Arquiteturas de Aplicações Mobile

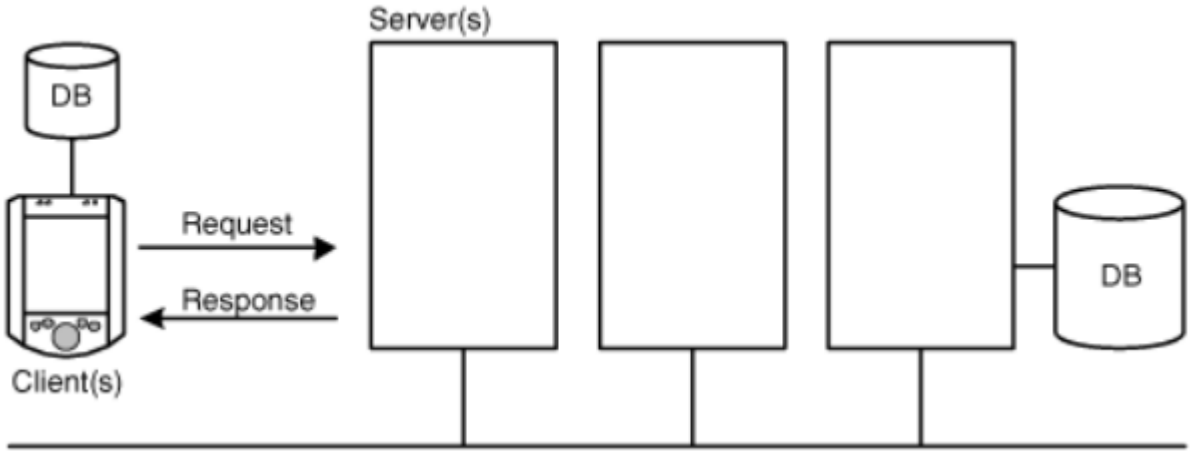
Sandro Andrade

INF628/ESPA04II – Desenvolvimento de Sistemas Mobile

sandroandrade@ifba.edu.br

Arquiteturas de Aplicações Mobile

- Arquiteturas cliente-servidor



Arquiteturas de Aplicações Mobile

- Arquiteturas em camadas
  - Melhora o reuso, segurança e facilita a evolução
  - Os clientes geralmente possuem de zero (thin) a três (fat) camadas
  - Os servidores geralmente possuem de uma a três camadas

Arquiteturas de Aplicações Mobile

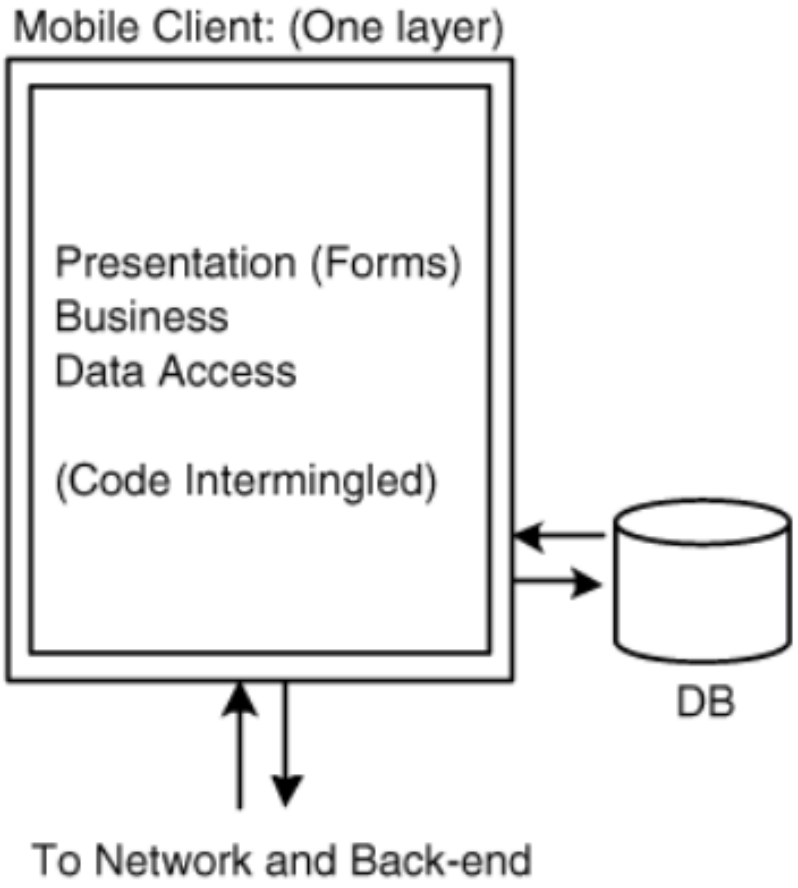
- Tipos de clientes:
  - Thin clients
    - ✔ Fáceis de manter
    - ✖ Requerem constante comunicação com o servidor

Arquiteturas de Aplicações Mobile

- Tipos de clientes:
  - Fat clients
    - ✔ Operam de forma desconectada
    - ✖ Maior dependência com o SO e tipo do dispositivo
    - ✖ Gerenciamento de releases
      - ✖ Múltiplas versões em múltiplos dispositivos

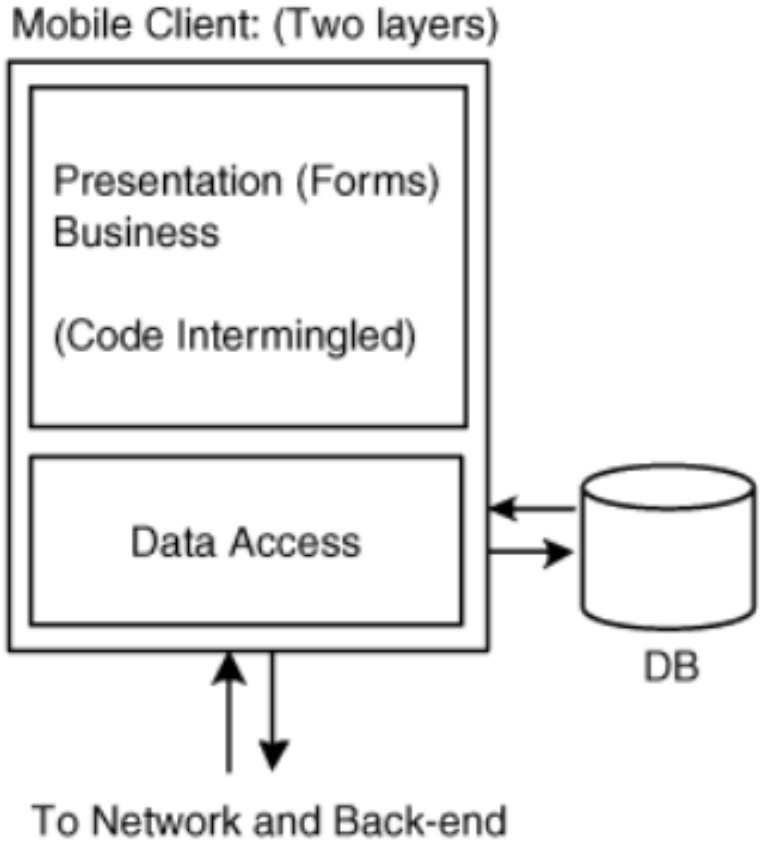
Arquiteturas de Aplicações Mobile

- Tipos de clientes:
  - Fat clients (1 camada)



