir que a funcionalidade de procura de scrims automática encontre a equipa mais adequada);
    - Interoperabilidade (Ex: O Sistema deve permitir ver o perfil de equipas através de uma API);
    - Segurança (Ex: O sistema deve encriptar todos os dados sensíveis pertencentes às equipas e utilizadores);
    - Conformidade de usabilidade (Ex: O sistema deve garantir a segurança na gestão de dados do

cliente, segundo o regulamento SOC).

- Manutenção
    - Capacidade de análise (Ex: O sistema deve garantir que todas as alterações dos dados do sistema fiquem registadas na base dados);
    - Mutabilidade (Ex: O sistema deve garantir que o código do software possa ser modificado facilmente);
    - Testabilidade (Ex: O sistema deve permitir verificar onde ocorreram erros no front-end através da consola do browser);
    - Estabilidade (Ex: O sistema deve garantir a responsividade em horas de maior afluência).
- Confiabilidade
    - Maturidade (Ex: O sistema deve verificar a introdução de caracteres inválidos nos respetivos campos);
    - Tolerância ao erro (Ex: O sistema deve se manter estável mesmo que os utilizadores introduzam informação ou tipos de dados errados nos campos de "input" da aplicação);
    - Recuperabilidade (Ex: O sistema deve permitir que o utilizador consiga recuperar a sua conta caso se tenha esquecido da password).
- Eficiência
    - Comportamento do tempo (Ex: O sistema deve garantir que o login demore menos de 15 segundos a ser concluído);
    - Utilização de recursos (Ex: O sistema deve apagar as variáveis temporárias utilizadas no browser após a utilização das mesmas).
- Usabilidade
    - Apreensibilidade (Ex: O sistema deve garantir que os utilizadores aprendam a utilizar o sistema de forma rápida);
    - Inteligibilidade (Ex: O sistema deve garantir que o utilizador rapidamente compreenda as capacidades do sistema);
    - Operabilidade (Ex: O sistema deve garantir que o utilizador consiga alterar que campos estão visíveis no seu perfil).

TRQ2	Descrição
Pior caso:	O sistema não funciona corretamente em nenhum dos browsers mais populares
Planeado:	O sistema funcionar corretamente nos 4 browsers mais utilizados
Teste:	Testar o funcionamento do sistema nos mais populares browsers. Foram considerados o Firefox, Chrome, Internet Explorer e Microsoft Edge.
Unidades:	Número de sucessos em 4 dos browsers

Tabela 25 - Teste do requisito de qualidade de "O sistema deve funcionar em qualquer tipo de Browser"

Requisitos Ambientais

ID/ Categoria/Descrição

Categorias: Hardware (PC), Software (Qualquer sistema que suporte o programa), Linguagem (JAVA), Browser

(Qualquer browser lançado depois de 2012), Ambiente de desenvolvimento (Qualquer programa que dê para desenvolver JAVA, etc.), necessário para o cliente correr a aplicação (Qualquer sistema que suporte o programa) e Normas/Legislação a seguir (Utilização do padrão de desenvolvimento de software MVC (Model-view-Controller)).

Projeto

- Procedimento completo de desenvolvimento de software;
- Realizado de acordo com as metodologias de execução, num período de tempo especificado para obter o produto pretendido.

Projeto de Desenvolvimento

- Propõe a criação de novos produtos de software sem modificar outros existentes;
- Os requisitos funcionais são levantados, priorizados e incluídos no produto final.

Projeto de Manutenção

- Propõe a modificação de um produto de software existente;
- Ocorrem por incidência, mudança de ambiente, novas necessidades ou melhorias;
- A inclusão ou evolução de novos requisitos depende do tipo de projeto de manutenção.

Projeto de Correção

- Tipo de projeto manutenção que modifica o software para corrigir defeitos na sua constituição;
- Não há inclusão de novos requisitos.

Projeto de Adaptação

- Tipo de projeto de manutenção que modifica o software para acomodar alterações ocorridas no ambiente externo;
- Não há inclusão de novas funcionalidades;
- Estas modificações podem ser aplicadas no ambiente operacional (hardware, OS, legislação, normas, …).

