---- perguntar à stora ---- para modificar ---- para rever

---- feito ---- não feito

**Architecture Notebook**

1. **Propose**

Este artefacto descreve a arquitetura de software, mais concretamente a filosofia, as decisões, as restrições, elementos significativos e quaisquer outros aspetos gerais do sistema que moldam o design e a implementação. Este preocupa-se com a compreensão de como o sistema deve ser organizado e com a estrutura geral do mesmo. É o elo crítico entre o projeto e a engenharia de requisitos, pois identifica os principais componentes estruturais de um sistema e os relacionamentos entre eles.

É imprescindível pois ajuda numa melhor captura de informação e tomada de decisões arquitetónicas relativas ao produto de modo a que este permaneça com as caraterísticas e requisitos desejados.

Com a utilização deste artefacto somos capazes de cumprir o objetivo principal do sistema, prevenir o suicídio em estabelecimentos prisionais e conseguir explicar as nossas decisões aos desenvolvedores para um melhor resultado final.

1. **Architecture Goals and Philosophy**

O objetivo a que a nossa empresa se propõe com o desenvolvimento da ***SuddenAlert*** é melhorar os sistemas de prevenção do suicídio nas prisões para que haja cada vez menos suicídios. A nossa solução ajuda no monitoramento dos reclusos de maneira a prever e agir antecipadamente a situações que possam originar tendências suicidas, assim como obter uma informação viável mais rápida do estado de saúde mental do recluso.

Este consiste em inserir manualmente na aplicação informações sobre o comportamento do recluso que serão analisadas e assim as entidades prisionais obtêm uma melhor perceção do estado psicológico do recluso. Também haverá a opção de consultar o horário pessoal da entidade e permitirá aos psicólogos registar as suas consultas.

1. **Assumptions and Dependecies**

De seguida enumeramos os limites, suposições e dependências que nos levaram à tomada de decisões arquitetónicas. Em relação aos limites consideramos os seguintes:

Depois

* Legislação.

A nossa aplicação terá de conter todos os requisitos legais para o desenvolvimento da mesma. Entre estes destacam-se:

1. Lei de Proteção da Propriedade Intelectual de Programas de Computador
2. Lei de Propriedade Industrial
3. Código de Defesa do Consumidor

* Norma ISO/IEC 9126 "Engenharia de software - Qualidade do produto"

1. Usability – Eficiência na tarefa

Terminar uma tarefa de dificuldade mínima em 1 minuto; dificuldade média em 3 minutos: dificuldade máxima em 5 minutos.

1. Reliability – Disponibilidade

Software a trabalhar durante 72h (3 dias), manutenções se necessárias de 3 em 3 dias.

- Recuperabilidade

Backups semanais.

- Frequência e gravidade nas falhas

Pouca frequência nas falhas e lidar com as falhas em 1 dia se for “minor”, em 3 dias se for ”significant” e em 5 dias se for “critical”

1. Performance
2. Supportability

* Disponibilidade das entidades prisionais utilizarem a aplicação.

Antes

* Lidar com grandes quantidades de dados;
* Perdas de dados constantes;
* Dificuldades na detenção de falhas e situações de alerta;
* Não consiga ter uma monotorização em tempo real.

**Leis**

<https://parceirolegal.fcmlaw.com.br/leis/5-leis-ao-criar-um-aplicativo/>

<https://www.startse.com/noticia/nova-economia/tecnologia-inovacao/criacao-de-um-aplicativo-por-onde-comecar>

Esta aplicação tem de estar de acordo com a nova lei europeia relativa à proteção de dados pessoais, Regulamento (EU) 2016/679 aprovada a 27 de abril de 2016, que se baseia no direito/obrigação à privacidade de qualquer tipo de informação recolhida pelas empresas/softwares.

**No trabalho:**

Quanto às restrições do sistema, temos que a linguagem de implementação de software será em português. Relativamente às ferramentas de desenvolvimento, componentes de terceiros ou bibliotecas de classes, suporte de plataforma, limites de recursos e requisitos sobre a forma, tamanho e peso do hardware resultante que suporta o sistema ainda não temos ideia de quais restrições haverá.

(F)URPS+

Functionality - RF

Usability, Reliability, Performance, Supportability, (Design constraints, Implementation constraints, Interface constraints, Physical constraints) - RNF

1. **Architecturally significant requirements**

Os requisitos que devem ser implementados na nossa arquitetura são os seguintes.

* Sistema de login seguro;
* O diretor deverá poder aceder a toda a informação registada no sistema;
* Os guardas e psicólogos deverão atualizar constantemente os dados do sistema para uma maior perceção do que se está a passar;
* Avisos de vigilância antecipados de determinados reclusos;
* Aplicação atualizada em tempo real.