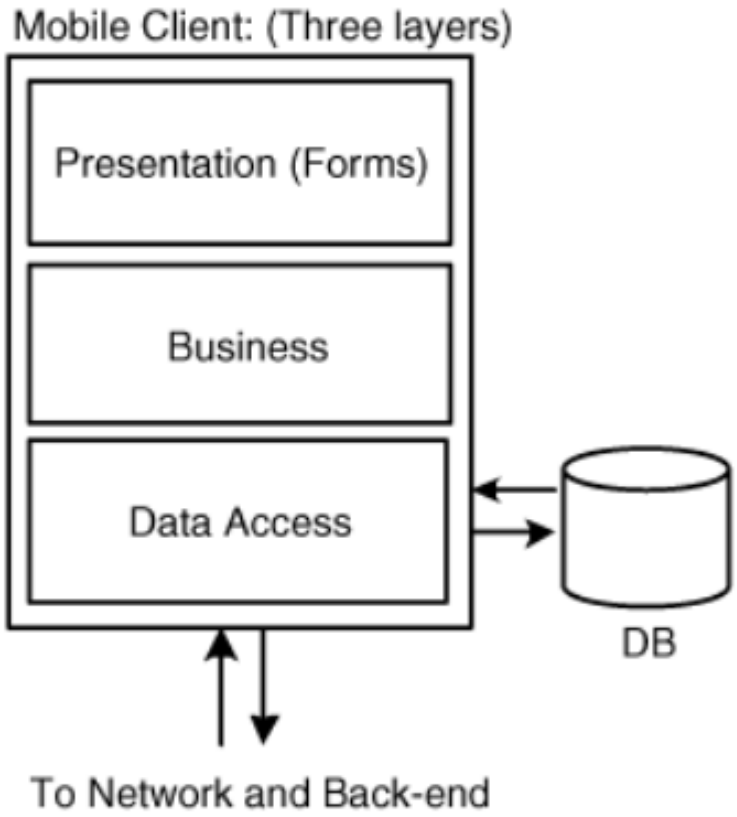
Arquiteturas de Aplicações Mobile

- Tipos de clientes:
  - Fat clients (2 camadas)



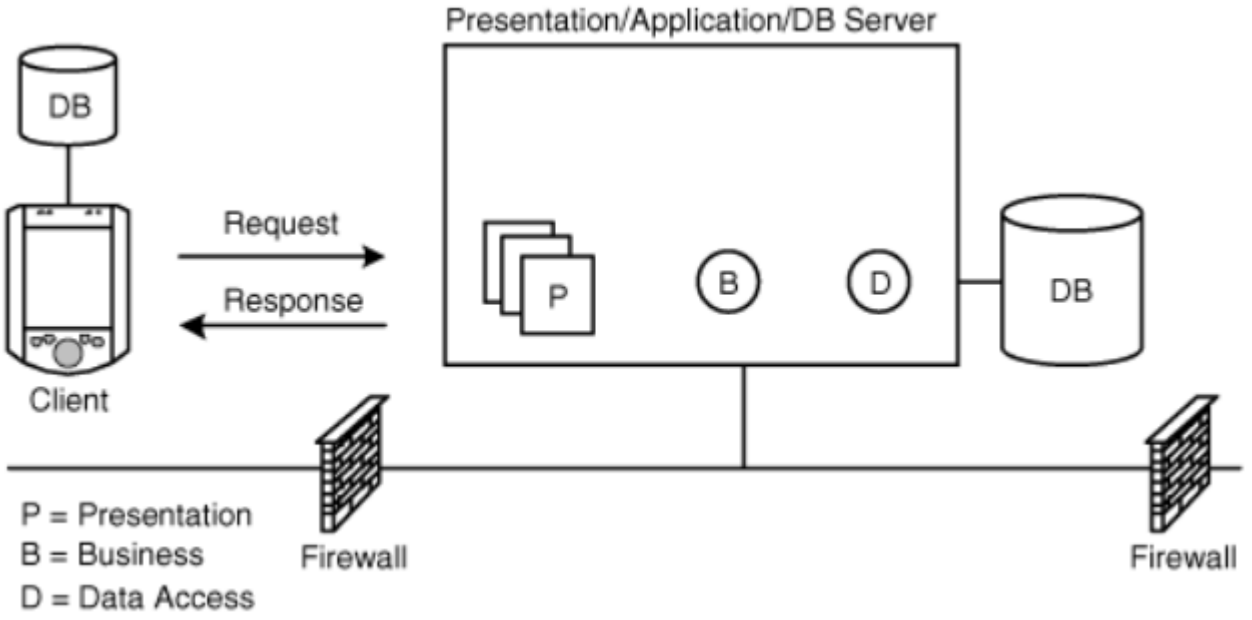
Arquiteturas de Aplicações Mobile

- Tipos de clientes:
  - Fat clients (3 camadas)



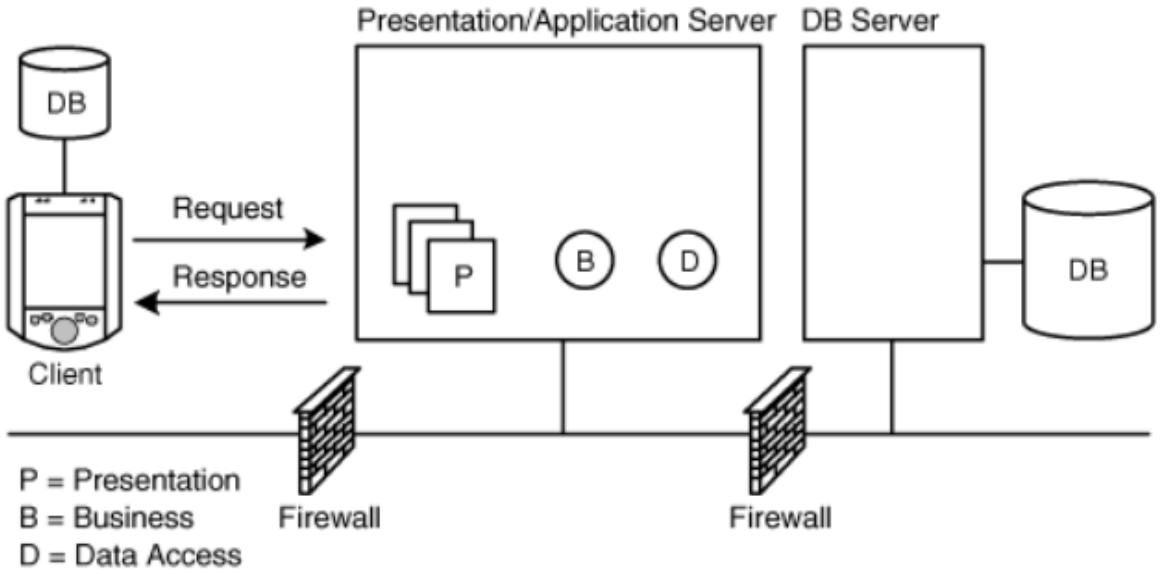
Arquiteturas de Servidores

- 1-tier architectures
  - ✔ Conveniência
  - ✔ Fácil de desenvolver e implantar
  - ✖ Pouco escalável
  - ✖ Problemas de segurança