Projeto de Evolução/Melhoria

- Tipo de projeto de manutenção que estende o software além dos requisitos originais;
- Há inclusão de novas funcionalidades;
- Trata-se de sistemas de grande escala que nunca estão completos;
- Ex.: Windows, Linux.

Projeto de Prevenção

- Tipo de projeto de manutenção que modifica o código do programa de modo que possa ser corrigido, adaptado e evoluído mais facilmente;
- Não há inclusão de novas funcionalidades;
- É motivado por fraca documentação, conhecimento limitado do sistema por parte dos *developers* ou custos elevados de manutenção;
- Reestruturação da documentação pode ser um processo caro, pelo que a documentação deve ser criada apenas quando se altera o sistema;
- Engenharia inversa é o processo de recuperação de projetos,

traduzindo-o a uma representação num nível de abstração maior;
- Reestruturação do código trata-se de *refactoring* que fornece mais qualidade às funcionalidades a nível de código;
- Reestruturação de dados torna o acesso aos dados num processo mais eficiente (ex.: migração de dados);
- *Forward engineering* utiliza ferramentas para recuperar usar as informações de um software existente para alterar ou reconstituir o mesmo melhorando a qualidade;
- Migração de dados substitui sistemas legados que não conseguem acompanhar os crescentes requisitos de desempeno, o que pode reduzir consumos de energia e espaço reduzindo também os custos (ex.: mover os dados para a *cloud*).

Projeto de Aposentaria

- Tipo de projeto de manutenção que tem como objetivo remover um produto de software ou hardware do ambiente de produção;
- É motivado por custos elevados de manutenção, duplicação de funcionalidades em vários sistemas ou a retirada do software das máquinas em causa;
- Quando se decide que um sistema deve ser reformado, faz-se uma análise o sistema para criar um plano de reforma;
- Após o estudo do impacto de reforma do sistema, faz-se arquivação do mesmo.

Como gerir um projeto

- Definição do espaço/propósito;
- Elaboração de estratégias;
- Integração de colaboradores;
- Monitorização;
- Encerramento.

Definição do espaço/propósito

- Consideração do prazo, orçamento e níveis de qualidade de projeto;
- É essencial estabelecer os requisitos de software.

Elaboração de estratégias

- Consideração das ferramentas necessárias à gestão do projeto de software, a equipa responsável, as estratégias e os sistemas;
- *Scrum* e *Kanban* são boas estratégias/metodologias de gestão de um projeto.

Integração de colaboradores

- Etapa de integração/alinhamento dos objetivos, métodos, orçamento e equipa para executar o projeto.

Monitorização e controlo

- Verificação do funcionamento das estratégias face ao esperado no escopo do projeto;
- Análise dos indicadores e parâmetros estabelecidos no planeamento;
- Fase de testes é fundamental;
- Possibilita a verificação de resultados e a reformulação de estratégias, se necessário.

Encerramento

- Etapa que encerra o projeto num prazo acordado com o cliente.

4 Ps de Gestão de Projetos

- Uma gestão eficaz foca:
  - Pessoas;
  - Processo;
  - Produto;
  - Projetos.

Pessoas

- Todas as pessoas envolvidas no ciclo de vida de um software;
- Devem ser organizadas para realizar o trabalho atribuído eficientemente.

Processo

- Modelos, metodologias e tecnologias utilizadas durante a execução de um projeto de software.

Produto

- Estimativa do custo, tempo e esforço necessário à produção do projeto de software.

Projeto

- Medidas corretas a tomar para a conclusão bem-sucedida do projeto de software;
- O objetivo do projeto deve ser claro;
- A análise de requisitos deve ser feita;
- As necessidades devem ser monitorizadas.

Stakeholders

- Entidades envolvidas no projeto;
- Um *stakeholder* é identificado quando uma dada entidade tem impacto sobre o orçamento ou o resultado alcançado pelo projeto ou quando o seu trabalho e/ou decisões alteram o escopo do projeto ou quando o resultado obtido do projeto afeta uma entidade.

Project Manager

- Desempenham o papel de liderança no planeamento, execução, monitorização, controlo e encerramento do projeto;
- Responsáveis pelo planeamento do projeto, a equipa de trabalho, os recursos e o sucesso ou fracasso do projeto;
- Devem liderar, motivar, comunicar, organizar e adaptar.

