1. Registo de Tempo

Com Linguagem Natural

1. **O contabilista diz:** "Hoje passei 2 horas a preparar a declaração IVA do cliente ABC e 30 minutos numa reunião com o cliente XYZ."
2. **O sistema:**
    - O texto é processado pelo serviço associado à classe NLPProcessor
    - Identifica os clientes (Client) "ABC" e "XYZ" através de consulta à base de dados
    - Reconhece as atividades e associa a objetos TaskCategory ("declaração IVA" e "reunião")
    - Extrai os tempos (2 horas e 30 minutos)
    - Cria dois objetos TimeEntry distintos:
        - TimeEntry 1: client=ABC, minutes_spent=120, description="preparar a declaração IVA", original_text="..."
        - TimeEntry 2: client=XYZ, minutes_spent=30, description="reunião", original_text="..."
    - Calcula o campo monetary_value com base no hourly_rate do objeto Profile do utilizador atual
    - Atualiza registos em ClientProfitability para os clientes afetados

Com Rastreamento Automático

1. **O contabilista:** Clica em "Iniciar rastreamento" quando começa a trabalhar numa tarefa
2. **O sistema:**
    - Cria um registo AutoTimeTracking com user=utilizador_atual, start_time=agora
    - Começa a armazenar activity_data (JSON) sobre a atividade
3. **O contabilista:** Clica em "Parar" quando termina
4. **O sistema:**
    - Atualiza o registo AutoTimeTracking com end_time=agora
    - Usa o NLPProcessor para analisar a atividade registada
    - Sugere valores para os campos client, task e category
    - Após confirmação, cria um registo TimeEntry e marca AutoTimeTracking.processed=True

2. Gestão de Tarefas e Planeamento

Registando Tarefas Pendentes

1. **O contabilista diz:** "Preciso de entregar a declaração IVA do cliente ABC até sexta-feira, rever os extratos do cliente XYZ com urgência média, e preparar a folha de pagamento do cliente DEF até dia 10."
2. **O sistema:**
    - Processa o texto com NLPProcessor.process_text()
    - Cria três objetos Task:
        - Task 1: client=ABC, title="Declaração IVA", deadline=sexta-feira, priority=1
        - Task 2: client=XYZ, title="Revisão de extratos", priority=3
        - Task 3: client=DEF, title="Folha de pagamento", deadline=dia_10, priority=2
    - Consulta o histórico de TimeEntry para tarefas similares e preenche o campo Task.estimated_time_minutes
    - Associa cada tarefa à TaskCategory apropriada

Planeamento Diário

1. **O contabilista:** Acede à visão "O Meu Dia" no início da manhã
2. **O sistema:**
    - Consulta objetos Task deste utilizador com status="pending" ou "in_progress"
    - Verifica objetos TeamWorkload para analisar a capacidade do utilizador
    - Usa algoritmo que considera Task.priority, Task.deadline e Client.importance para ordenação
    - Exibe tarefas ordenadas e gera um objeto AIInsight com sugestões de planeamento
    - Cria ou atualiza um objeto TeamWorkload para o dia atual

Execução de Tarefas

1. **O contabilista:** Seleciona uma tarefa para trabalhar
2. **O sistema:**
    - Atualiza o objeto Task.status para "in_progress"
    - Verifica se há objetos Document relacionados à Task ou Client
    - Oferece opção de iniciar um novo objeto AutoTimeTracking
3. **O contabilista:** Marca a tarefa como concluída

4. **O sistema:**
    - Atualiza Task.status para "completed" e Task.completed_at para timestamp atual
    - Verifica se a tarefa está associada a algum WorkflowStep
    - Se sim, avança para o próximo passo no workflow usando a relação WorkflowStep.next_steps
    - Gera uma Notification para utilizadores relevantes

3. Processamento de Documentos

Receção e Classificação

1. **O cliente:** Envia um documento via email ou WhatsApp para o escritório
2. **O sistema:**
    - A mensagem é recebida através de um objeto IntegrationChannel configurado
    - IntegrationChannel.process_incoming() extrai o documento anexado
    - Cria um novo objeto Document com source="email" ou source="whatsapp"
    - Utiliza IA para identificar o cliente e atualiza Document.client
    - Processa o conteúdo e preenche Document.extracted_data (JSON)
    - Marca Document.is_processed como True quando concluído
    - Se forem detetadas despesas, cria registos Expense associados