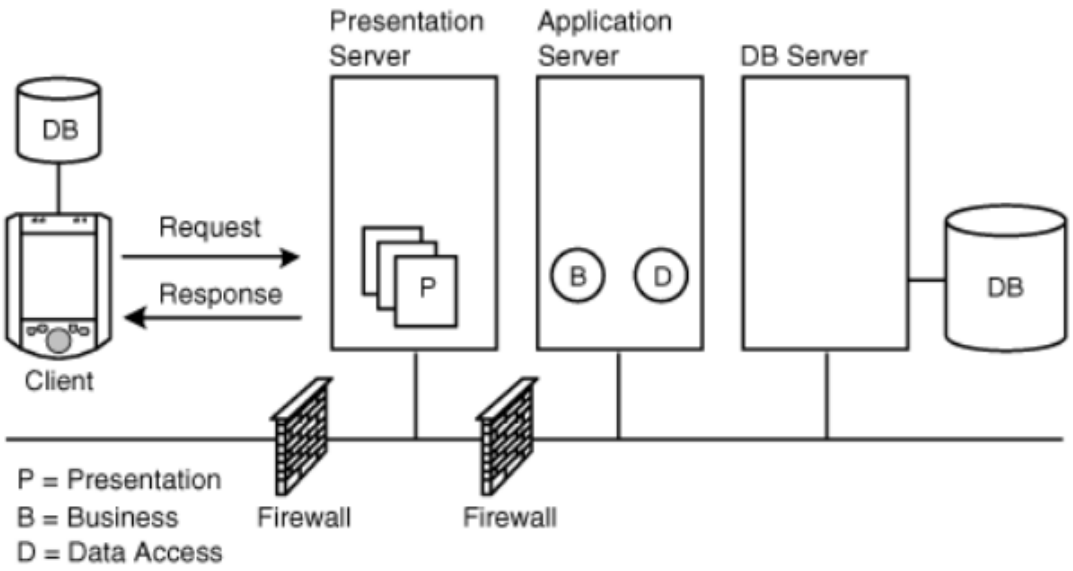
Arquiteturas de Servidores

- 2-tier architectures
  - ✔ Conveniência
  - ✔ Permite especialização do SGBD
  - ✖ Pouco escalável
  - ✖ Problemas de segurança
  - ✖ Maior custo



Arquiteturas de Servidores

- 3-tier architectures
  - ✔ Escalável
  - ✔ Seguro (firewall e zonas)
  - ✔ Permite especialização do SGBD
  - ✖ Overengineering
  - ✖ Difícil de desenvolver
  - ✖ Maior custo

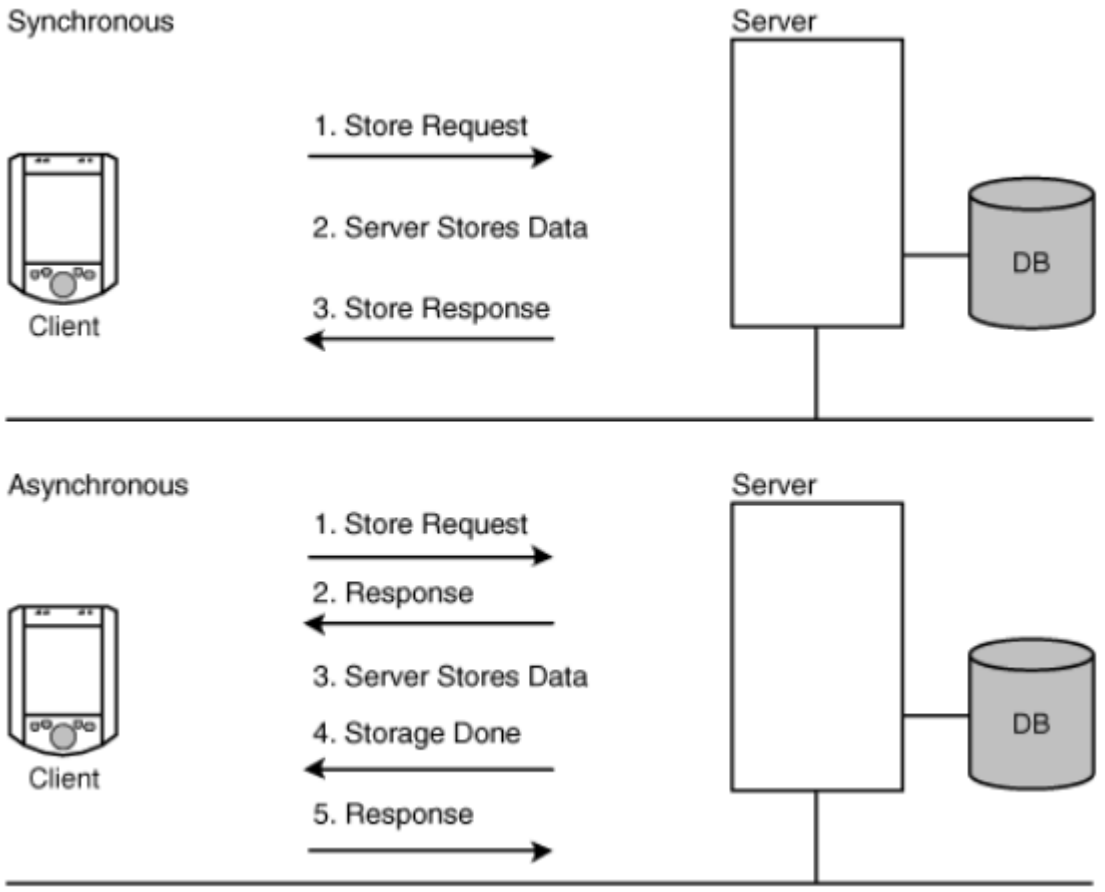


Tipos de Conexão e Sincronização

- Tipos de conexão:
  - Always connected
  - Partially connected
  - Never connected
- Tipo de sincronização:
  - Contínua
  - Store-and-Forward