Requisitos que poderão ser adiados para versões futuras do sistema:

* Versão da aplicação móvel para computador.

1. **Decisions, constraints and justifications**

Antes

* Restrições no login à prova de falha de maneira a assegurar a segurança dos dados do sistema;
* Aviso antecipados de mudanças de horários;
* Criação de horários que evitem possíveis conflitos;
* Avisos de elaboração de relatório para o diretor poder analisar.

Depois

**Decisions:**

* Sistema de login seguro e à prova de falha de modo a assegurar a segurança dos dados do sistema.

**Constraints:**

Relativamente às restrições podemos dividi-las em 4 tipos segundo o FURPS+, sendo estes: Design Constraint, Implementation Constraint, Interface Constraint, Physical Constraint.

**Design:**

* Programa de implementação do design é o Android Studio.
* Resolução: 1080x2160: 440 dpi.

**Implementation:**

* Linguagem de implementação de software em português.
* Programa de implementação do código é o Visual Studio Code.
* Bibliotecas de classe a serem implementadas:

java.lang.\* – é o único pacote que é automaticamente importado.

java.lang.Math – para usar métodos matemáticos

java.lang.Throwable – para representar situações de erro

java.util.\* – para utilizar classes úteis ao projeto (Arrays, Calendar,…)

com.google.zxing.\* – para trabalhar com QR codes.

java.awt.\* – para unir o design com o código

java.sql – para conectar à base de dados

**Interface:**

* Cores neutras com o uso da cor de tijolo (cor da engenharia) mais avermelhada como *default*, no entanto cada entidade terá a opção de mudar a cor no seu perfil.
* Acesso necessário a uma base de dados.
* Uso da câmara para o scan.
* Uso da localização.
* Comunicação com os servidores através de encaminhadores.
* Não poderá ter acesso à internet.

**Physical:**

* A aplicação deve ser adaptável ao peso e tamanho do hardware.

1. **Architectural Mechanisms**

Os mecanismos arquiteturais representam conceitos técnicos fundamentais que serão padronizados por toda a solução. Eles são refinados durante o projeto em três estados, representados pelas três categorias de Mecanismos Arquiteturais:

* Mecanismo de Análise, que dá ao mecanismo um nome, uma descrição resumida e alguns atributos básicos derivados dos requisitos do projeto
* Mecanismo de Design, que são mais concretos e assumem alguns detalhes do ambiente de implementação
* Mecanismo de Implementação, que especifica a exata implementação de cada mecanismo

**Mecanismos de Análise**

Os mecanismos de análise são os estados iniciais para os mecanismos arquiteturais. São identificados no início do projeto e representam marcações para futuro desenvolvimento de software. Eles permitem à equipa focar na compreensão dos requisitos sem se distrair com as especificidades de uma implementação complexa. Os mecanismos de análise são descobertos pelo levantamento dos requisitos e pela procura de conceitos técnicos recorrentes. Segurança, persistência e interfaces legadas são alguns exemplos. Com efeito, o Arquiteto reúne os requisitos que descrevem tópicos arquiteturalmente importantes colocando-os em uma única lista. Tornando-os mais fáceis de serem gerenciados.

Os mecanismos de análise são descritos com termos simples:

Nome: identifica o mecanismo.

Atributos Básicos: definem os requisitos do mecanismo. Estes atributos podem variar dependendo do mecanismo que está sendo analisado. Veja Example: Atributos de Mecanismo Arquitetural para mais orientação.

Uma vez que a lista de mecanismos de análise tenha sido definida, ela pode ser priorizada e os mecanismos refinados de acordo com os objetivos da iteração. Não é necessário desenvolver todo o conjunto de mecanismos arquiteturais em software funcional de uma única vez. Muitas vezes é mais sensato desenvolver apenas os mecanismos necessários para suportar a funcionalidade a ser entregue na iteração atual.

**Mecanismos de Design**

Os mecanismos de design representam decisões sobre tecnologias concretas que serão usadas para desenvolver mecanismos arquiteturais. Por exemplo, a decisão de usar um RDBMS para persistência. Normalmente nada mais complexo do que isso (embora, evidentemente, o esforço envolvido na tomada de decisões pode, por vezes, ser muito complexo).

A decisão sobre quando aperfeiçoar um mecanismo arquitetônico do estado de análise para o de projeto é em grande parte arbitrária. Frequentemente haverá restrições em vigor no projeto que forcem a tomada de decisão a respeito de algumas destas questões. Por exemplo, pode existir um padrão corporativo para bancos de dados significando que a decisão para a adoção de um mecanismo de persistência possa ser tomada no primeiro dia do projeto.

