UNIVERSIDADE DE SÃO PAULO (USP)

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO (ICMC)

DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO (SCC)

Gabriel Costa - 14785489

Isabella Arão - 9265732

Marina Fagundes - 9265405

AgroWise

São Carlos

2023

Gabriel Costa - 14785489

Isabella Arão - 9265732

Marina Fagundes - 9265405

AgroWise - Metodologia de Desenvolvimento

Relatório apresentado à disciplina Metodologia de Desenvolvimento de Software, como parte dos requisitos para aprovação na matéria oferecida pela Universidade de São Paulo, na área de Computação.

Prof.: Ellen Francine Barbosa

Disciplina: Metodologia de Desenvolvimento de Software

Turma: Turma SSC 0532

São Carlos

2023

1. Introdução e objetivos

O *AgroWise* é um *software* GIS (Sistema de Informações Geográficas), cujo **objetivo geral** é a coleta de dados por meio de sensores, inserção manual por parte dos usuários e um banco de dados prévio, que visa propor soluções para otimizar o uso da água, detectar precocemente pragas e doenças, monitorar a qualidade do solo, prever safras e melhorar a eficiência energética, as condições climáticas e os impactos ambientais.

A tecnologia GIS permite o mapeamento das áreas de cultivo, a identificação dos tipos de cultura e o monitoramento do crescimento de plantas. Juntamente às informações geoespaciais de topografia, clima, características físico-químicas do solo, consegue-se analisar detalhadamente o rendimento de colheitas e seus padrões de produtividade, o manejo de recursos e, dessa forma, apoiar a tomada de decisões.

O *AgroWise* tem como **objetivos específicos**:

1) O **monitoramento da qualidade do solo**;

2) O **monitoramento de condições climáticas** e integração do sistema com o INMET (Instituto Nacional de Meteorologia);

3) A **predição de safras**, objetivando ajudar na previsão de plantio e de colheita e estimativa de rentabilidade e de perdas;

4) Criação de um **calendário** que sintetiza todas as informações relevantes geradas pelos módulos do *software*;

5) A **detecção precoce de pragas e doenças**, visando avaliar as pragas e doenças mais comuns para cada planta e o estágio de crescimento da planta e o seu estado;

6) A **otimização do uso de água**, as previsões de chuvas e a quantidade de água que seria possível coletar durante esses eventos, além de contabilizar o consumo de água;

7) O **monitoramento de eficiência energética**, a quantificação do uso de energia para cada processo e a avaliação sobre o gasto de cada insumo e da infraestrutura a médio e longo prazo;

8) O **monitoramento de impactos ambientais**, para auxiliar na avaliação e diminuição dos impactos ambientais causados pela agropecuária no ecossistema através de gráficos diários e relatórios;

Cada um desses objetivos foi traduzido em uma funcionalidade do sistema, as quais estão explicitadas mais adiante (tópico 6).

2. Metodologia

Na primeira entrega do projeto, o grupo que idealizou o *AgroWise* adotou a **metodologia incremental**, dividida em sete fases, em que cada uma é finalizada com a entrega de uma funcionalidade ou incremento: **(i)** planejamento inicial, **(ii)** desenvolvimento do *software*, **(iii)** testes e *feedback*, **(iv)** expansão de módulos, **(v)** integração e validação completa, **(vi)** lançamento e avaliação real e, por fim, **(vii)** manutenibilidade.

Esse modelo combina elementos dos modelos tradicionais cascata e prototipação. Do primeiro, utiliza as definições de fases de projeto, mas com a possibilidade de voltar a fases anteriores se necessário. Do segundo, utiliza a filosofia iterativa. O modelo incremental propõe a união de forças entre equipe e cliente para que os requisitos sejam coletados da melhor forma, além de elencar as funcionalidades que deverão ter maior importância. Após isso, serão definidos estágios de entrega, em que cada estágio fornece um subconjunto das funcionalidades do sistema como um todo, para que seja entregue ao cliente uma versão executável, que poderá ser testada pelo usuário e então ter os feedbacks necessários.

Já na segunda entrega, escolhemos o ***Scrum***, metodologia ágil que permite as entregas também sejam feitas de maneira incremental, com módulos entregues ao final de cada *sprint* de forma a gerar valor para o cliente antes mesmo que o sistema seja concluído como um todo. Dessa forma, a metodologia permite que o projeto se adapte melhor a mudanças conforme o desenvolvimento avança e os módulos a serem desenvolvidos podem ser organizados de acordo com a prioridade e o risco para o sistema final. Além disso, o *Scrum* é marcado pelo *feedback* contínuo das partes interessadas e pelo envolvimento ativo e maior integração entre os integrantes do time *Scrum*.