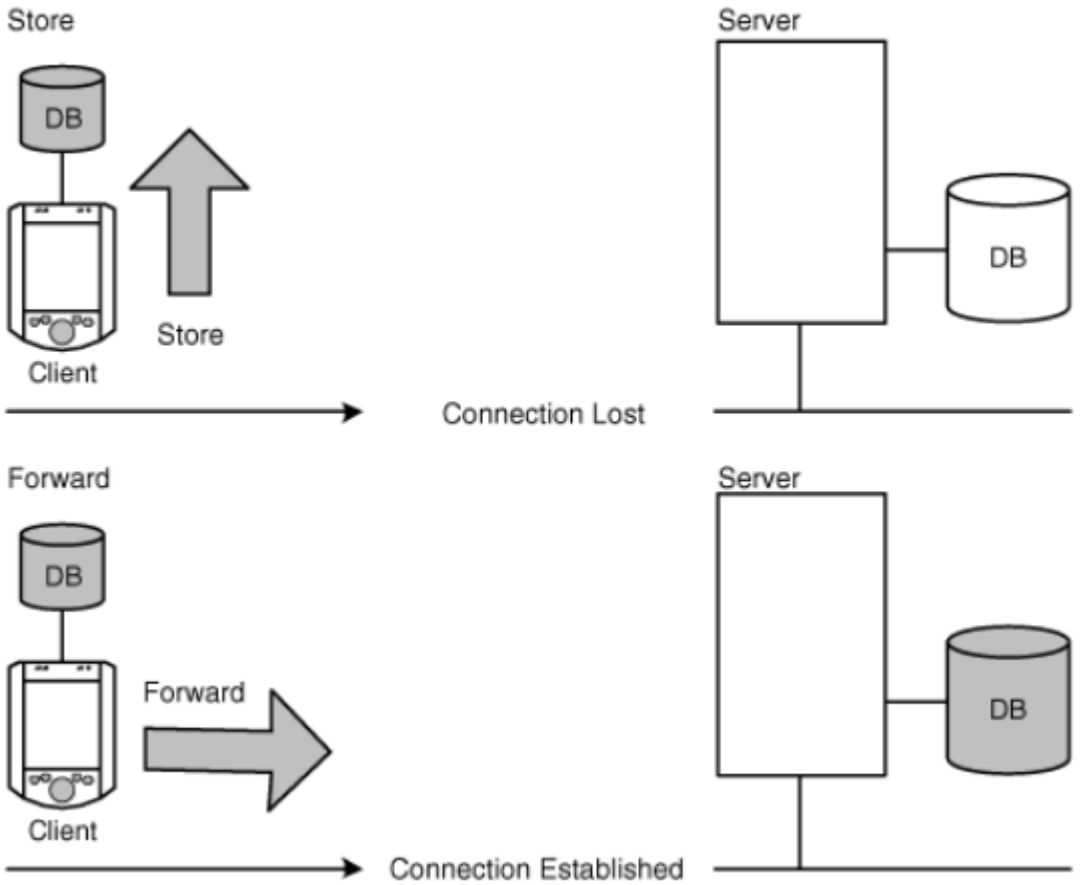
Tipos de Conexão e Sincronização

- Sincronização contínua:



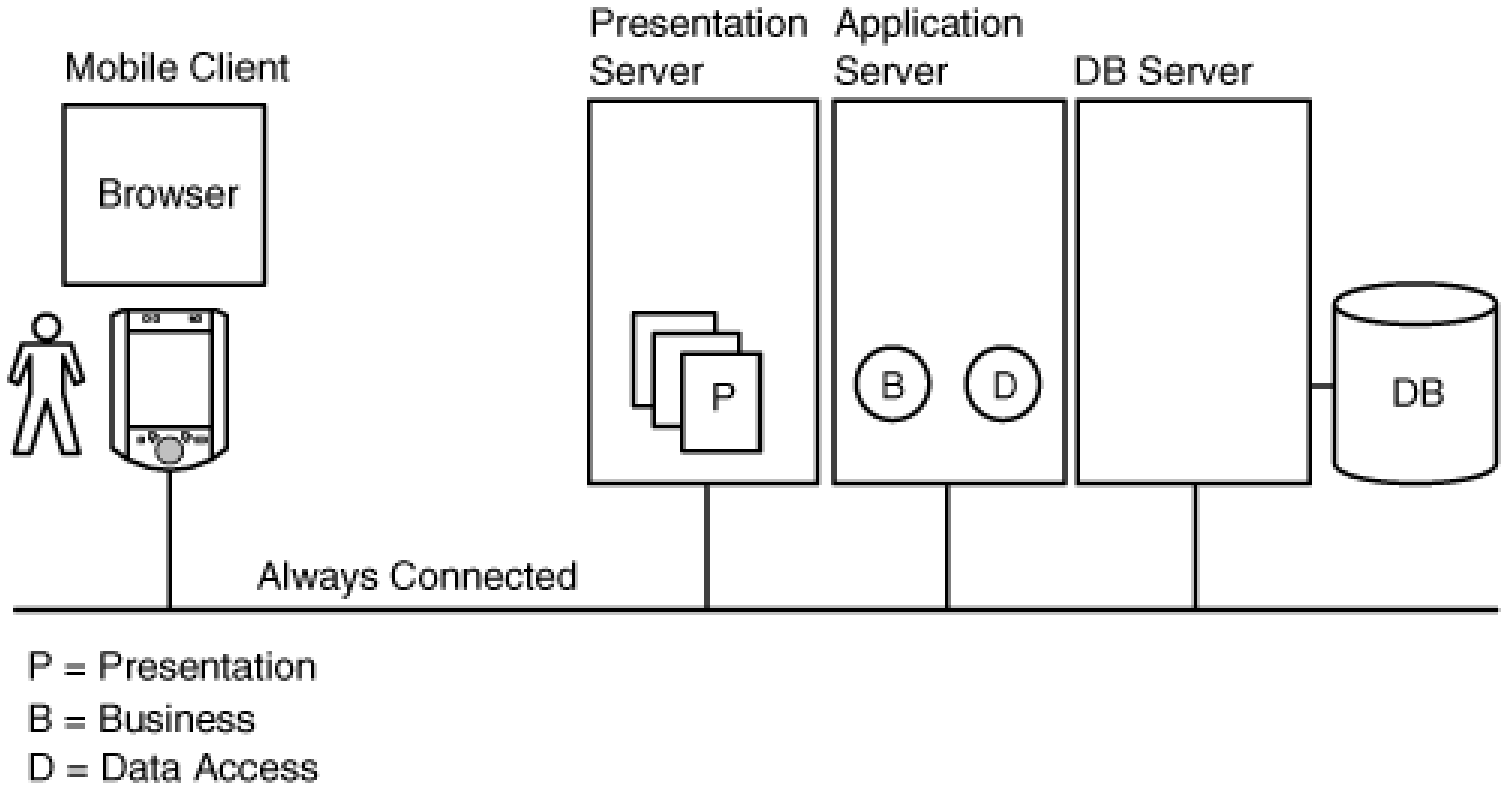
Tipos de Conexão e Sincronização

- Sincronização store-and-forward:



