GESTÃO DE ENTREGA DO EASYTODO

Versão 1.0

Histórico da Revisão

Data	Versão	Descrição	Autor
21/11/20	1.0	Gestão de entrega do produto	Luana Cristina

Sumário

1. Introdução

1.1 Finalidade

O EasyToDo é um aplicativo empresarial para planejar e acompanhar as atividades individuais e da equipe em um projeto, empresa ou pessoal. Ele permitirá adicionar e acompanhar lista, atividades, tarefas gerando relatórios para mapear e/ou visualizar os gargalos. Por meio de gamificação permitirá aos membros da equipe ou individualmente haverá recompensas por atividades concluídas. Há a possibilidade de compartilhar por e-mail desde as listas até os relatórios.

1.2 Escopo

Nome do produto: EasyToDo

Missão do produto: Facilitar a priorização e definição das tarefas em cada atividade da lista e dar apoio ao gerenciamento e mapeamento das atividades em cada lista. Otimizar a gestão por meio dos relatórios e visualizações do progresso da lista com todos os responsáveis por atividade.

1.3 Definições, Acrônimos e Abreviações

- CLDC (Connected, Limited Device Configuration): Define o Java como a plataforma padrão.

- MIDP (Mobile Information Device Profile): Define a arquitetura e APIs associadas necessárias para prover o ambiente desenvolvimento aberto para MIDs (Móbile Information Devices).

1.4 Referências

- RUP para Projetos Pequenos
- Canvas
- Documentos de requisitos
- Glossário
- http://viniciusgarcia.me/education/a-biblioteca-do-desenvolvedor-de-software-dos-dias-de-hoje/
- User Stories Applied: For Agile Development

2. Visão Geral do Projeto

2.1 Finalidade, Escopo e Objetivos do Projeto

O objetivo é facilitar o controle e acompanhamento das atividades. Definir padrões e práticas a serem utilizadas na concepção até a entrega do produto.

2.2 Suposições e Restrições

Até o momento não houveram restrições.

- Por conveniência da equipe e autorização do cliente, a linguagem utilizada será o PHP e Android com o Kotlin.

- O banco de dados será o PostgreSQL.

2.3 Produtos Entregues do Projeto

Datas das entregas constam no Plano de Desenvolvimento do EasyToDoList. Abaixo segue a data prevista de entrega do MVP apenas.

Data	Versão	Descrição	Autores
19/09/2021	1.0	Primeira versão de acordo com a prioridade dada a cada requisito do produto.	Luana, Joel, Paulo, Heládio, Mariana.

3. Organização do Projeto e Processo de Gerenciamento

3.1 Estrutura Organizacional

Pessoa	Papéis
Luana Cristina	**Designer/Tester/Analista de negócio**
Heládio Alves	**Administrador dos dados/Desenvolvedor**
Joel Galdino	**Gerente de Projeto/Product Owner**
Paulo Varejão	**Gerente de Produto/Desenvolvedor**
Mariana Duque	**Scrum Master/Desenvolvedora**

3.2 Contatos Externos

Usuários de aplicativos de gerenciamento e criação de tarefas.

3.3 Metodologia XP

Essa metodologia é a melhor para a nossa aplicação por causa da simplicidade em usar, do rápido feedback, permite melhorar e incrementar mudanças, adaptável e se preocupa com a qualidade e isso ajudará a equipe a se comunicar entre a equipe, aprender, melhorar e mudar com rapidez tanto o software quanto a própria equipe.

