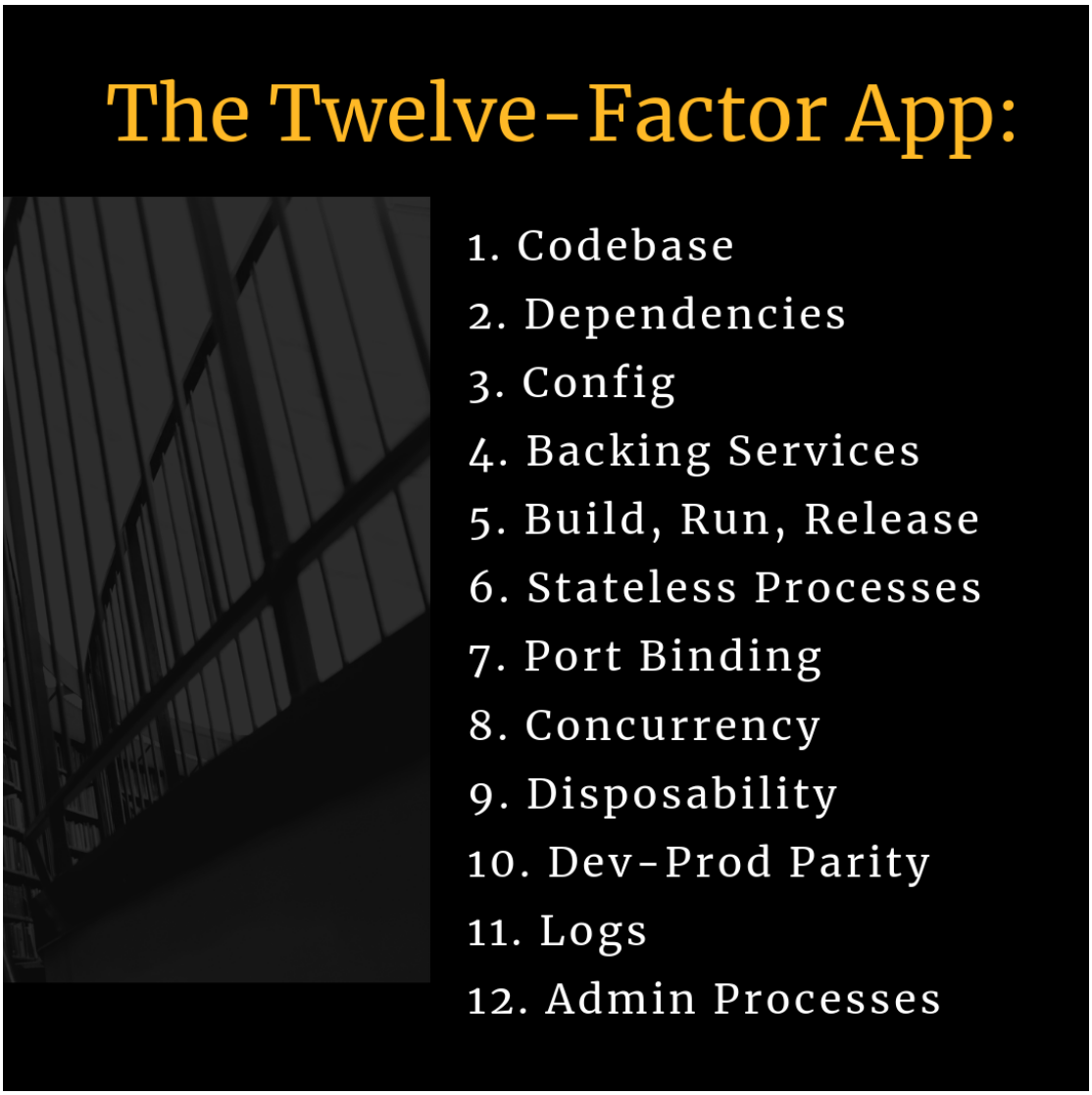
PROGRAMAÇÃO WEB
Curso de Engenharia de Software - UniEVANGÉLICA
Disciplina de Peogramação WEB
Dev: João Vitor de Pina Menezes - 2110972
DATA: 29/06/2023

ARP - 3

Introdução:

Nesta apresentação, exploraremos a aplicação dos 11 fatores da metodologia no desenvolvimento do Sistema de Cadastro e Controle de Pacientes da Rede Pública de Saúde proposto pela Software House UniSoft. Esses fatores são princípios importantes para a construção de aplicações modernas, escaláveis e de fácil manutenção. Vamos discutir cada um deles e apresentar exemplos de como eles se aplicam ao sistema em questão.



1. Base de Código

A base de código é um fator crucial para a manutenibilidade e colaboração eficaz no desenvolvimento de software. Para o Sistema de Cadastro e Controle de Pacientes, adotaremos práticas como controle de versão com o uso de um sistema de gerenciamento de código, como o Git, para garantir o controle de mudanças e rastreabilidade. Utilizaremos estratégias de organização do código-fonte, como modularização e componentização, para facilitar a compreensão e manutenção do sistema.

2. Dependências

Gerenciar as dependências de um projeto é essencial para garantir a

reprodutibilidade do ambiente de desenvolvimento e implantação. Utilizaremos gerenciadores de pacotes, como o npm (Node Package Manager) ou o Maven, para controlar as dependências do sistema. Serão definidos arquivos de manifesto (por exemplo, package.json) para listar todas as dependências necessárias, bem como suas versões compatíveis. Dessa forma, garantiremos que todas as bibliotecas externas sejam controladas e facilmente instaladas em qualquer ambiente.