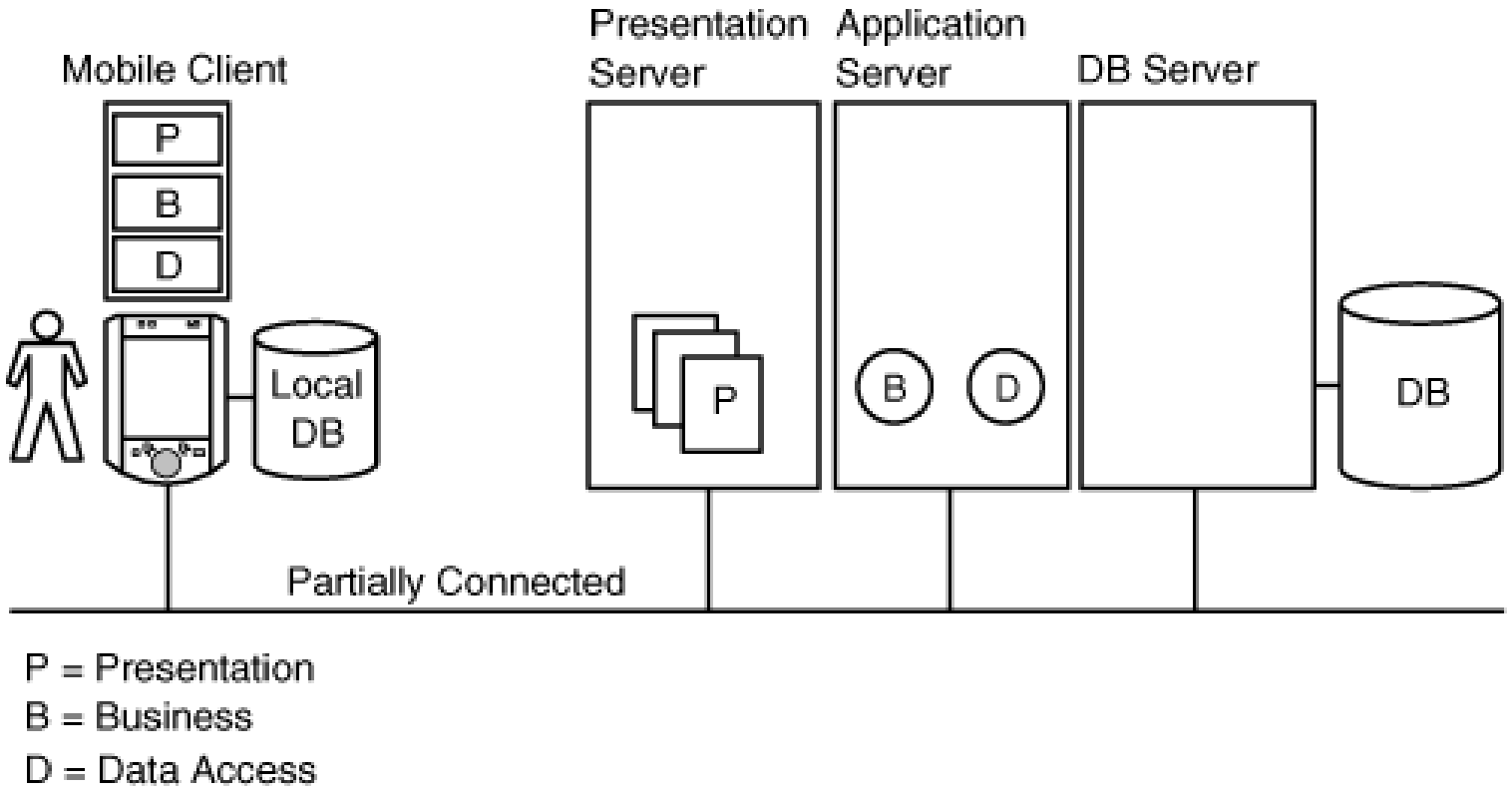
Padrões Arquiteturais Comuns

- Zero-Camadas + 3-tier + always connected

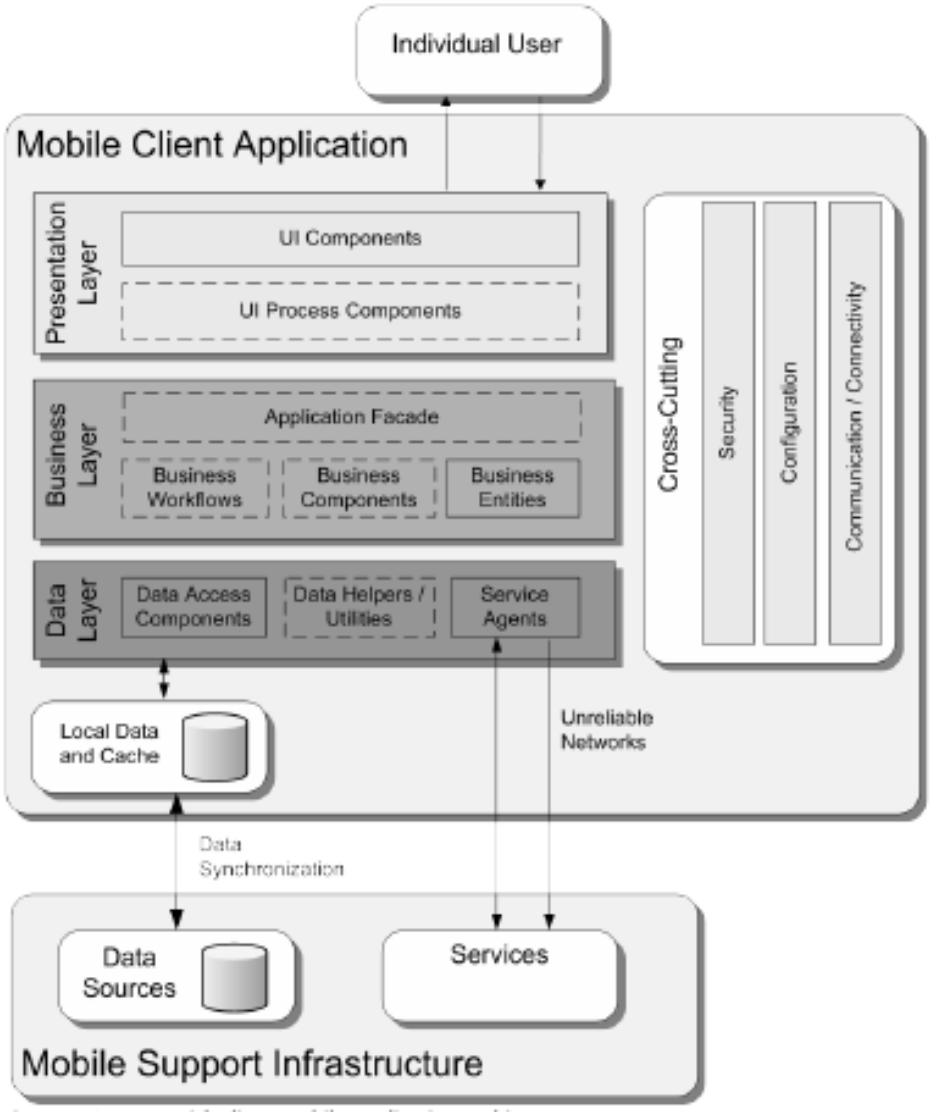


Padrões Arquiteturais Comuns

- 3-Camadas + 3-tier + partially connected



Padrões Arquiteturais Comuns



Considerações de Projeto Arquitetural

- Decida entre fat client, thin (web) client ou Rich Internet Application (RIA – nativo)
- Determine os tipos de dispositivos a suportar
- Projete considerando conectividade limitada
  - Cache, gerenciamento de estados, acesso a dados em momentos de ausência de conexão
- Projete de forma específica para mobile (first)
  - Memória, bateria, tamanho de tela, segurança e rede