Em outras ocasiões, a decisão pode direcionar para produtos que a equipe do projeto ainda não adquiriu. Neste caso, a decisão tem de ser tomada em tempo hábil para permitir os produtos necessários sejam colocados à disposição da equipe.

Muitas vezes, pode ser útil desenvolver um código prototípico para provar que estas decisões estão sólidas. O Arquiteto deve estar seguro de que as tecnologias selecionadas sejam capazes de atender os requisitos. Os atributos capturados com os mecanismos de análise correspondentes deverão ser usados como critérios para comprovar a validade das decisões.

**Mecanismo de Implementação**

Um mecanismo de implementação especifica a implementação real para o mecanismo arquitetural (daí o nome). Ele pode ser modelado como um padrão de design ou representado como código de exemplo.

O melhor momento para produzir o mecanismo de implementação é normalmente quando a primeira parte da funcionalidade necessária é agendada para desenvolvimento. O Arquiteto e o Desenvolvedor trabalham juntos para desenvolvê-lo.

1. **Key abstractions**
2. **Layers or Achitectural Framework**

A aplicação possui páginas semelhantes relativamente aos horários dos psicólogos e guardas. Possui também uma área de monitorização de reclusos que lhe permite visualizar todos os reclusos, adicionar informações, como situações de alerta, e apenas o diretor pode registar e remover reclusos com a devida justificação.

**Spring** para uma melhor eficiência, rapidez, segurança e produtividade. (falar mais)

**NativeScript** para o design no Android Studio.

**GeradorQRCode** por willian C. para conseguirmos gerar e trabalhar com QR codes.

1. **Architectural Views**

A arquitetura de software é normalmente organizada em visões,[4] as quais são análogas aos diferentes tipos de plantas utilizadas no estabelecimento da arquitetura. Na Ontologia estabelecida pela ANSI/IEEE 1471-2000, visões são instâncias de pontos de vista, onde cada ponto de vista existe para descrever a arquitetura na perspectiva de um conjunto de stakeholders e seus consortes.

Algumas possíveis visões são:

* Visão funcional/lógica
* Visão de código.
* Visão de desenvolvimento/estrutural
* Visão de concorrência/processo/thread
* Visão física/evolutiva
* Visão de ação do usuário/retorno

**Restrições, suposições e limitações**

<https://www.slideshare.net/Hironaka1/ieee-830-61158149>

(página 29)

<https://slideplayer.com.br/slide/2264877/>

(apartir do tópico 6)

<https://medium.com/@marciofrayze/encontrando-depend%C3%AAncias-desatualizadas-em-aplica%C3%A7%C3%B5es-java-eff3f3a85577>

**Visões**

Wikipédia

<http://mds.cultura.gov.br/extend.bus_model/guidances/concepts/business_architecture_D437B035.html>

**Arquitetura em geral**

<https://slideplayer.com.br/slide/2459045/>

<https://slideplayer.com.br/slide/356345/>

<https://pt.slideshare.net/marcelohyamaguti/arquitetura-de-software-54412307>

<https://www.slideserve.com/dawn-mack/introdu-o-arquitetura-de-software>

<http://www.machado.mg.gov.br/files/concursos/1cf11cf161fe4eb688dfeec880d6b4d9.pdf?__cf_chl_jschl_tk__=951a4af3e5416cce3bc80085e383d27adbe72504-1583336845-0-AU4P5AS51WUxRJN_KAiFIdM5RoUYPTnJcUyjlgYfR_YyVhOdVOrzdSYDoNU5v28-Sx1YZQGiem3TLMiQDDse0QByVsLN4l-dwZ1YtCmOfRxnmCntD10SIxRVzwt9MQ3zPKiq1G9qalujVPQ6TAaLkkDCky5HwWbSnMnlHcA0W_sJa21bfhnYhAf11a_C_Jt_9AxeE6hvBLOw1WPeVGnvKZhpLaMToo-AnGse76EDkDalMSKZTKzsstAcGR4saEwjarV_muCHRiFHSlMSzE91VguDgvqRsXOlhxN51w9zlyX4UUw2O7OMckjxQz8XrS6couDRkZq5uFgDxzQQRHW-aWmYIY7zgk1cqStpcUZN3vM3qKNfrvnEiKWtJtcNIDEEVqIQ6GTghrjS7kfNmG2bpuE>

<https://www.devmedia.com.br/arquitetura-de-software-desenvolvimento-orientado-para-arquitetura/8033>

**Importancia da arquitetura**

<https://blog.atmdigital.com.br/saiba-o-que-e-arquitetura-de-software-e-qual-a-sua-importancia/>

<https://slideplayer.com.br/slide/1658539/>