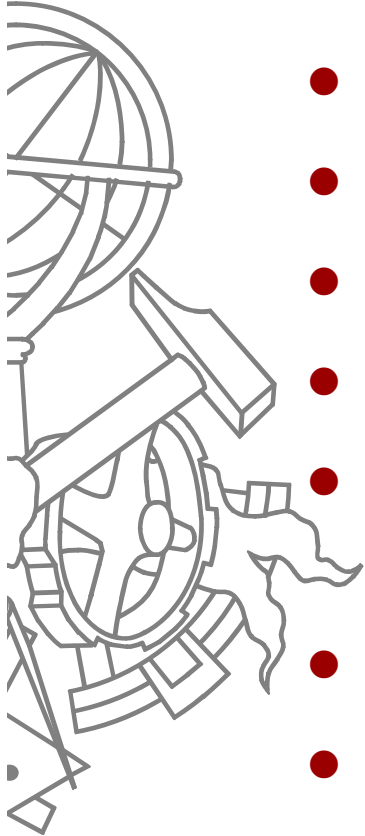




Sinopse e Discussão

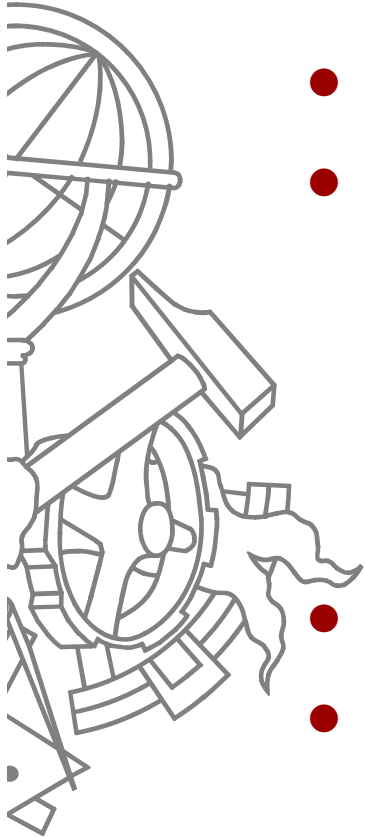
Parte 3

Questões



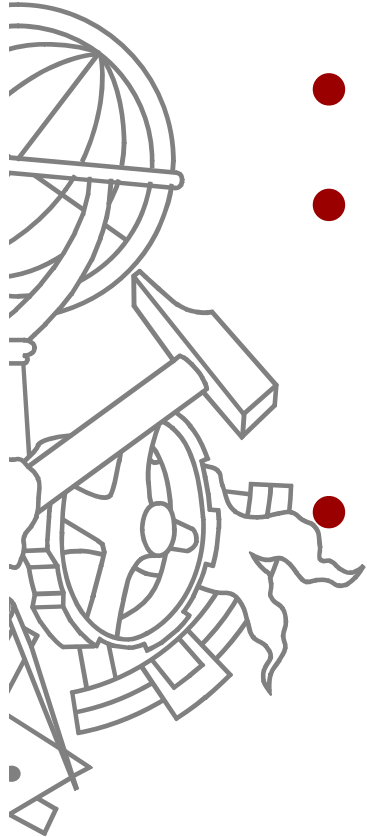
- Como dividir a aplicação?
- Como representar as entidades de negócio?
- Como representar a lógica de negócio?
- Como persistir as entidades de negócio?
- Como garantir a consistência dos dados da base de dados?
- Como tratar a interacção com o utilizador?
- Como resolver questões relacionadas com a distribuição física da aplicação?

Como dividir a aplicação?



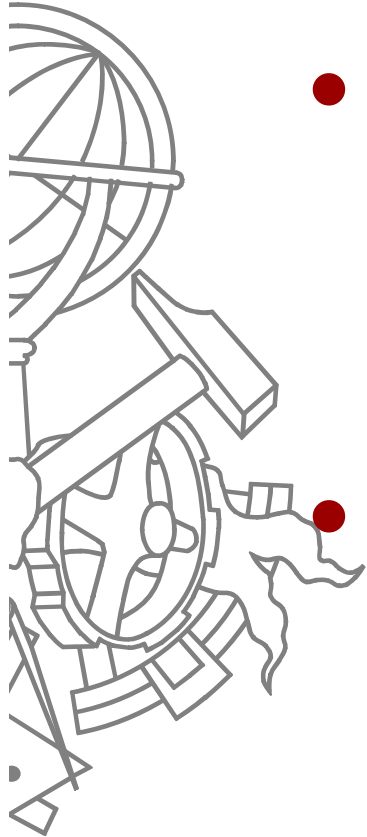
- Que camadas implementar?
- Como dividir a aplicação em projectos/componentes?
 - Que projectos?
 - Que componentes em cada projecto?
 - Que interface em cada componente?
- Manter ou não estado nos componentes?
- Que tipo de interface colocar nos componentes de DAL?
 - CRUD ou negócio?