Considerações de Projeto Arquitetural

- Adote uma arquitetura em camadas
  - Facilita a manutenção e o reuso
- Projeto considerando restrições de recursos
  - Bateria, memória e velocidade de processamento

Principais Problemas

Category	Key issues
Authentication and Authorization	• Failing to authenticate in occasionally connected scenarios • Failing to authorize in occasionally-connected scenarios • Failing to use authentication and authorization over a virtual private network (VPN) • Failing to authenticate during synchronization over the air • Failing to authenticate during synchronization with the host PC • Failing to authenticate for all connection scenarios, such as over the air, cradled, Bluetooth, and Secure Digital (SD) cards • Failing to appreciate the differences between security models of different devices

Principais Problemas

Caching	• Caching unnecessary data on a device that has limited resources • Relying on cached data that may no longer be available in occasionally-connected scenarios • Choosing inappropriate cache locations and formats • Caching sensitive data in unencrypted form • Failing to choose an appropriate caching technology
Communication	• Failing to protect sensitive data over the air • Failing to secure Web service communication • Failing to secure communication over a VPN • Not appreciating the performance impact of communication security on limited-bandwidth connections • Not managing limited-bandwidth connections efficiently • Not managing connections to multiple network services efficiently • Not designing to work with intermittent connectivity • Not considering connection cost or allowing the user to manage connections • Not designing to minimize power usage when running on battery power • Failing to use the appropriate communication protocol

Principais Problemas

Configuration Management	• Failing to restore configuration state after a reset • Failing to consider configuration management synchronization over the air • Failing to consider configuration management synchronization with the host PC • Choosing an inappropriate format for configuration information • Failing to protect sensitive configuration information • Failing to consider the techniques used by different manufacturers for loading configuration settings
Data Access	• Failing to implement data-access mechanisms that work with intermittent connectivity • Not considering database access performance • Navigating through large datasets when not required • Failing to consider appropriate replication technologies and techniques • Failing to consider access to device database services such as Microsoft SQL Server® Compact Edition

Principais Problemas

Device	• Failing to consider device heterogeneity, such as screen size and CPU power • Not presenting user-friendly error messages to the user • Failing to protect sensitive information • Failure to consider the processing power of the device
Exception Management	• Not recovering application state after an exception • Revealing sensitive information to the end user • Not logging sufficient details about the exception • Using exceptions to control application flow
Logging	• Not considering remote logging instead of logging on the device • Not considering how to access device logs • Not considering resource constraints when logging • Failing to protect sensitive information in the log files
Porting	• Failing to rewrite the existing rich client UI to suit the device • Failing to explore the available porting tools

Principais Problemas

Synchronization	• Failing to secure synchronization when communicating • Failing to manage synchronization over the air as opposed to cradled synchronization • Failing to manage synchronization interruptions • Failing to handle synchronization conflicts • Failing to consider merge replication where appropriate
Testing	• Failing to appreciate debugging costs when choosing to support multiple device types • Failing to design with debugging in mind; for example, using emulators instead of the actual devices • Failing to debug in all connection scenarios

Principais Problemas

UI	• Not considering the restricted UI form factor • Not considering the single window environment • Not considering that only one application can be running • Not designing a touch-screen or stylus-driven UI for usability • Not including support for multiple screen sizes and orientations • Not managing device reset and resume • Not considering the limited API and reduced range of UI controls compared to the desktop
Validation	• Not validating input and data during host PC communication • Not validating input and data during over-the-air communication • Failing to protect hardware resources, such as the camera and initiation of phone calls • Not designing validation with limited resources and performance in mind

Potenciais Design Patterns

Category	Relevant patterns
Caching	• Lazy Acquisition
Communication	• Active Object • Communicator • Entity Translator • Reliable Sessions
Data Access	• Active Record • Data Transfer Object • Domain Model • Transaction Script
Synchronization	• Synchronization
UI	• Application Controller • Model-View-Controller • Model-View-Presenter • Pagination

Metas Arquiteturais

- Atendimento aos requisitos
- Independência de tecnologia
- Alto desempenho
- Alta disponibilidade
- Escalabilidade (horizontal x vertical)

Diretrizes Arquiteturais

- Separation of Concerns
- Princípio da Responsabilidade Única
- Princípio do Conhecimento Mínimo (LoD)
- DRY
- Evite grandes designs antecipados
- Composição é melhor que herança

Principais Problemas Arquiteturais

Area	Key issues
Authentication	• Lack of authorization across trust boundaries • Granular or improper authorization
Caching	• Caching volatile data not needed in an offline scenario • Caching sensitive data • Incorrect choice of caching store
Communication	• Incorrect choice of transport protocol • Chatty communication across physical and process boundaries • Failure to protect sensitive data
Concurrency and Transactions	• Not protecting concurrent access to static data • Deadlocks caused by improper locking • Not choosing the correct data concurrency model • Long-running transactions that hold locks on data • Using exclusive locks when not required

Principais Problemas Arquiteturais

Area	Key issues
Configuration Management	• Lack of or incorrect configuration information • Not securing sensitive configuration information • Unrestricted access to configuration information
Coupling and Cohesion	• Incorrect grouping of functionality • No clear separation of concerns • Tight coupling across layers
Data Access	• Chatty calls to the database • Business logic mixed with data access code
Exception Management	• Failing to an unstable state • Revealing sensitive information to the end user • Using exceptions to control application flow • Not logging sufficient details about the exception
Layering	• Incorrect grouping of components within a layer • Not following layering and dependency rules

Principais Problemas Arquiteturais

Logging and Instrumentation	• Lack of logging and instrumentation • Logging and instrumentation that is too fine-grained • Not making logging and instrumentation an option that is configurable at run time • Not suppressing and handling logging failures • Not logging business-critical functionality
State Management	• Using an incorrect state store • Not considering serialization requirements • Not persisting state when required
User Experience	• Not following published guidelines • Not considering accessibility • Creating overloaded interfaces with unrelated functionality
Validation	• Lack of validation across trust boundaries • Failure to validate for range, type, format, and length • Not reusing validation logic
Workflow	• Not considering management requirements • Choosing an incorrect workflow pattern • Not considering exception states and how to handle them

Potenciais Architectural Patterns

Category	Relevant patterns
Caching	• Cache Dependency
Communication	• Pipes and Filters • Service Interface
Concurrency and Transactions	• Optimistic Offline Lock • Pessimistic Offline Lock
Coupling and Cohesion	• Adapter • Dependency Injection
Data Access	• Active Record • Query Object • Row Data Gateway • Table Data Gateway
Layering	• Façade • Layered Architecture