Fatores que afetam um projeto

- O insucesso de um projeto advém de:
  - Falhas na comunicação;
  - Falhas no planeamento;
  - Incumprimento de requisitos;
  - Problemas com os recursos utilizados.

Falhas na comunicação

- Um PM que não incentiva a boa comunicação com o cliente corre o risco de construir soluções para os problemas errados.

Falhas no planeamento

- Um PM que inicia o projeto sem um bom planeamento põe em causa o sucesso do desenvolvimento do produto.

Incumprimento de requisitos

- A especificação de requisitos deve ser feita com consciência na fase de levantamento, para conhecer as necessidades do cliente e os recursos a utilizar;
- Requisitos complexos devem ser compartimentados para ser perceberem melhor;
- Todos os requisitos devem ser validados antes de se iniciar a execução do projeto.

Problemas com os recursos

- A equipa deve conhecer as metodologias e tecnologias necessárias ao desenvolvimento;
- A competência da equipa utilizar as técnicas escolhidas consiste num risco de habilidade;
- Pode haver modificação de recursos durante o desenvolvimento do projeto, desde recursos orçamentais a humanos;
- A incorporação de novos membros na equipa implica tempo de adaptação.

Fatores de toxicidade das equipas

- A toxicidade da equipa de desenvolvimento de um projeto pode aumentar com conflitos como:
    - Excesso de competição no ambiente de trabalho;
    - Diferenças de personalidade;
    - Escassez de recursos necessários;
    - Disparidade de informação e perceção.

Qual o aspeto mais importante na gestão de pessoas?

- Gestão de pessoas é o conjunto de práticas que englobam os processos de aquisição de talento, otimização de talento e retenção de talento, fornecendo apoio contínuo ao negócio e orientação para os colaboradores;
- A organização das pessoas em equipa leva a uma melhor projeção das qualidades de cada um;
- A organização das pessoas em equipas possibilita a cada elemento trabalhar no que gosta ou no que se candidata, reduzindo a falta de motivação e interesse.

Estrutura de equipas

- *Generalist*:
    - Equipa de indivíduos com diversas qualidades.
- *Specialist*:
    - Equipa que envolve experts com qualidades específicas.
- *Hybrid*:
    - Combinação dos dois;
    - Equipa híbrida que se encontra apta a projetos genéricos e especializados.

Orçamento

- Documento de despesas de uma entidade, com um custo estimado.

O que se coloca num orçamento?

- Nome, NIF, morada e contacto de entidade organizacional e do cliente;
- Datas de emissão e validade;
- Descrição detalhada do trabalho a realizar, mão de obra e matérias;
- Prazos de execução;
- Formas de pagamento;
- Adiamentos;
- Garantias.

O que deve ter um orçamento?

- Salário de equipa/mão de obra;
- Número de horas a despender;
- Materiais e custo correspondente;
- Custos de produção;
- Despesas operacionais ou gestão fixos.

Case studies

- Processo de estudo de uma entidade que fornece informação sobre o esforço de trabalho necessário ao desenvolvimento de um projeto para contabilizar essa mesma informação no orçamento.

Como elaborar um orçamento?

- A estimativa do custo de projeto de software é conseguida através a avaliação do tipo de projeto, de discriminação da sua complexidade e da localização da equipa de desenvolvimento.

Tipo de projeto de software

- Os custos de cada tipo de projeto variam, pois utilizam recursos diferentes.
- Tipos/custo:
    - PC Software / 10 000€ - 800 000 €
    - Web App / 5 000€ - 300 000€
    - Mobile App / 5 000€ - 500 000 €
    - Enterprise software / 30 000€ - 55 000€

Complexidade do projeto

- O custo de um projeto de software é diretamente proporcional à sua complexidade;
- Quanto mais complexo, mais tempo de desenvolvimento;
- Um projeto simples contém funcionalidades e componentes UI simples, sem integração de APIs e com desenvolvimento *backend* básico;
- Um projeto moderado contém integração de APIs, funcionalidades de pagamentos e desenvolvimento de *backend*;
- Um projeto complexo contém arquiteturas de *backend* complicadas e integração *third party*;
- Complexidade / exemplo:
    - Simples / Software de mensagens;
    - Moderado / Redes sociais;
    - Complexo / Videojogos e *streaming* software.