Como representar entidades de negócio?



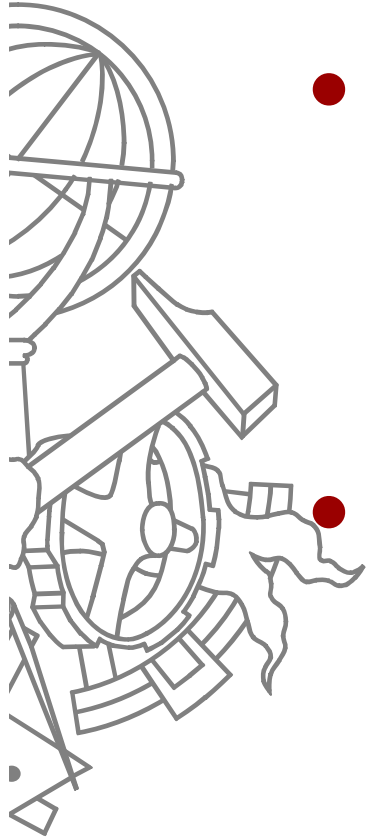
- Custom, XML, DataSet, ResultSet, ...
- Para a plataforma destino em causa há algum modelo mais comum? Ou melhor suportado?
- Como passar estas classes entre camadas (eventualmente remotas)?

Como representar a lógica de negócio?



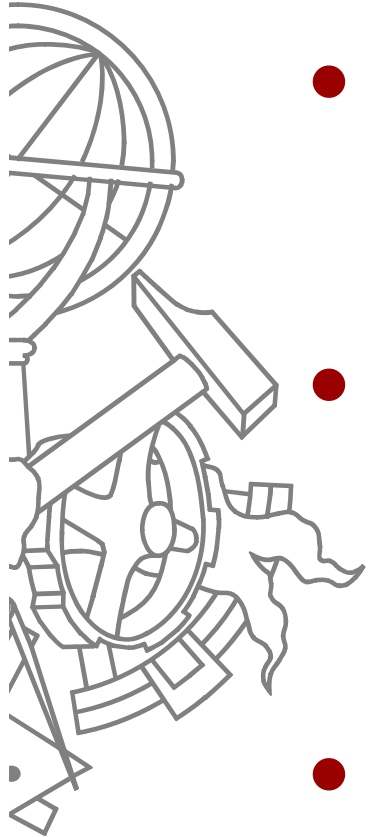
- Lógica simples a complexa e bom suporte na plataforma para *Record sets*
→ Table Module
- Lógica muito complexa
→ Domain Model

Como representar a lógica de negócio?



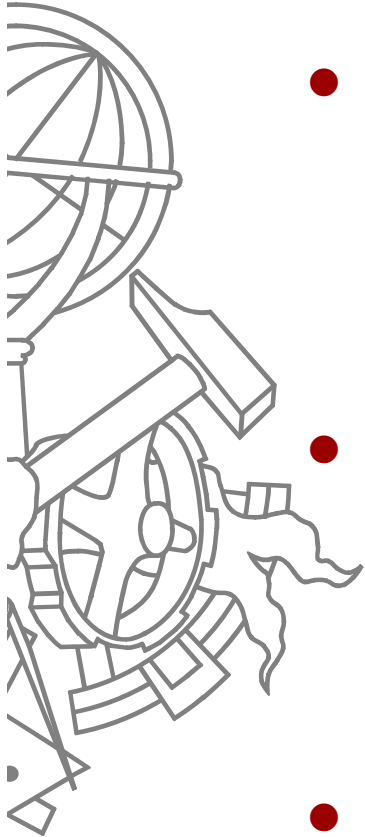
- *Table Module* é muito adequado quando a plataforma suporta *Record Sets* nas várias camadas (especialmente UI)
- *Domain Model* é mais complexo mas mais flexível (paradigma OO puro)

Como persistir as entidades de negócio?