Padrões de Projeto para Aplicativos Móveis

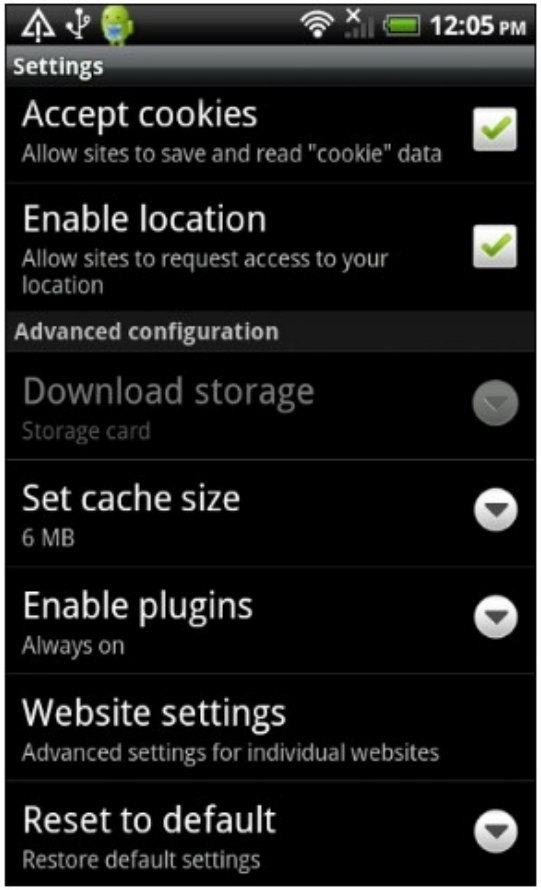
- Padrões de Interação
  - Back-and-Save, Guess-Don’t-Ask, A-la-Carte-Menu, Sink-or-Async, Logon-and-Forget,
- Padrões para Apresentação
  - Babel-Tower, Do-as-Romans-Do, List-and-Scroll
- Padrões Comportamentais
  - Predictive Fetch, Memento-Mori, As-soon-as-Possible,

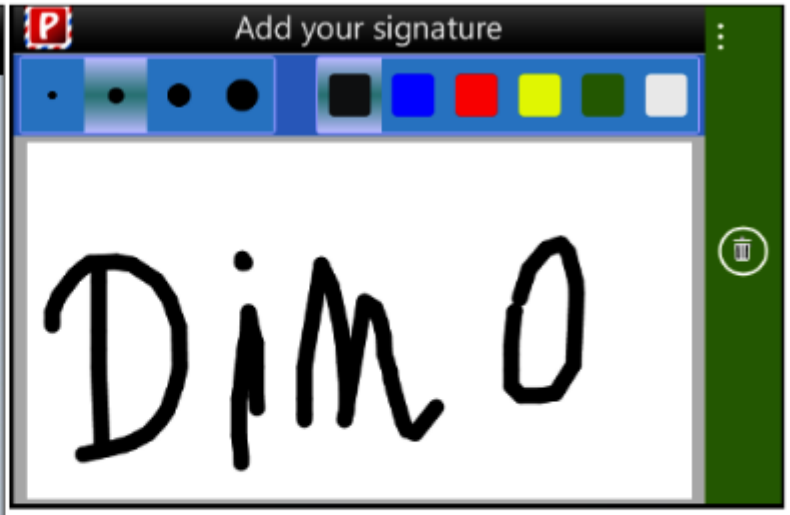
Padrões de Projeto para Aplicativos Móveis

- Back-and-Save
  - “Salve o conteúdo das telas de entrada de dados quando o usuário sair (ou for forçado a sair) desta tela, geralmente via tecla back” (Back-and-Save)
  - “Salve o conteúdo das telas de entrada de dados periodicamente” (Auto-Save)

Padrões de Projeto para Aplicativos Móveis

- Back-and-Save



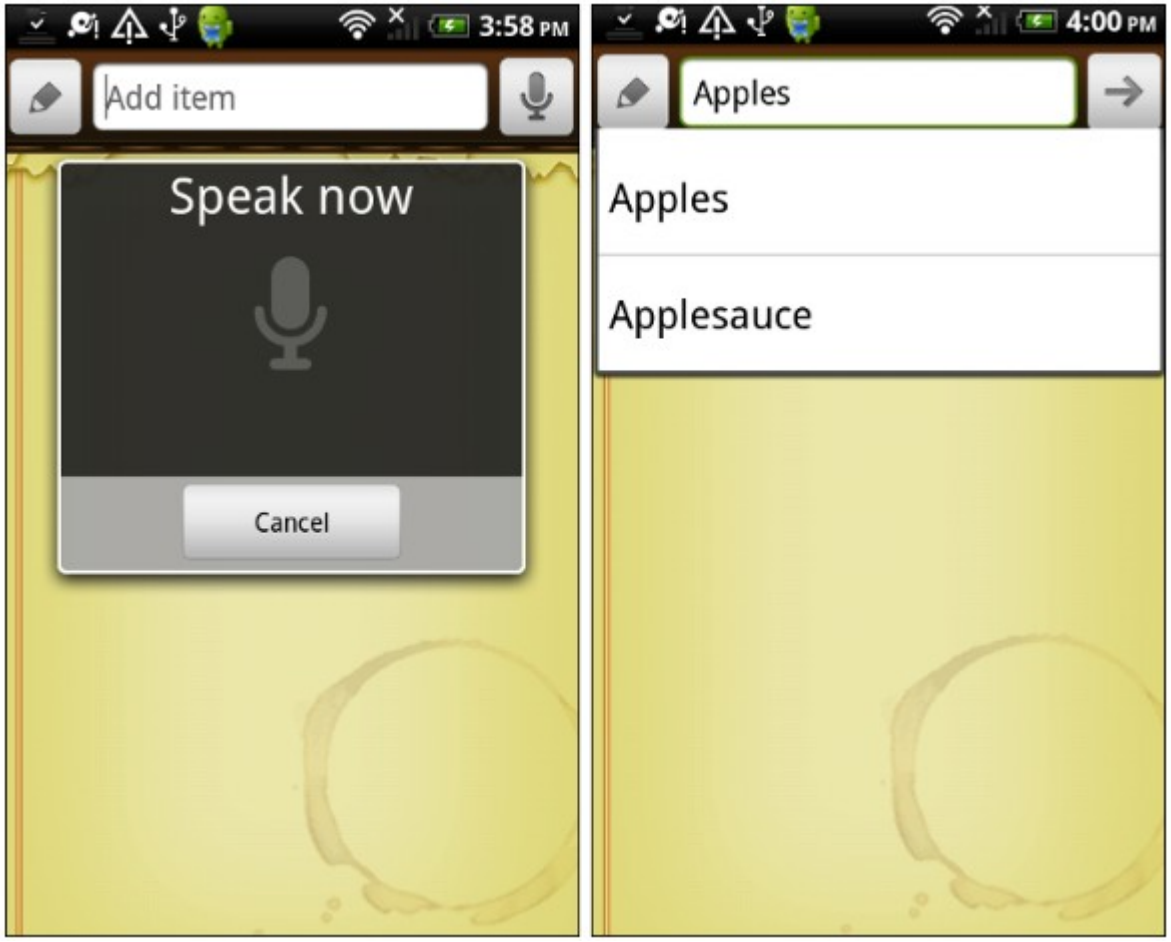


Padrões de Projeto para Aplicativos Móveis

- Guess-Don’t-Ask
  - “Use qualquer recurso disponível para tomar decisões proativar e evitar o máximo possível de interação do usuário”

Padrões de Projeto para Aplicativos Móveis

- Guess-Don't-Ask



Padrões de Projeto para Aplicativos Móveis

- A-la-Carte-Menu
  - “A qualquer momento deve ser claro para o usuário qual ação tomar e quantas opções ele tem”.