Nesta terceira entrega, adotamos um modelo híbrido, que busca mesclar os pontos positivos das duas metodologias anteriores. Suas características principais estão listadas a seguir:

1) Há uma **junção da divisão em fases e da divisão em *sprints***. Nas fases iniciais, as *sprints* são estruturadas de maneira mais detalhada, o que reflete as características de metodologias tradicionais: maior detalhe na definição dos requisitos e do planejamento, por exemplo. Nesse primeiro momento, a equipe será

reduzida e aumentará nas etapas seguintes, uma característica típica do método incremental. Ao decorrer do processo, as *sprints* incorporarão mais as características do *Scrum*: foco na agilidade, maior contato com o cliente e *feedbacks* contínuos, maior flexibilidade, entre outros;

2) Do modelo incremental, adotaremos uma **documentação mais detalhada**, inclusive adicionando mais tempo para essa etapa nas *sprints* presentes no cronograma (tópico 8). Essa documentação detalhada contribuirá para a manutenção do sistema a longo prazo e para sua melhor compreensão. Além disso, destacamos a **importância dos testes**, presente em ambas as metodologias;

3) A **priorização pode ser dinâmica**, se necessário. Essa característica se alinha ao *Scrum* e proporciona maior flexibilidade. Isso contribui para maior flexibilidade e adaptação a possíveis mudanças ao longo do desenvolvimento;

4) Também do *Scrum*, priorizamos o **contato com o cliente e os *feedbacks* contínuos**, em especial durante o planejamento e ao final de cada *sprint*, através das *sprints reviews*, que contam com a participação de diferentes partes interessadas e buscam avaliar os pontos positivos e negativos da *sprint* em relação às questões técnicas. Adicionalmente, durante as *sprints* que julgamos necessárias, adicionamos uma atividade para **atualizar o cliente através de *e-mails*** e receber seu *feedback*, evitando possíveis problemas e melhorando a comunicação entre o time e os clientes;

5) Os eventos de avaliação das *sprints* permitirão a **avaliação contínua da metodologia e monitoramento do progresso**, que possibilitam ajustes para melhorar o seu desempenho e aprendizado contínuo por parte dos membros da equipe e contribuem para a melhor adaptabilidade do método;

Os métodos incremental e *Scrum* já apresentam **várias características em comum**, como o desenvolvimento do projeto através de incrementos. O método híbrido aqui adotado, portanto, explora essas semelhanças e mescla as características distintas que melhor se encaixem ao nosso processo de desenvolvimento de *software*, equilibrando a estruturação detalhada e rigorosa dos métodos tradicionais e a flexibilidade e adaptabilidade dos métodos ágeis.

3. Escopo

A fim de melhorar a tomada de decisão ao longo do processo de desenvolvimento do programa, é necessário traçar limites para ele, especificando com mais detalhes o escopo do projeto. Em função disso, a seguir, deixamos listado tudo o que julgamos ser coerente para estar dentro do escopo e o que estará fora dos limites do projeto.

3.1. Dentro do escopo

O sistema concentra-se na coleta de dados geográficos que irão auxiliar nos processos de negócios de uma fazenda, auxiliando no planejamento de safras e aumentando a produtividade agrícola. Logo, dentro dos limites do projeto, destacamos:

1) Centralização e gerenciamento das informações coletadas;
2) Integração com sistema de previsão climática;
3) Registro e armazenamento de dados de sensores em tempo real;
4) Visualização de gráficos e relatórios sobre as condições de clima e de solo nas áreas de plantio;
5) Previsão de safras;
6) Recomendação de culturas a serem plantadas (e em que época do ano);
7) Base de dados centralizada;

3.2. Fora do escopo

Assim como é de extrema importância destacar o que está dentro do escopo do projeto, também é essencial destacar o que estará fora dele, pois, dessa forma, o desenvolvimento pode seguir mais focado e com as prioridades elencadas desde o início. Os itens que estão fora do escopo estão listados a seguir:

1) Integração com sistemas legados que possam vir a existir;
2) Integração com sistema de irrigação e maquinário específico automatizado;
3) Recursos de personalização da interface do usuário;
4) Funcionalidades de gerenciamento de estoque e inventário de insumos agrícolas;
5) Configuração de alertas/notificações que são disparados a partir de limites que são previamente definidos para os sensores;
6) Gestão de recursos humanos: assuntos relacionados a funcionários, folha de pagamento etc.