Práticas	Descrição
Small releases	Entregaremos primeiramente o MVP e depois releases a cada 15 dias. As divisões das releases deverão ser definidas por meio do Scrum Poker com a equipe nas cerimônias. Cada feature, de acordo com as entregas dos requisitos terão pontuações que serão determinadas se serão entregues na semana vigente da entrega ou na outra.
The Planning Game	Usaremos as user stories para estimar as releases por meio de toda a equipe usando o Scrum Poker. Após entendimento e concordância da equipe iremos definir a pontuação e adicionar a release. A pontuação deve ser mais alta de acordo com a dificuldade de construir e concluir a atividade. Isso envolve desde o conhecimento da equipe na tecnologia determinada quanto para novos integrantes. As seleções das estórias serão por meio da determinação da prioridade dos requisitos ou dos stakeholders ou PO. A equipe definirá quais features conseguem entregar nos 15 dias definidos e as demais serão colocada para a próxima quinzena. Obs.: Não pode ultrapassar os 15 dias, se as features selecionadas forem para 17 dias terão que colocar a feature de 2 ou 3 dias para a próxima quinzena. Não se pode iniciar uma feature que não será concluída na quinzena planejada.
Refatoring	Refatoração do código para reestrutura ou reescreve-lo para melhorar a comportamento sem alterar a lógica e melhorar a performance.
Testing	Os desenvolvedores devem pensar antes no teste (tanto positivos quanto negativos) e depois desenvolver a solução relacionada a story user. Podem se basear nos testes outlines. Testes unitários automatizados realizados pelos desenvolvedores da feature e outro responsável (alguém com experiência em teste e diferente da pessoa que desenvolveu a feature) para realizar os testes outlines, funcionais e integração (teste de aceitação do usuário). Smoke test antes de colocar em produção.
Pair Programming	Cada feature ou bug serão feitas por 2 programadores. Em que um vai desenvolver e outro vai acompanhar, mas os dois vão criar a solução para criar os testes antes de desenvolver e no próprio desenvolvimento. Os pares e papéis devem ser e podem ser alterados.
Sustainable	O time pode definir a quantidade de horas diárias e o horário

Pace	de trabalho, assim como o local para trabalhar. A única restrição será quando for programarem a par porque precisam definir uma hora, dia e local para os dois trabalharem ao mesmo tempo, mas pode ser remotamente o local. Porém, as atividades têm que ser entregues na data estabelecida.
Team Code Ownership	Todo o código deve ser conhecido por toda a equipe e por isso deve ter treinamento e repasse de conhecimento a cada Sprint. A refatoração do código deve ser feito por outra pessoa que não codificou e a pessoa que estiver pareando pode ser a que desenvolveu.
Coding Standards	Haverá um documento definindo como deve ser nomeado as variáveis do código e do banco. Tanto as classes quanto os métodos. A formatação deve ser definida também, incialmente iremos utilizar a "Prettier" e caso aja mudanças terá que constar no documento que define os padrões.
Simple Design	Quanto mais simples o design das releases melhor. 1. Código e testes cobrindo o comportamento do código; 2. Sem duplicação de código; 3. Menor número possível em classe em uso; 4. Menor número possível em métodos em uso.
Metaphor	Simplificar os meios de pensar sobre a ajuda e comportamento do sistema.
Continous Integration	Integração diária e smoke test após as atividades entrarem em "Done" e forem intregadas.
On-Site Customer	Usuários, time de desenvolvedores, testes, gerentes, PO e etc não tem hierarquias e sempre trabalham sem barreiras. Cada um tem acesso a cada um. Se houver alguma dúvida pode entrar em contato com os responsáveis sem burocracias.

3.4 User Story

Criação de User Story para cada requisito irá facilitar no entendimento de todos os envolvidos. Cada User Story terão testes outlines para validar o que o responsável desenvolveu.
As User Story serão usadas para colocar no Kaban.
Exemplo de User Story para ser usado:

Nº	Story Title	Descrição	Definição	Test outline
SCN01	User Create Web and App	O objetivo dessa história é criar um usuário	1.O usuário deve cadastrar para usar e acessar as aulas; 2. Deverá ter campos obrigatórios como o "Nome", "CPF", "e-mail", "Senha", etc. 3.Deve solicitar e apresentar ao	1.Os campos obrigatórios são exibidos para o usuário e se ele não preencher a aplicação vai solicitar que

			usuário os termos de compromisso e utilização do aplicativo que deve está marcado para finalizar o cadastro; 4. Após a finalização do cadastro deve enviar um e-mail de confirmação para o usuário; 5. Após a finalização do cadastro deve abrir em seguida a página inicial.	ele preencha esses campos; 2. Se o usuário digitar um CPF inválido será exibido uma mensagem informando isso no pop up ou na própria página; 3. O botão para finalizar o cadastro será ativo para usuário quando ele marcar o box do Termos de contratação; 4. O usuário recebeu um e-mail de confirmação de cadastro; 5. O usuário é encaminhado para a página inicial.

3.5 Metodologia Kaban para acompanhar as entregas das releases.

Facilita no acompanhamento das atividades e cobrança.

Quadros	Descrição
To Do	Essa parte constará as features, teste de integração, testes funcionais, relatórios de testes, smoking test, e bugs que ainda não tem dono e que precisam ser entregues nessa semana.
Doing	Aqui terá as features e bugs que algum responsável pegou e começou a fazer a atividade. Terá uma label em cima para demonstrar o status da atividade baseado na data de início e na data de entrega: 1. Exbirá a cor amarelo se estiver próximo da entrega da feature estiver próxima de finalizar.