Padrões de Projeto para Aplicativos Móveis

- A-la-Carte-Menu

Padrões de Projeto para Aplicativos Móveis

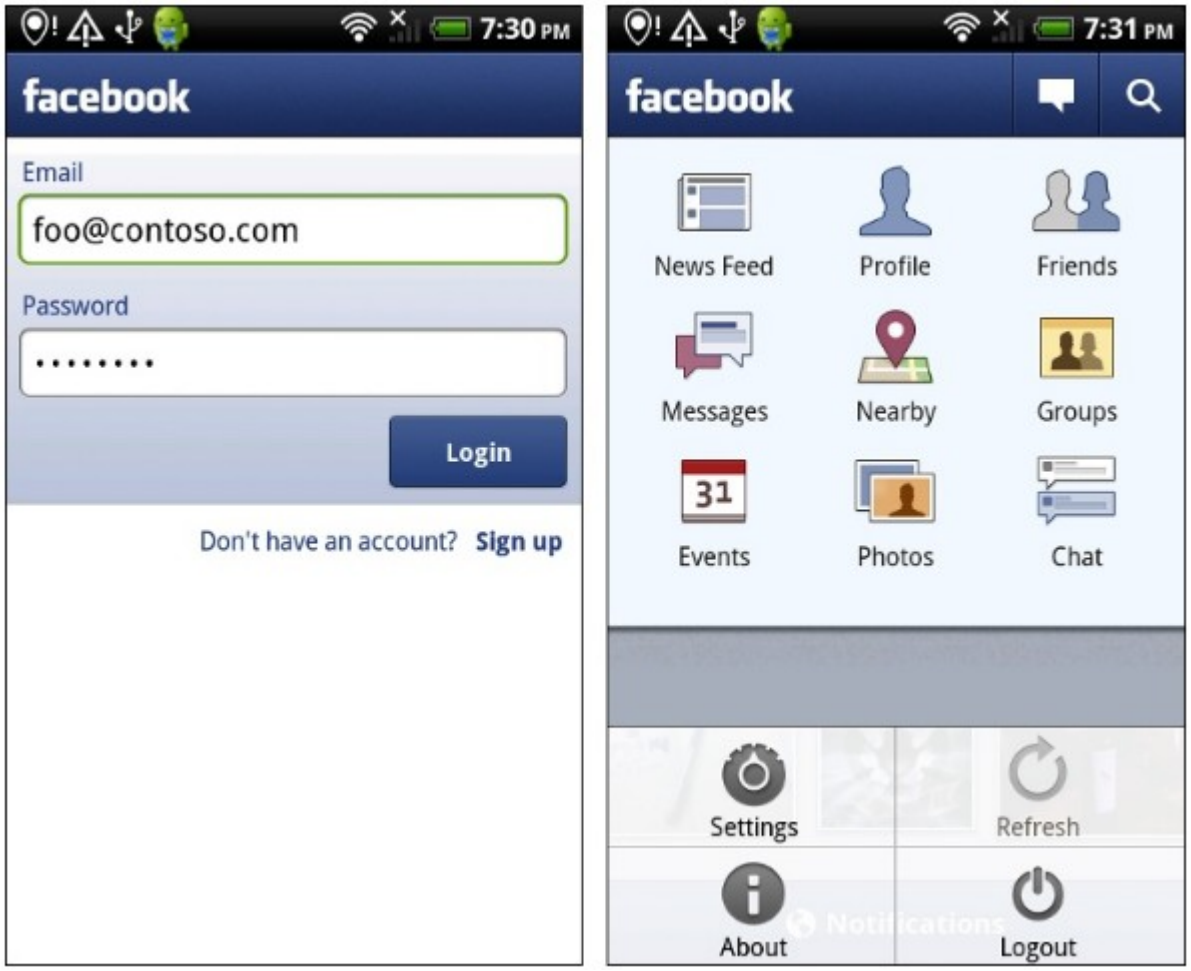
- Sink-or-Async
  - “Implemente de forma assíncrona qualquer operação que possa demorar mais do que poucos milissegundos”
  - Exemplos no Qt:
    - XmlHttpRequest
    - QFuture / Qt Concurrency Framework

Padrões de Projeto para Aplicativos Móveis

- Logon-and-Forget
  - “O aplicativo deve solicitar credenciais uma única vez, armazená-las de forma segura e autenticar o usuário de forma transparente a cada sessão”

Padrões de Projeto para Aplicativos Móveis

- Logon-and-Forget



Padrões de Projeto para Aplicativos Móveis

- Babel-Tower
  - “Evite texto hard-coded e com layout fixo. Projete sua aplicação para suportar a injeção dinâmica de texto traduzido”.
  - i18n/l10n
  - Exemplo: internacionalização e localização no Qt

Padrões de Projeto para Aplicativos Móveis

- Do-as-Romans-Do
  - NUI (Natural User Interfaces)
  - “Se você está em Roma, faça como os Romanos fazem”
  - “É convincente para os usuários e, possivelmente, vantajoso para os desenvolvedores, respeitar o look-and-feel e os recursos do sistema operacional hospedeiro”
  - Exemplo: look’n’feel nativo no Qt

Padrões de Projeto para Aplicativos Móveis

- List-and-Scroll
  - “Não tenha medo de usar listas (verticais) no seu aplicativo, mesmo listas longas, com mais de 100 elementos”
  - Exemplo: lazy load nos models do Qt

Padrões de Projeto para Aplicativos Móveis

- Predictive Fetch
  - “Se o aplicativo opera no modo Sempre Conectado, faça o download de dados prováveis de serem usados mais tarde e certifique-se que existem dados suficientes para suportar uma eventual falta de conectividade”
  - Exemplo: Facebook

Padrões de Projeto para Aplicativos Móveis

- Memento-Mori
  - “Lembre-se que você vai morrer”
  - “Os aplicativos devem sempre salvar o seu estado relevante quando o sistema operacional os colocar em background”

Padrões de Projeto para Aplicativos Móveis

- As-soon-as-Possible
  - “Operações remotas críticas à aplicação devem ser implementadas em modo protegido e confirmadas várias vezes antes de uma falha. Em caso de falha, a operação deve ser registrada e executada assim que a conectividade retornar”

Outros recursos interessantes

- https://unitid.nl/androidpatterns/
- https://issuu.com/noeemi/docs/mobile_design_pattern_gallery__2nd_
- https://www.slideshare.net/hassandar18/architecture-of-mobile-software-applications
- https://www.youtube.com/watch?v=lgLih6RQti8
- http://www.rapidvaluesolutions.com/wp-content/uploads/2013/04/How-to-Choose-the-Right-Technology-Architecture-for-Your-Mobile-Application.pdf

Arquiteturas de Aplicações Mobile

Sandro Andrade

INF628/ESPA04II – Desenvolvimento de Sistemas Mobile

sandroandrade@ifba.edu.br