4. Identificação dos riscos e estratégias

O projeto de um *software* envolve muitas variáveis que podem apresentar riscos ao desenvolvimento do programa. Os **riscos gerais** que mais se destacam, considerando a adoção de uma metodologia híbrida, estão listados a seguir, juntamente com a estratégia mais adequada para mitigar tais ameaças.

1) **Definição de requisitos:** nesta etapa de desenvolvimento, buscamos adotar práticas positivas dos dois métodos, o tradicional e o ágil. Assim, dedicamos **mais tempo para a definição de requisitos**, visando começar o projeto com maior clareza de seus objetivos e, assim, eliminar mudanças evitáveis. Entretanto, as práticas de metodologias ágeis ajudam a fazer com que a **equipe esteja melhor preparada para lidar com possíveis alterações ao longo**, sem que isso signifique desperdício de recursos e esforço considerável. Portanto, apesar de dedicar mais tempo na definição de requisitos, a metodologia é capaz de se adaptar e acolher possíveis mudanças que sejam necessária ao longo do projeto;

2) **Prazos não cumpridos:** a perda de prazos pode causar, também, altos custos ao projeto, o que pode ter um impacto negativo sobre a equipe e sobre a relação com o cliente. Uma estratégia a ser colocada em prática para que isso não aconteça é a **definição de um cronograma com folgas e que anteveja possíveis imprevistos antes do começo do projeto**, elencando quanto tempo cada fase terá e o que será feito em cada uma delas. Além disso, deve-se ter atenção e monitoramento em todas as etapas, para se ter controle do andamento do projeto;

3) **Orçamento excedido:** exceder o orçamento original é algo que pode gerar muitos impactos negativos no projeto. Para que isso não ocorra, é necessário fazer um **planejamento financeiro detalhado**, com previsão do que será gasto em cada etapa e que inclua margens de erros, a fim de que o orçamento seja respeito do início ao fim;

4) **Qualidade do código:** a qualidade do código é de fundamental importância, pois é um dos pilares para se ter um *software* de qualidade. Para que não haja riscos da qualidade do código ser impactada, devem ser feitas **constantes revisões** em cada estágio de entrega, para que se tenha conhecimento do que foi feito até então. Além disso, uma **boa documentação**, típica do modelo tradicional, será uma prática adotada

nessa metodologia híbrida, pois é essencial para que erros não sejam carregados para etapas futuras. Por fim, o ***feedback* contínuo** também auxilia no controle de qualidade;

5) **Dificuldades com o uso do novo sistema:** implantar um novo sistema é desafiador, pois as novas funcionalidades podem não ser tão intuitivas a princípio. Para que isso não se torne um grande problema, os **usuários finais do *software* devem ser treinados** para o uso do novo sistema. Em complemento à capacitação, é essencial uma **documentação completa** de todas as funcionalidades para que os usuários tenham acesso e possam consultar se ocorrer dúvidas. E, por fim, também é muito importante o **suporte da equipe de desenvolvimento pós-implantação**, dando o apoio necessário;

Estes são os principais riscos elencados que podem vir a comprometer o desenvolvimento e implantação de um novo software. Além do que foi listado, é necessário prever uma **boa gestão de riscos**, para que quando um novo risco for identificado, a gerência do projeto ter condições, juntamente com a equipe, de tomar a melhor decisão para que o projeto não seja prejudicado.

Além dos citados acima, temos **riscos específicos para o sistema**, apresentados a seguir:

1) Garantir a **integridade dos sensores eletrônicos** do sistema proposto: existem diversas vulnerabilidades desses dispositivos a condições ambientais adversas (variações de temperatura, umidade elevada, poeira, entre outras) que podem comprometer seu funcionamento e vida útil e, portanto, **os dispositivos devem ser protegidos**. Além disso, a proteção deve evitar que os aparelhos sofram danos causados por animais, como roedores e aves, que podem roer cabos e componentes eletrônicos;

2) O **monitoramento contínuo exige uma fonte de energia constante**, o que pode ser um problema em locais remotos ou de difícil acesso. **Obstáculos físicos** podem impactar na eficácia da coleta de dados ao limitar o alcance dos sensores. Assim, deve-se providenciar uma **fonte de energia sustentável** para alimentar esses dispositivos e **tomar cuidados no seu posicionamento**, de forma a, na medida do possível, evitar obstáculos;

3) A **dificuldade de gerenciamento de grandes volumes de dados** gerados pelos sensores torna necessário **o uso de técnicas de análise avançada** para garantir a qualidade dos relatórios gerados.