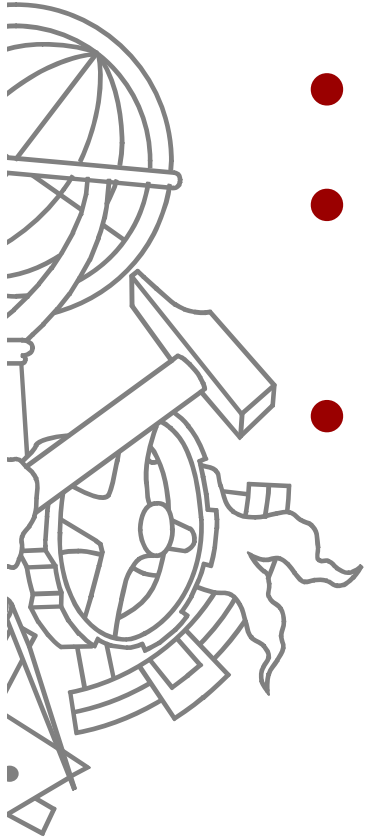
- *Table Module*
→ Table Data Gateway
- *Domain Model* muito similar à estrutura da BD
→ Active Record
- *Domain Model* complexo
→ Data Mapper

Como persistir as entidades de negócio?



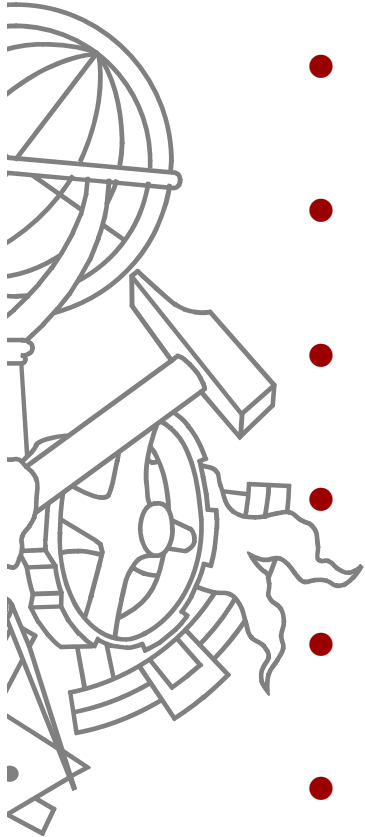
- Garantir que os dados em memória só são actualizados num sítio
→ Identity Map
- Garantir ligação entre objectos de domínio (memória) e entidades na BD
→ Identity field
- Evitar carregar para memória toda a BD
→ Lazy Load

Como persistir as entidades de negócio?



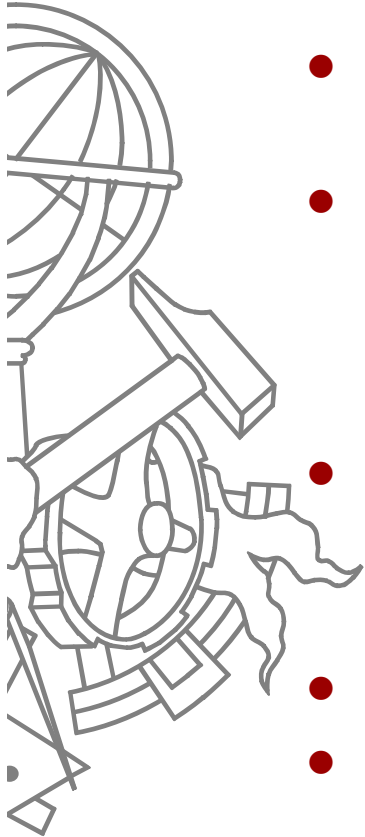
- Abstrair o formato de dados da BD?
- Desenvolver classes próprias ou usar modelo relacional?
- Como mapear o formato relacional na BD para as classes da linguagem?

Como persistir as entidades de negócio?