3. Configurações

As configurações do sistema são frequentemente dinâmicas e podem variar entre ambientes, como desenvolvimento, teste e produção. Para o Sistema de Cadastro e Controle de Pacientes, as configurações serão armazenadas externamente, de preferência em arquivos de configuração separados, como arquivos de propriedades. Isso permitirá que as configurações sejam facilmente modificadas sem a necessidade de recompilação do código-fonte, garantindo a adaptabilidade do sistema em diferentes ambientes.

4. Backing Services

Backing services referem-se a serviços externos que o sistema depende, como bancos de dados, serviços de autenticação ou sistemas de armazenamento em nuvem. Para o Sistema de Cadastro e Controle de Pacientes, iremos utilizar um banco de dados relacional, como o PostgreSQL, para persistir os dados dos pacientes. Os serviços de autenticação e armazenamento em nuvem serão fornecidos por provedores confiáveis, como o Firebase ou AWS, garantindo a escalabilidade e segurança adequadas.

5. Build, Release, Run

A divisão clara entre as etapas de build, release e run é importante para um processo de implantação eficiente e automatizado. Utilizaremos ferramentas de automação de CI/CD (Integração Contínua/Implantação Contínua), como o Jenkins ou GitLab CI/CD, para facilitar a compilação, empacotamento e implantação do sistema. Isso garantirá uma entrega contínua e confiável, além de permitir a rápida correção de bugs ou atualização de recursos.

6. Processos

Para a gestão eficiente do desenvolvimento do Sistema de Cadastro e Controle de Pacientes, adotaremos uma abordagem ágil, como o Scrum. Definiremos sprints com tarefas claras, estimativas de tempo e responsáveis. Faremos

reuniões diárias de acompanhamento (daily stand-ups) e utilizaremos ferramentas de gerenciamento de projetos, como o Jira ou Trello, para rastrear o progresso, facilitar a comunicação e identificar possíveis gargalos.

7. Portabilidade

A portabilidade é importante para garantir que o sistema possa ser implantado em diferentes ambientes de forma consistente. O Sistema de Cadastro e Controle de Pacientes será desenvolvido utilizando tecnologias multiplataforma, como frameworks web responsivos (por exemplo, React.js) para a interface móvel e APIs RESTful para comunicação entre os dispositivos móveis e o sistema online. Isso permitirá que o sistema seja executado em diferentes dispositivos e sistemas operacionais, proporcionando flexibilidade aos usuários.

8. Concorrência

O Sistema de Cadastro e Controle de Pacientes precisa lidar com possíveis cenários de concorrência, como o acesso simultâneo aos dados por vários usuários. Utilizaremos mecanismos de bloqueio e transações adequados no banco de dados para garantir a consistência e integridade dos dados. Também empregaremos técnicas de escalabilidade horizontal para lidar com o aumento da carga de acesso e distribuir o processamento entre diferentes servidores.

9. Descartabilidade

A descartabilidade refere-se à capacidade de iniciar, parar e escalar instâncias do sistema de forma rápida e segura. Utilizaremos estratégias de implantação em contêineres, como o Docker, para facilitar a criação e destruição de ambientes consistentes. Dessa forma, garantiremos a rápida recuperação em caso de falhas ou atualizações do sistema, minimizando o tempo de inatividade e mantendo a disponibilidade para os usuários.

10.Logs

Os logs são fundamentais para monitorar o sistema, solucionar problemas e obter informações valiosas sobre seu desempenho. Utilizaremos ferramentas de registro e monitoramento, como o ELK Stack (Elasticsearch, Logstash e Kibana) ou o Splunk, para coletar, analisar e visualizar os logs do Sistema de Cadastro e Controle de Pacientes. Isso nos permitirá identificar possíveis erros, gargalos de desempenho e melhorar a eficiência do sistema.

11.Processos de Administração/Conciliação

Processos de administração e conciliação são necessários para garantir a integridade e consistência dos dados. Utilizaremos estratégias de backup e recuperação para proteger os dados do Sistema de Cadastro e Controle de Pacientes contra falhas ou desastres. Também implementaremos rotinas de verificação e validação dos dados, utilizando técnicas como checksums ou assinaturas digitais, para detectar possíveis inconsistências ou adulterações.

Conclusão

Nesta apresentação, discutimos a aplicação dos 12 fatores da metodologia no desenvolvimento do Sistema de Cadastro e Controle de Pacientes da Rede Pública de Saúde proposto pela Software House UniSoft. Exploramos cada um dos fatores e apresentamos exemplos concretos de como eles podem ser aplicados no sistema, considerando suas características mínimas.

A aplicação desses fatores garantirá que o sistema seja escalável, de fácil manutenção, adaptável a diferentes ambientes, e seguirá as melhores práticas de desenvolvimento de software. Esperamos que essa solução contribua para melhorar a eficiência e qualidade dos serviços de saúde pública, beneficiando pacientes e profissionais da área.

Referências

Heroku. (s.d.). The Twelve-Factor App. Retirado de https://12factor.net/

Fowler, M. (2012). Continuous Integration. Retirado de https://martinfowler.com/articles/continuousIntegration.html

Richards, M. (2014). Microservices: Lightweight Integration for Modern Architectures. O'Reilly Media.

Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns:

Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.
Hunt, A., & Thomas, D. (2017). The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley Professional.
Larman, C., & Basili, V. R. (2003). Iterative and Incremental Development: A Brief History. IEEE Computer Society.
Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
Sommerville, I. (2016). Software Engineering. Pearson Education.
Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.
Newman, S. (2017). Building Secure and Reliable Systems: Best Practices for Designing, Implementing, and Maintaining Systems. O'Reilly Media.
Cockburn, A. (2002). Agile Software Development: The Cooperative Game. Addison-Wesley Professional.