5. Recursos necessários

5.1. Recursos humanos

A fase inicial do projeto, terá uma equipe mais enxuta, priorizando o planejamento e a documentação da solução.

Nesta fase, os desenvolvedores, principalmente, arquitetos de software, juntamente com o *Product Owner* e os *stakeholders* desempenharão um papel crucial para a construção do projeto, criando o plano de ação, entendendo as necessidades dos clientes e usuários, delimitando cada elemento da solução e antecipando potenciais desafios.

Com o plano de ação sólido, o projeto avança para a fase de implementação das *sprints* de desenvolvimento e aqui, a equipe aumenta, sendo utilizado todos os atores do *Scrum*, como: *Product Owner*, *Scrum Master*, Time de Desenvolvimento e *Stakeholders*.

5.1.1. Product Owner

1) Representa os interesses dos clientes, podendo até mesmo ser o cliente;

2) Mantém claro os objetivos do produto;

3) Decide o que será entregue e quando será entregue;

4) Fornece atualizações dos itens do *backlog* nas reuniões da *Sprint*;

5) Interage com os *stakeholders* e clientes;

6) Aceita ou rejeita os resultados do trabalho na *Sprint Review*.

5.1.2. Scrum Master

1) Garante que o time se oriente pelos valores do *Scrum*;

2) Mantém a equipe funcional e produtiva;

3) Protege o time de interferências externas, como solicitações de mudanças;

4) Remove possíveis impedimentos para o progresso do time.

5.1.3. Time de desenvolvimento

1) De 5 a 9 pessoas;

2) Auto-organizado e multidisciplinar (desenvolvedores, arquitetos, testadores, UX/UI *designers*);

3) Transforma os requisitos em um entregável;

4) Mantém transparência sobre o progresso do trabalho diário.

5.1.4. Stakeholders

1) Podem participar das cerimônias;

2) Fornecem *feedback* em relação ao trabalho da equipe;

3) Auxiliam em atividades específicas de histórias.

5.2. Hardware

1) Sensores: é necessária a adesão de sensores responsáveis pela captação dos dados para processamento;

2) Servidor ou armazenamento em nuvem: é preciso ter um servidor dedicado para hospedar o sistema e o banco de dados, ou utilizar serviços de armazenamento em nuvem confiáveis;

3) **Computador:** é necessário máquinas com sistema operacional *MacOS* ou *Windows*, bem como espaço em disco suficiente para armazenar o *software* juntamente com informações relacionadas;

4) **Dispositivo móvel:** para implementação mobile é necessário dispositivos *Android* ou *IOS* com armazenamento disponível para a utilização do aplicativo desenvolvido.

5.3. Software

1) **Linguagens de programação e Frameworks:** para o desenvolvimento da solução, podendo ser: *Java*, *C#*, *JavaScript*, *Python*, *ReactJS*, *NextJS*, *NodeJS*, *Spring* e *React Native* ou *Flutter* no caso de desenvolvimento *mobile*;

2) **Ambiente de desenvolvimento:** editores de código ou IDEs como *Visual Studio*, *Visual Studio Code*, *IntelliJ*, dentre outros;

3) **Banco de dados:** sistema de gerenciamento de banco de dados como *MySQL*, *PostgreSQL*, SQL Server, dentre outros;

4) **Ferramentas de testes:** para a realização de diferentes tipos de testes do software, como o *JUnit, Selenium, Cucumber*, dentre outros.

5) **Ambiente de virtualização:** soluções para testar o projeto em diferentes ambientes, como o *VMware* ou o *VirtualBox*;

6) **Ferramentas de Controle de Versão:** para gerenciar o código fonte, podendo ser o *GitHub*;

7) **Ferramentas de Gerenciamento de Projeto:** soluções para registrar e rastrear as tarefas e progresso, como o *Jira*, *Trello*, *Stack* e outros.