Localização da equipa de desenvolvimento

- O salário mínimo de um *developer* varia entre países, o que afeta a estimativa do custo de um software.

Riscos em Orçamento

- Suposições e estimativas de um orçamento são imprecisas;
- Podem gerar problemas no controlo do orçamento.

Scope Creep

- Risco de orçamento onde a visão inicial do projeto face aos requisitos é estreita e vai alargando com o tempo.

Gold Plating

- Risco de orçamento onde são adicionadas funcionalidades extra ao projeto, sem que o cliente as tenha pedido, aumentado o custo.

Alteração do valor da moeda

- Risco de orçamento onde se trabalha num projeto internacional e as variações do valor da moeda podem levar à perda de dinheiro.

Falta de noção das datas

- Risco de orçamento onde a não determinação das datas de término do projeto podem levar a orçamentos mal calculados, potencialmente sem lucro ou com prejuízo.

Noção de custos

- Risco de orçamento onde um orçamento baixo pode levar à perda de dinheiro e um orçamento alto pode levar à incapacidade de pagamento do cliente, o qual desiste do investimento.

Contabilização dos riscos de projeto

- Um orçamento deve incluir os riscos de desenvolvimento do projeto;
- Efetua-se a gestão de riscos no projeto com a elaboração de um planeamento, identificação, análise e tratamento de riscos.

Risco

- Potencial problema no software que pode ocorrer independentemente do resultado;
- Deve ser identificado;
- Deve ser estimada a probabilidade de o risco ocorrer;
- Deve ser estimado o impacto da sua ocorrência;
- Deve ser feito um plano de contingência;
- Categoria / associação:
    - Tamanho do produto / tamanho do projeto;
    - Negócio / limitações aplicadas pela organização;
    - Características de *stakeholders* / capacidade de comunicação e planeamento do *stakeholder*;
    - Definição de processo / planeamento do software pela equipa;
    - Ambiente de desenvolvimento / disponibilidade e qualidade das ferramentas;
    - Implementação das tecnologias / complexidade do sistema;

    - Tamanho da equipa e experiência / capacidade técnicas e experiência da equipa.

Prevenção de riscos

- Os principais elementos da equipa e o cliente comprometem-se a dar suporto ao projeto;
- Os utilizadores finais estão interessados no projeto e no produto resultante;
- Os requisitos são conhecidos e compreendidos pela equipa e pelo cliente;
- Os clientes estiveram diretamente envolvidos na definição dos requisitos;
- Os clientes têm expetativas realistas;
- O projeto encontra-se num ambiente estável;
- A equipa tem conhecimentos adequados;
- A equipa tem experiência com tecnologia a ser usada;
- Os clientes concordam com a importância do projeto e com os requisitos em que se baseia.

Importância da gestão de riscos

- Muitas coisas podem correr mal, muitas coisas estão erradas;
- Compreender os riscos e tomar medidas ativas para evitar ou combater os mesmos faz parte da boa gestão de um projeto;
- Um plano de contingência de riscos fornece um meio de combate ao agravamento do impacto da ocorrência de um risco.

Passos a tomar

- Reconhecer o que pode correr mal (identificação de riscos);
- Cada risco é analisado para determinar a probabilidade de ocorrer e o impacto que teria sobre o projeto;
- Os riscos são classificados;
- Desenvolve-se um plano de contingência.

Análise de impacto de um risco

- Natureza:
    - Indica os problemas que são prováveis de surgir;
    - Ex.: Uma interface que impede o design e testes prejudicam a qualidade do produto.
- Âmbito:
    - Conjuga a gravidade com a abrangência.
- Timing:
    - Considera quando e por quanto tempo o impacto de um risco será sentido.

Apuração de riscos

- O passar do tempo de desenvolvimento adquire mais informação do projeto, o que pode ser utilizado para refinar a descrição de um risco para que seja mais fácil de mitigar, monitorizar e gerir;
- CTC (*condition-transition-consequence*) é uma forma de representar um risco.

Risk Management Maturity Model (RMMM)

- Processo de gestão de riscos de um projeto;
- Avalia o projeto dos *stakeholders*, identifica os riscos, analise os riscos, avalia a resposta aos diferentes riscos, avalia a gestão do projeto e formula a cultura de gestão de risco;
- As seis perspetivas analisados do projeto fornecem uma avaliação global da capacidade de gestão de risco do mesmo.