	2. Exibirá a cor vermelha se estiver atrasado. 3. Exibirá nenhuma cor se estiver no prazo. Os testes de integração, funcionais e smoking test após finalizados, automaticamente irão para o quadro "Done".
Testing	As features, após finalizada, automaticamente serão colocadas nesse quadro para serem realizados os testes outlines.
Done	Todas as features, teste de integração, testes funcionais, smoking test, relatórios de testes e de entrega, e bugs finalizados e testados ficarão aqui. Apenas o gerente de projeto poderá retirar.
Waiting	Aqui ficará as atividades que foram iniciadas, mas devido a alteração de prioridade ou manutenção crítica que ocorreu o responsável teve que parar para pegar outra atividade. Qualquer outro responsável ou o mesmo pode mudar ele para o status de Doing.

Observações:

1. Se houver alguma atividade cancelada ela não será exibida no quadro.
2. Só será exibido o número e o título das atividades, ou seja, o número e título das Story User, para expandir a descrição deve dar um clique. Exemplo no tópico **User Story** acima.
3. Ao final de cada Sprint (15 dias para cada release) deve ser gerado automaticamente um relatório com as informações de tempo de cada atividade e recurso. Inclusive se a tarefa voltou ou foi para outro quadro e quanto tempo demorou para ser concluída. Quais atividades foram atrasadas e quais foram entregues antes do prazo. Manter o histórico de todos os projetos e atividades executadas no quadro. Apenas o gerente responsável terá disponível essa visão, para o restante dos recursos eles poderão ver apenas o histórico deles.

4. Padrão para obter os Custos e ferramentas para serem utilizadas

4.1 Estimativas do Projeto e ferramentas utilizadas

- O projeto terá uma estimativa de duração de 10 meses.
- Por ser um projeto acadêmico, não será cobrada nenhuma quantia aos 20 primeiros usuários porque a versão beta será para validar e melhorar o aplicativo.
- O custo definido foi inicial porque será refeito conforme demanda. E estamos negociando entre a equipe para alterar o Redmine pelo Asana e Slack porque são gratuitos.

Custo Total das ferramentas, softwares e licenças utilizadas durante o projeto.

Software/licença	Descrição	Preço anual ou 10 meses
Registro Br	Para registrar o domínio	R$ 40,00
Google Cloud	Hospedagem do Software	$ 26,109

Plataform	por 10 meses.	
Play Store	Conta desenvolvedor para publicar os aplicativos	R$ 0,00
Android Studio	IDE de desenvolvimento para aplicativos android	*R$ 0,00*
Symphony + twig	IDE de desenvolvimento para WEB	*R$ 0,00*
Kotlin	Linguagem de programação para android	*R$ 0,00*
PHP	Linguagem de programação para Web	*R$ 0,00*
PostgreSQL	Banco de Dados	*R$ 0,00*
PGAdmin	Para administrar o banco de dados	*R$ 0,00*
Redmine (easy.redmine.com)	Para controle das atividades e recursos.	$1,416
Conta Gmail	Contas de e-mail para o suporte e equipe.	R$ 0,00
Google Meet	Para reuniões com a equipe remotamente. Usaremos a conta criada acima.	R$ 0,00
Google Calendar	Usaremos para agendar as atividades, ausências, reuniões, etc.	R$ 0,00
app.diagrams.net	Para criar os UMLs, casos de uso, ou qualquer outro diagrama	R$ 0,00
Google Docs	Para documentar e visualizar.	R$ 0,00
Google Drive	Para guardar os documentos e arquivos produzidos.	R$ 0,00
GitHub	Para commit, versionamento, desenvolvimento e disponibilizar para apresentar para o cliente o código e documentos e o próprio software.	R$ 0,00

PDF	Para visualizar a documentação e orçamento.	R$ 0,00
Notebooks, Softwares, internet, espaço, etc.	Iremos utilizar as nossas máquinas pessoais e todos os recursos no CIN e a nossa casa para desenvolver a aplicação.	R$ 0,00

4.2 Plano do Projeto (Entregas da Equipe)

Plano das fases são importantes para definir a entrega dos documentos do projeto. Demais informações sobre as entregas do produto consta no **Plano de Desenvolvimento do EasyToDo.**