5.4. Outros

1) **Serviços de comunicação interna:** para facilitar a comunicação entre as equipes, podendo utilizar soluções como: *Teams*, *Zoom*, *Skype*;

2) **Recursos financeiros:** Orçamento para custos relacionados ao desenvolvimento, como licenciamento de software, hospedagem e demais custos operacionais.

6. Requisitos/funcionalidades

Conforme explicado na parte 2 do presente projeto, o *AgroWise* é dividido em sete funcionalidades principais:

1) O **monitoramento da qualidade do solo** gera informações e previsões com alguns dados fornecidos por dois sensores inteligentes: o sensor de PH (mede o nível de acidez e de alcalinidade do solo de forma a analisar suas condições químicas, que podem afetar o crescimento, o desenvolvimento e a qualidade da safra) e o sensor de umidade, temperatura e condutividade elétrica do solo (a partir dos dados coletados, é possível gerar relatórios e mapas de colheita para apoiar as decisões dos responsáveis);

2) O **monitoramento de condições climáticas** será feito por sensores inteligentes, que medem velocidade e direção dos ventos, radiação solar, temperatura e umidade do ar. Além disso, o sistema será integrado com o INMET (Instituto Nacional de Meteorologia), para medir a possibilidade de chuvas nos próximos sete dias;

3) A **predição de safras** se baseia na avaliação do tipo de cultura e dos dados coletados nos dois itens anteriores. Em seguida, geram-se relatórios de previsão de plantio e de colheita e estimativa de rentabilidade e de perdas;

→ O **calendário** sintetiza todas as informações relevantes geradas pelos módulos do *software*, incluindo: prazos para plantio e colheita, cronogramas de irrigação, cronogramas de solo;

4) A **detecção precoce de pragas e doenças*** necessita de dados históricos, semelhantes aos usados na predição de safras, de colheitas anteriores para gerar informações comparativas para o banco de dados, que também contará com informações gerais sobre diferentes plantios, coletados através de especialistas. Com esses dados, serão gerados dois relatórios, um preditivo (sobre as pragas e doenças mais comuns para cada planta) e um comparativo (sobre o estágio de crescimento da planta e o seu estado);

5) A **otimização do uso de água*** proporciona a criação de relatórios que detalham o uso diário de água para cada tipo de plantio, levando em consideração os equipamentos a serem usados e a vazão. Além disso, esses documentos contam com as previsões de chuvas e a quantidade de água que seria possível coletar durante esses eventos. Por último, gráficos que mostram o consumo de água em cada estágio e o consumo acumulado até o momento são gerados;

6) O **monitoramento de eficiência energética*** verifica o consumo de energia dos insumos agrícolas através de medidores instalados em aparelhos e sistemas de energia. Assim, são gerados gráficos diários do uso de energia para cada processo e um relatório detalhado que avalia individualmente o gasto de cada insumo e da infraestrutura a médio e longo prazo;

7) O **monitoramento de impactos ambientais***, por fim, auxilia na avaliação e diminuição dos impactos ambientais causados pela agropecuária no ecossistema através de gráficos diários e relatórios. Os dados físico-químicos necessários são obtidos por sensores, Espectrofotômetro UV-Visível, biossensores e especialistas (biólogos e agrônomos);

* A versão inicial do *software* inclui os três primeiros módulos, enquanto os demais podem ser adquiridos conforme a necessidade e o orçamento do cliente.

Para cada funcionalidade, listamos cinco histórias de usuário distintas, apresentadas no tópico a seguir.

6.1. Histórias de usuários

6.1.1. Monitoramento da qualidade do solo

1) Como **agricultor**, preciso receber notificações automáticas sobre as informações físico-químicas do solo para garantir ajustes nos tratamentos de correção do solo, na aplicação de tratamentos preventivos e na aplicação de nutrientes, o que facilita recomendações precisas;

2) Como **gerente de plantação**, quero receber relatórios periódicos que detalham as condições e características físico-químicas do solo, que me proporcionarão maior controle da produção;

3) Como **usuário do sistema**, desejo gerar mapas de colheita com informações físico-químicas para identificar padrões de crescimento e maturação de culturas;

4) Como **especialista**, desejo ter acesso a dados históricos sobre a qualidade do solo para análises comparativas e identificação de tendências;

5) Como **investidor**, preciso ter relatórios que destaquem a condição do solo com precisão, de modo a me auxiliar a tomar decisões de investimento embasadas em dados confiáveis e científicos.