Utilização de Documentos

1. **O contabilista:** Pesquisa documentos de um cliente
2. **O sistema:**
    - Consulta objetos Document filtrando por Document.client
    - Apresenta documentos agrupados por Document.file_type e Document.upload_date
3. **O contabilista:** Seleciona um documento para processamento contabilístico
4. **O sistema:**
    - Exibe os dados extraídos de Document.extracted_data
    - Gera sugestões usando AIInsight baseadas no conteúdo e padrões históricos

4. Análise de Rentabilidade e Alertas

Monitorização Contínua

1. **O sistema constantemente:**
    - Acumula registos TimeEntry filtrados por cliente e período
    - Consulta Profile.hourly_rate dos profissionais para calcular custos
    - Atualiza registos em ClientProfitability
    - Executa ClientProfitability.calculate_profit() para recalcular lucro e margens

Alertas Automáticos

1. **Quando um cliente se torna não rentável:**
    - O sistema deteta que ClientProfitability.is_profitable mudou para False
    - Cria um objeto Notification com:
        - user=account_manager do cliente
        - related_client=cliente em questão
        - notification_type="profitability_alert"
    - Envia a notificação através de Notification.send()
2. **O gestor:** Acede ao dashboard de rentabilidade
3. **O sistema:**
    - Consulta registos ClientProfitability, TimeEntry e Expense do cliente
    - Gera AIInsight com sugestões de otimização

5. Aprovações e Workflows

Declaração Fiscal (exemplo de workflow)

1. **O contabilista:** Prepara uma declaração fiscal para um cliente
2. **O sistema:**
    - Consulta o WorkflowDefinition associado a "Declaração Fiscal"
    - Identifica o WorkflowStep atual e o próximo baseado em WorkflowStep.order
    - Atualiza o objeto Task para o novo estado
    - Cria uma Notification para o utilizador identificado em WorkflowStep.assign_to
3. **O revisor:** Revê e aprova a declaração

4. **O sistema:**
    - Atualiza Task.approved_by e Task.approval_date
    - Avança para o próximo WorkflowStep
    - Gera novas Notification para os envolvidos
5. **O contabilista:** Envia ao cliente através do sistema
6. **O cliente:** Recebe através de canal configurado em IntegrationChannel

6. Insights e Sugestões Inteligentes

Otimização de Produtividade

1. **O sistema analisa padrões e deteta:**
    - Discrepâncias entre objetos TimeEntry de diferentes utilizadores para tarefas similares
    - Utiliza histórico de TeamWorkload para identificar padrões
2. **O sistema:**
    - Cria um objeto AIInsight com insight_type="productivity_optimization"
    - Gera uma Notification para o gestor com sugestões baseadas no insight

Previsão de Sobrecarga

1. **O sistema deteta:**
    - Múltiplos objetos Task com prazos próximos para um mesmo utilizador
    - Calcula TeamWorkload.overload_risk baseado no histórico
2. **O sistema:**
    - Cria um objeto AIInsight com insight_type="workload_warning"
    - Sugere redistribuição baseado em outros objetos TeamWorkload

7. Relatórios Automáticos

Para Gestão Interna

1. **No final do mês:**
    - O sistema consulta SystemSettings para verificar configurações de relatórios
    - Cria um objeto Report com report_type="monthly_profitability"
    - Executa Report.generate() para processar dados de ClientProfitability
    - Armazena o resultado em Report.file_path

- Utiliza Report.send() para enviar aos destinatários configurados

Para Clientes

1. **Quando configurado:**
    - O sistema aciona criação de Report com report_type="client_activity"
    - Consulta todos os objetos TimeEntry, Task e Document do cliente
    - Usa IntegrationChannel para enviar o relatório ao cliente

8. Gestão de Tempo em Equipa

Distribuição de Carga

1. **O gestor:** Verifica o dashboard de carga de trabalho da equipa
2. **O sistema:**
    - Consulta objetos TeamWorkload de todos os utilizadores
    - Analisa objetos Task pendentes e a sua distribuição
    - Gera objetos AIInsight com sugestões de redistribuição
3. **O gestor:** Redistribui uma tarefa
4. **O sistema:**
    - Atualiza Task.assigned_to
    - Recalcula TeamWorkload para os utilizadores afetados
    - Cria objetos Notification para informar as mudanças
    - Regista a mudança em ActivityLog

Cobertura de Ausências

1. **Um contabilista:** Regista ausência no sistema
2. **O sistema:**
    - Atualiza o perfil do utilizador
    - Identifica todos os objetos Task afetados
    - Gera AIInsight com sugestão de redistribuição
    - Usa Notification para alertar gestores
3. **O gestor:** Aprova as sugestões
4. **O sistema:**
    - Atualiza os objetos Task.assigned_to conforme aprovado
    - Regista todas as alterações em ActivityLog