Risk Information Sheets (Exemplos)

Risk Information Sheet			
Risco ID: 1	**Data:** 26/03/2021	**Probabilidade:** 60%	**Impacto:** Moderado
Descrição: Integração da biblioteca externa			
Refinamento/contexto: Com o intuito de aceder às milhares de músicas/artistas existentes pretendemos utilizar uma biblioteca externa. Qualquer componente externo tem os seus riscos de implementação uma vez que pode estar em baixo ou mudar os seus termos e condições.			
Mitigação/Monitorização: Tomar atenção às mudanças da biblioteca externa e garantir uma implementação que permita mudar no caso de imprevistos.			
Plano de Gestão/Contigência/Gatilho: Adaptar as funcionalidades de modo a conseguir uma alteração eficiente para uma biblioteca diferente ou até mesmo a realização de uma base de dados pessoal.			
Estado Atual: 26/03/2021. Ocorreu. Soluções a ser implementadas			
Originado por: Miguel Sequeira		**Responsável:** Miguel Sequeira	

Risk Information Sheet			
Risco ID: 2	**Data:** 26/03/2021	**Probabilidade:** 55%	**Impacto:** Insignificante
Descrição: Não implementação de todos os requisitos definidos nas fase iniciais do projeto.			
Refinamento/contexto: Este projeto está a ser desenvolvido tendo em conta uma metodologia ágil podendo portanto ter a possibilidade de mudanças nos requisitos. Da mesma maneira que poderão não ser implementadas funcionalidades iniciais também poderão vir a ser implementadas novas funcionalidades.			
Mitigação/Monitorização: Fazer uma avaliação detalhada sobre os prós e contras da implementação de certas funcionalidades.			
Plano de Gestão/Contigência/Gatilho: Realizar o melhor trabalho possível dentro do período de tempo definido.			
Estado Atual: 26/03/2021. Alguma funcionalidades tiveram de ser alteradas.			
Originado por: Miguel Sequeira		**Responsável:** Miguel Sequeira	

Risk Information Sheet			
Risco ID: 3	**Data:** 26/03/2021	**Probabilidade:** 35%	**Impacto:** Crítico
Descrição: Escalonamento de acessos			
Refinamento/contexto: Num website com diferentes níveis de acesso e informações pessoais sobre os utilizadores é de maior importância um bom escalonamento de acessos em diferentes partes do website. Desde apenas o utilizador conseguir mudar coisas no seu próprio perfil, a administradores de fórum e administradores de grupos se estes módulos forem implementados, todos esses diferentes níveis de acessos têm que se distinguir e ser atribuídos aos utilizadores certos.			
Mitigação/Monitorização: Testar acessos com diferentes níveis de acessos e autenticações ao longo da implementação do projeto.			
Plano de Gestão/Contigência/Gatilho: Proteger certas áreas do website em termos de acesso por User Roles e pela autenticação.			
Estado Atual: 26/03/2021 - Começar a trabalhar nos Roles			
Originado por: Bernardo Teixeira		**Responsável:** Bernardo Teixeira	

Risk Information Sheet			
Risco ID: 11	**Data:** 05/04/2021	**Probabilidade:** 30%	**Impacto:** Catastrófico
Descrição: Dificuldades de comunicação com o cliente			
Refinamento/contexto: Ao realizar um projeto tem que se ter em conta as orientações dos clientes logo se existirem dificuldades a receber orientações dos mesmos não se conseguirá realizar o produto que o cliente pretende.			
Mitigação/Monitorização: Realizar reuniões regulares com o cliente.			
Plano de Gestão/Contingência/Gatilho: Caso existam problemas ou duvidas nas intenções do cliente deve-se reunir com o mesmo de modo a conseguir chegar a conclusões.			
Estado Atual: 05/04/2021 - Comunicações bem realizada.			
Originado por: Miguel Sequeira		**Responsável:** Miguel Sequeira	

Matriz de riscos

	Insignificante	Moderado	Critico	Catastrófico
Muito Provável(80-100%)		10		
Provável(60-79%)		1,14		
Possível(40-59%)	2	6	9	
Improvável(20-39%)		5	3, 4, 7	11, 12
Muito Improvável(0-19%)		8	13	