6.1.2. Monitoramento de condições climáticas

1) Como **gerente de plantações**, preciso de alertas automatizados sobre mudanças climáticas que impactarão a plantação, para que eu possa proporcionar a proteção necessária;

2) Como **agricultor**, necessito de relatórios para decidir o melhor momento para o plantio, com base nas condições climáticas ideais;

3) Como **proprietário**, desejo relatórios que identifiquem áreas suscetíveis a ventos mais intensos, para que eu possa implementar medidas de proteção;

4) Como **especialista**, necessito de dados precisos sobre as condições climáticas, que me permitam estudar seus efeitos no crescimento e na evolução de diferentes culturas;

5) Como **produtor**, preciso de previsões sobre as chuvas de modo a otimizar a irrigação e evitar o desperdício de água.

6.1.3. Predição de safras

1) Como **gerente**, preciso de relatórios periódicos que me informem quais culturas foram mais rentáveis, de forma a apoiar o meu planejamento a longo prazo;

2) Como **agricultor**, desejo previsões de plantio e colheita para planejar a distribuição e a venda desses produtos;

3) Como **fornecedor**, desejo previsões de plantio para ajustar minha previsão para a demanda do meu cliente;

4) Como **especialista**, preciso de relatórios sobre rentabilidade das safras para estudos de longo prazo e para desenvolvimento de técnicas agrícolas inovadoras;

5) Como **proprietário**, gostaria de ter um relatório que estimasse perdas de forma precisa, visando desenvolver estratégias para diminuir tais perdas e otimizar a produção.

6.1.4. Síntese: calendário

1) Como **gerente**, preciso de um cronogramas personalizados para cada cultura, de forma a auxiliar no gerenciamento;

2) Como **agricultor**, necessito de um calendário que destaque os períodos ideais para o plantio e colheita de diferentes culturas;

3) Como **proprietário**, preciso de um calendário que inclua lembretes para a manutenção de insumos, como equipamentos e máquinas agrícolas;

4) Como **planejador**, preciso de um calendário que inclua as datas de colheitas diferentes para planejar a logística da propriedade;

5) Como **investidor**, preciso de um calendário que inclua os períodos de descanso do solo e de rotação de culturas, de forma a otimizar o rendimento;

6.1.5. Detecção precoce de pragas e doenças

1) Como **agrônomo**, preciso de alertas automatizados sobre o surgimento de pragas específicas em determinadas culturas para realizar as intervenções de forma precisa;

2) Como **gerente**, gostaria de ter acesso a um comparativo do estado de saúde das plantas em áreas diferentes para identificar os padrões de desenvolvimento e riscos possíveis e específicos;

3) Como **agricultor**, desejo receber notificações sobre possíveis doenças específicas que podem surgir em um tipo de cultivo, para poder realizar as medidas preventivas necessárias;

4) Como **especialista**, necessito de dados históricos sobre pragas e doenças para identificar padrões e sazonalidades;

5) Como **fornecedor**, desejo ter acesso a relatórios preditivos sobre a possibilidade de pragas ou doenças, para ajustar a produção de defensivos;

6.1.6. Otimização do uso de água

1) Como **gerente**, preciso de relatórios periódicos sobre o consumo de água por diferentes tipos de cultura, visando ajustar o programa de irrigação;

2) Como **agricultor**, necessito de previsões de chuvas para otimizar o armazenamento de água e aproveitar esse recurso;

3) Como **proprietário**, desejo gráficos que mostrem a relação entre o consumo de água e o estágio de crescimento das plantas, de forma a ajustar a estratégia de irrigação;

4) Como **especialista**, preciso de dados históricos sobre o consumo de água em diferentes culturas, para identificar tendências e padrões;

5) Como **investidor**, queria ter acesso a relatórios sobre a viabilidade econômica do uso de métodos de captação de água da chuva para a irrigação.

6.1.7. Monitoramento de eficiência energética

1) Como **gerente**, preciso de relatórios periódicos sobre o consumo de energia em diferentes operações para identificar áreas de alto consumo e focar esforços para aumentar a eficiência energética nelas;

2) Como **técnico**, necessito de alertas automáticos sobre o consumo excessivo de energia em equipamentos, de forma a avisar sobre a necessidade de manutenção preventiva;

3) Como **proprietário**, busco relatórios sobre o consumo de energia ao longo do tempo para ajustar as práticas de uso;

4) Como **agricultor**, preciso de relatórios sobre o consumo de energia por unidade de área em diferentes métodos de cultivo, que me ajudem a selecionar práticas mais sustentáveis;

5) Como **investidor**, queria ter acesso a relatórios comparativos de eficiência energética entre diferentes fazendas, para apoiar a minha decisão de investir em uma delas.