4.2.1 Plano de Fase

Nome	Data Inicial	Data Final
Iteração 1		
➢ Apresentação para o Cliente	**02/11/2020**	**02/11/2020**
➢ Reunião com Cliente	**03/11/2020**	**04/11/2020**
➢ Plano de Desenvolvimento	**11/11/2020**	**21/11/2020**
➢ Canvas do Modelo do Projeto	**25/10/2020**	**01/11/2020**
➢ Canvas do Modelo de Negócio	**25/10/2020**	**01/11/2020**
➢ Plano de Gerenciamento de Configuração	**11/11/2020**	**19/11/2020**
➢ Documento de Caso de Uso (UML)	**25/10/2020**	**01/11/2020**
➢ Documento de Requisitos	**25/10/2020**	**01/11/2020**
➢ Protótipo do Sistema (Web e Android)	**25/10/2020**	**01/11/2020**

4.2.2 Objetivos das Iterações

Seguir os critérios (organização das informações em um banco de dados acessível em qualquer lugar via internet e aparelhos específicos para o software) do cliente e usuários projetando o software a atender as suas expectativas que constam detalhados no **Documento de Requisitos do EasyToDo.**

4.2.3 Releases

A primeiro release (entrega) será o MVP. No decorrer das iterações e até a conclusão do projeto haverá releases, que constam detalhadas no GitHub e no **Documento de Desenvolvimento do EasyToDo.**

4.2.4 Recursos do Projeto

Precisaremos da licença do Google Cloud e do Registro BR e alguns aparelhos celulares e notebooks com o mínimo definido no **Documento de Requisitos** para realizar os testes no software Web e APP.

4.3 Controle e Monitoramento da Entrega do Projeto

- **Gerenciamento de Requisitos:** Os mecanismos de controle são as iterações com o cliente de no máximo 2 dias para conseguir seguir de maneira coerente o que foi solicitado ou especificado.
- **Controle de Programação e de Orçamento:** Após os requisitos serem coletados, haverá reuniões entre os integrantes para definir as licenças, hardware, software, tempo (Scrum Poker), prioridades, papéis a serem executados para controlar e monitorar o Projeto definindo o orçamento e as possíveis correções quando forem necessárias.
- **Controle de Qualidade:** Os testes e o protótipo do produto com os feedbacks dos clientes e usuários com inspeção dos analistas na utilização de cada marco da iteração para correção ou alteração antes da entrega do novo protótipo até a conclusão do produto, serão os métodos utilizados para adaptar o produto a eles. Os detalhes constam no **Documento de Controle de Qualidade.**
- **Relatórios e Métricas:** O Andamento, Estabilidade, Adaptabilidade, Modularidade, Qualidade, Maturidade e Perfil de despesas, serão acompanhados através de relatórios de resumo, para controlar e verificar o cronograma das atividades e o orçamento do produto.
- **Gerenciamento de Riscos:** Inicialmente, os maiores riscos levantados foi a questão do tempo e conscientização da equipe, devido ao cronograma ser muito pequeno não será possível entregar alguns documentos obrigatórios na iteração, fica difícil administrar. Para sanar os riscos nós determinamos uma data de entrega para concluir seus respectivos papéis e dar suporte aos demais que tiverem dificuldades. Os detalhes constam no **Documento de gerenciamento de riscos do EasyToDo.**
- **Encerramento do Projeto**: Depois de atender a satisfação dos clientes e a usabilidade do software entregue que foram atendidos de acordo com os requisitos levantados e revisados no decorrer das iterações, não havendo discordâncias ou atualizações e passado o tempo de acompanhamento acordado no contrato, será encerrado o software e iniciará as possíveis manutenções e/ou atualizações e inserções de novas funcionalidades.
- **Gerenciamento de Configuração:** O controle das versões, o acompanhamento dos prazos determinados para a entrega das novas versões, backups, serão feitas de acordo com as solicitações. No documento **Plano de Gerenciamento de Configuração** tem mais detalhes.

- **Resolução de Problemas:** Os métodos será conversar com o cliente para obter mais detalhes, esclarecer, alinhar e convencê-lo que o nosso objetivo é atender às suas necessidades e realizar manutenções de acordo com a necessidade e abordagem do cliente.
- **Treinamento:** Ao entregar o **EasyToDo** além de ter um tutorial explicando como usar, daremos treinamento de como utilizar a aplicação Web e App.
- **Suporte:** Iremos observar os feedbacks dado na Play Store e enviados para o nosso e-mail. E iremos planejar e verificar se deve ou não ser melhoria ou bug e iremos implementar e entregar uma nova versão. Para suporte usaremos um e-mail gratuito do Gmail.