A arquitetura de um **Sistema de Controle de Inadimplência** deve ser planejada para gerenciar e automatizar os processos relacionados ao controle de pagamentos, identificação de inadimplentes, geração de relatórios e análise de dados. A estrutura deve garantir que o sistema seja escalável, seguro e eficiente. A seguir estão os detalhes para a arquitetura de tal sistema, considerando que ele será desenvolvido com Python e PyQt6 para a interface desktop, e usará Excel (.XLS) para a importação de dados financeiros.

**1. Camadas da Arquitetura**

1. **Camada de Apresentação (Frontend)**:
   * **PyQt6 (Interface Gráfica)**: A interface deve ser simples, intuitiva e responsiva. O usuário deve ser capaz de visualizar dados financeiros, acessar alertas de inadimplência, gerar relatórios e importar arquivos Excel. A UI pode incluir:
     + **Dashboard** com gráficos e resumos de inadimplência.
     + **Formulários** para entrada de dados (como cadastro de clientes, pagamentos realizados).
     + **Notificações** para alertas de inadimplência.
     + **Opção de Exportação** para gerar relatórios em Excel ou PDF.
     + **Filtros** para visualização detalhada dos dados (por exemplo, por data, valor, status do pagamento).
2. **Camada de Lógica de Negócio (Backend)**:
   * **Python (Lógica do Sistema)**:
     + **Processamento de Arquivos Excel**: Usar bibliotecas como pandas ou openpyxl para importar, manipular e processar arquivos .XLS.
     + **Análise de Inadimplência**: Implementação das regras de negócios para identificar clientes inadimplentes, considerando:
       - Data de vencimento.
       - Status de pagamento.
       - Cálculo de juros e multas.
       - Definição de alertas para vencimentos maiores que 30 dias, 60 dias, etc.
     + **Cálculos de Pendências**: Comparação dos valores pagos e devidos, gerando relatórios de inadimplência detalhados.
     + **Armazenamento de Dados**: O sistema pode usar uma base de dados local (SQLite ou CSV) para armazenar informações históricas de inadimplência.
     + **Alertas e Notificações**: Função para enviar notificações quando um cliente atingir um status de inadimplência específico.
     + **Relatórios Dinâmicos**: Geração de relatórios detalhados em Excel ou PDF.
3. **Camada de Persistência (Armazenamento de Dados)**:
   * **Banco de Dados Local (SQLite ou CSV)**:
     + **Estrutura do Banco**:
       - Tabelas principais: Clientes, Pagamentos, Inadimplência, Parcelas.
       - Relacionamentos entre as tabelas para garantir integridade dos dados.
     + **Armazenamento Incremental**: Ao importar novos arquivos Excel, as informações são adicionadas ao banco sem sobrescrever os dados anteriores.
     + **Backup e Recuperação de Dados**: Funcionalidade para garantir que os dados armazenados possam ser recuperados em caso de falhas.
4. **Camada de Integração e Processamento**:
   * **Leitura e Processamento de Arquivos Excel**: A lógica para processar arquivos Excel pode ser dividida em funções específicas, como:
     + **Importação de Dados**: Extrair informações de créditos, saldos, dívidas, pagamentos e atrasos dos arquivos Excel.
     + **Validação de Dados**: Verificar se os dados importados estão corretos (por exemplo, datas, valores, formatos).
   * **Cálculo de Inadimplência**: Após a importação dos dados, o sistema realiza o cálculo da inadimplência com base nas datas de vencimento e pagamentos realizados.
5. **Camada de Segurança**:
   * **Autenticação de Usuários**: Sistema de login para diferentes níveis de acesso (administrador, usuários com permissão limitada).
   * **Criptografia de Dados Sensíveis**: Garantir que informações sensíveis, como dados financeiros, sejam criptografadas durante o armazenamento e transmissão.
   * **Controle de Acesso**: Permitir que apenas usuários autorizados acessem e editem informações importantes.
6. **Camada de Relatórios e Visualização de Dados**:
   * **Visualização Gráfica**: O uso de bibliotecas como matplotlib ou PyQtGraph pode ser utilizado para apresentar gráficos de inadimplência, evolução de pagamentos, etc.
   * **Relatórios Exportáveis**: A funcionalidade de gerar relatórios em Excel ou PDF para facilitar o compartilhamento e análise externa.
   * **Filtros e Pesquisa**: Permitir que o usuário filtre dados e personalize os relatórios com base em critérios como data, valor e status de pagamento.

**2. Fluxo do Sistema**

1. **Importação de Dados**:
   * O usuário importa um arquivo Excel contendo dados financeiros.
   * O sistema valida e organiza os dados, realizando o tratamento necessário (ex: remoção de dados inválidos).
2. **Análise de Inadimplência**:
   * O sistema verifica os pagamentos realizados em comparação com os valores devidos e as datas de vencimento.
   * Gera alertas e marca clientes como inadimplentes conforme a regra de negócios (ex: atraso superior a 30 dias).
3. **Exibição de Resultados**:
   * O Dashboard mostra uma visão geral da inadimplência, com gráficos e resumos.
   * Os relatórios detalhados são gerados conforme a necessidade do usuário, com informações sobre o status de cada cliente e valor pendente.
4. **Geração de Relatórios**:
   * O sistema gera relatórios financeiros e de inadimplência que podem ser exportados para Excel ou PDF.
5. **Notificação e Ações**:
   * O sistema pode enviar notificações sobre inadimplência para os administradores ou clientes, caso haja essa funcionalidade configurada.

**3. Tecnologias e Ferramentas**

* **Frontend (UI)**: PyQt6 para a construção da interface desktop.
* **Backend (Lógica de Processamento)**: Python com bibliotecas como pandas para manipulação de dados e openpyxl para lidar com arquivos Excel.
* **Banco de Dados**: SQLite para armazenamento local ou CSV para simplicidade.
* **Visualização de Dados**: matplotlib ou PyQtGraph para gráficos.
* **Relatórios**: Bibliotecas como openpyxl para geração de relatórios Excel ou ReportLab para PDFs.

**4. Considerações Finais**

* **Escalabilidade**: A arquitetura pode ser expandida para incluir novas funcionalidades, como a integração com sistemas de pagamento online ou o envio de e-mails automáticos para inadimplentes.
* **Usabilidade**: A interface deve ser projetada para ser simples e intuitiva, com foco na facilidade de navegação e no acesso rápido às informações mais relevantes.
* **Manutenibilidade**: O código e a estrutura do banco de dados devem ser organizados de forma a permitir manutenções e atualizações futuras sem causar problemas na operação diária do sistema.

Essa arquitetura proporciona um sistema robusto, flexível e eficiente, com todas as funcionalidades necessárias para o controle de inadimplência de forma automatizada e otimizada.