6.1.8. Monitoramento dos impactos ambientais

1) Como **gerente**, preciso de relatórios periódicos sobre as emissões de gases do efeito estufa provenientes das operações agrícolas, visando implementar estratégias de redução;

2) Como **especialista**, necessito de dados sobre a biodiversidade em áreas agrícolas para entender os impactos causados por essa atividade;

3) Como **proprietário**, desejo alertas sobre a degradação do solo para implementar técnicas de recuperação o mais rápido possível;

4) Como **agricultor**, preciso de relatórios sobre a saúde do ecossistema em torno da fazenda, objetivando garantir práticas agrícolas mais sustentáveis;

5) Como **investidor**, queria ter acesso a relatórios comparativos sobre os impactos ambientais entre diferentes fazendas, para apoiar a minha decisão de investir em uma delas.

6.2. Backlog

A partir da criação das histórias de usuário, o *backlog* de atividades e *sprints* foi desenvolvido a fim de elencar as prioridades e o tempo definido para desenvolvimento de cada uma das funcionalidades, além da correção de eventuais erros que possam vir a ocorrer. Além do que já foi desenvolvido na entrega 2, com a metodologia *Scrum,* adicionaremos o que será necessário para a metodologia híbrida, que juntará a metodologia ágil com a tradicional Incremental.

Os itens elencados no *backlog* foram os seguintes:

1) Fase inicial:

a) Planejamento necessário;
b) Coleta de requisitos juntamente com as partes interessadas;
c) Prototipação inicial da interface;
d) Codificação inicial do sistema;
e) Documentação extensa e completa de todos os requisitos, funcionalidades e detalhes de interface que foram planejados;
f) Report para o cliente via reunião.

→ Entregável: versão inicial do sistema para testes;

2) Monitoramento da qualidade do solo:
a) Implementação de notificações automáticas para auxiliar o agricultor na correção de solo;
b) Implementação da geração de relatórios das condições do solo (para os gerentes de plantação e *stakeholders*);
c) Implementação de mapas de colheita com informações sobre o solo;
d) Implementação de banco de dados com dados históricos das condições do solo;
e) Documentação de tudo que foi feito em cada *sprint*;
f) *Report* para o cliente via *e-mail*.

→ Entregáveis: incremento do sistema com notificações e geração de relatórios;

3) Monitoramento de condições climáticas:
a) Implementação dos alertas automáticos sobre mudanças climáticas;
b) Implementação de geração de relatórios sobre o momento ideal para o plantio;
c) Especificação de relatórios com identificação de áreas suscetíveis a ventos intensos e com previsões de chuva;
d) Integração do sistema *AgroWise* com sistema INMET;
e) Documentação de tudo que foi feito em cada *sprint*;
f) *Report* para o cliente via *e-mail*.

→ Entregáveis: incremento do sistema com novos alertas e novos tipos de relatórios;

4) Predição de safras:
a) Implementação de relatórios para gerentes com as culturas mais rentáveis;
b) Implementação de funcionalidade para previsão de plantio e colheita;

c) Implementação de relatórios de financeiro da safra (venda de produtos e taxa de perda);
d) Documentação de tudo que foi feito em cada *sprint*;
e) *Report* para o cliente via *e-mail*.

→ Entregáveis: incremento do sistema com novas funcionalidade para previsão do andamento da safra, além de mais tipos de relatórios disponíveis para auxílio do negócio;

5) Calendário:
a) Implementação de cronogramas personalizados de cada cultura;
b) Implementação de calendário indicando períodos ideias para plantio e colheita;
c) Implementação de notificações para manutenção de insumos;
d) Implementação de calendário indicando períodos de descanso do solo e rotação de culturas;
e) Documentação de tudo que foi feito em cada *sprint*;
f) *Report* para o cliente via *e-mail*.

→ Entregáveis: incremento do sistema com novos alertas, além de calendários que auxiliarão na tomada de decisão.

7. Fases e atividades

O desenvolvimento do programa contará com uma mescla de características das metodologias Incremental e *Scrum*. Por isso, as fases e atividades são compostas levando em consideração esses dois métodos de desenvolvimento.

Mantivemos a estrutura das *sprints*, com duração de 4 semanas por *sprint*, porém dentro de cada uma delas, adicionamos algumas atividades, que estão listadas no backlog, e que são importantes para o desenvolvimento de acordo com a metodologia Incremental.

A fase inicial de planejamento agora é a mais longa, com 6 *sprints*, contando com maior prazo para a coleta de requisitos e para que a documentação possa ser feita com maiores detalhes. Além disso, adicionamos o reporte com o cliente, via reunião presencial nessa primeira etapa, para mantê-lo atualizado durante o processo de desenvolvimento e ter maior *feedback* durante o planejamento, além da reunião de *sprint review*, que já estava presente. Ao final dessa fase, há uma revisão dos requisitos e documentação final deles, a fim de que possam ser revistos um a um e colocados em ordem de prioridade.

As próximas fases do cronograma não tiveram alterações de tamanho. Porém, para que pudéssemos incorporar mais características do modelo Incremental, adicionamos tarefas de documentação em todas as *sprints*, além dos reportes periódicos para o cliente.

O cronograma em anexo exibe o planejamento de desenvolvimento completo, com quantidade de *sprints* em cada etapa e atividades por *sprint*.

8. Referências bibliográficas

BARBOSA, E. F. Material didático apresentado na disciplina SCC0532 - Metodologia de Desenvolvimento de Software. São Carlos/SP. s.d..

9. Anexo

Planejamento

Fases/Funcionalidades	Sprint planning	Duração	Sprint 01	Sprint 02	Sprint 03	Sprint 04	Sprint 05	Sprint 06	Incremento	Sprint review	Sprint retrospective
Planejamento	8 horas	6 sprints (4 semanas cada)	Planejamento inicial Coleta e análise dos requisitos juntamente com o cliente Documentação das fases do planejamento Documentação dos requisitos gerais	Coleta de requisitos necessários para integração com INMET Documentação dos requisitos de integração	Fluxograma para prototipação da interface Prototipação da interface Documentação dos requisitos de interface **Reunião com o cliente**	Início da codificação geral do sistema Documentação da codificação	Revisão de requisitos Documentação final dos requisitos	Testes iniciais	Versão inicial da interface *Feedback* do cliente	5 horas	3 horas
Monitoramento da qualidade do solo	8 horas	5 sprints (4 semanas cada)	Implementação de banco de dados com dados históricos das condições do solo Implementação de funcionalidade para inserção de dados sobre o solo manualmente Documentação	Implementação de geração de relatórios das condições do solo Implementação de mapas de colheita com informações sobre o solo Documentação	Implementação de notificações de correção de solo Documentação	Implementação de geração de relatórios das condições do solo para fins de investimento Incremento da interface do sistema **Reporte via e-mail para o cliente**	Testes Pendências		Incremento do sistema para uso *Feedback* do cliente	5 horas	3 horas
Monitoramento de condições climáticas	8 horas	4 sprints (4 semanas cada)	Integração com sistema INMET Implementação de alertas automáticos sobre mudanças climáticas Documentação de requisitos do INMET	Implementação de mapas com as áreas com ventos mais intensos Implementação de funcionalidade que preveja chuvas Documentação	Implementação de geração de relatórios com condições climáticas Implementação de geração de relatórios com melhores momentos para plantio Documentação **Reporte via e-mail para o cliente**	Testes Pendências Incremento da interface do sistema			Incremento do sistema para uso *Feedback* do cliente	5 horas	3 horas
Predição de safras	8 horas	3 sprints (4 semanas cada)	Implementação de geração de relatórios com culturas mais rentáveis Implementação de funcionalidade para previsão de plantio e colheita Implementação de geração de relatórios de venda de culturas Documentação	Implementação de geração de relatórios com taxa de perda de cada cultura Incremento da interface do sistema Documentação **Reporte via e-mail para o cliente**	Testes Pendências				Incremento do sistema para uso *Feedback* do cliente	5 horas	3 horas
Calendário	8 horas	3 sprints (4 semanas cada)	Implementação de cronogramas para cada cultura Implementação de um calendário com períodos de plantio e colheita Implementação de calendário com períodos de descanso do solo Documentação	Implementação de notificações para manutenção de insumos Implementação de cronograma com rotação de culturas Documentação **Reporte via e-mail para o cliente**	Testes Pendências				Incremendo sistema e entrega da versão final *Feedback* do cliente	5 horas	3 horas