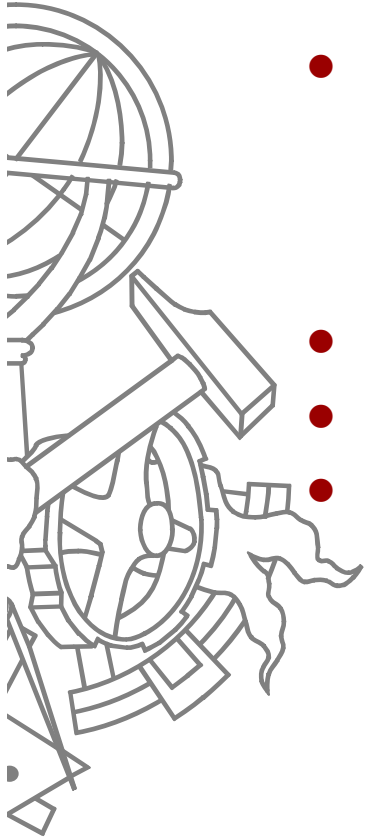
- Hibernate e nHibernate
 - www.hibernate.org
- LINQ
 - <http://msdn.microsoft.com/netframework/future/linq/>
- JDO
 - <http://java.sun.com/products/jdo/>
- FastObjects j2
 - http://www.versant.net/eu_en/products/fastobjects_j2/
- FastObjects.NET
 - http://www.versant.net/eu_en/products/fastobjects_net/
- Prevayler
 - <http://www.prevayler.org/wiki.jsp>

Notas e questões sobre SQL



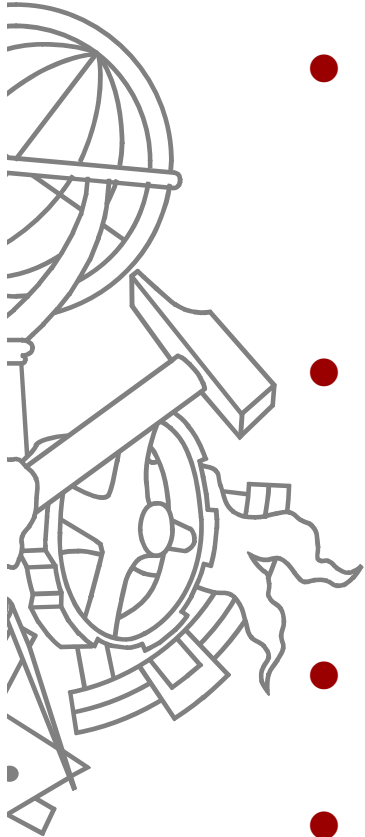
- Código SQL **apenas** nas classes da camada de acesso a dados
- Não esquecer que existem *views* no SQL...
 - Facilita a construção e compreensão de *strings* de SQL no código
 - Permite reutilização de “lógica”
- Lógica de negócio no SQL?
 - Os programadores estão à vontade com SQL?
 - Eficiência e performance
- *Stored procedures* ou SQL dinâmico?
- Usar *Stored procedures*
 - apenas para operações CRUD?
 - implementar lógica de negócio?

Notas sobre SQL



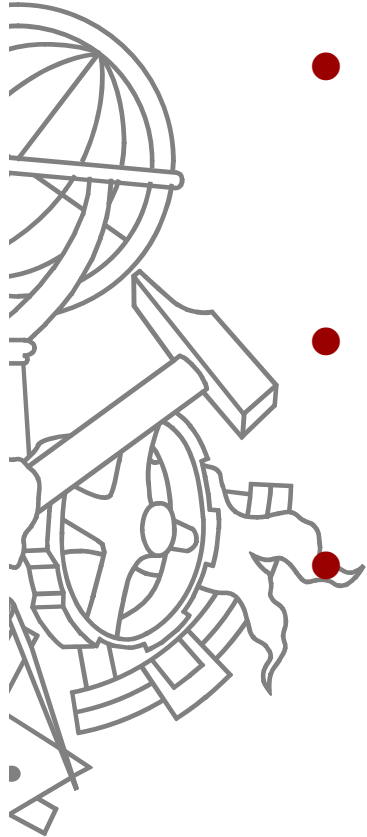
- Sempre que possível usar parâmetros nas strings de SQL em vez de concatenação
 - Performance
 - segurança
- DELETE é um comando pouco utilizado
- Operações de INSERT não provocam problemas de lock
- Operações de UPDATE devem ser cuidadosas
 - Acessos concorrentes
 - Actualizações incrementais
 - Ex: actualizar stock de um produto
 - ~~● UPDATE Products SET Stock = 10~~
 - UPDATE Products SET Stock = Stock + 2

Como garantir a consistência dos dados?



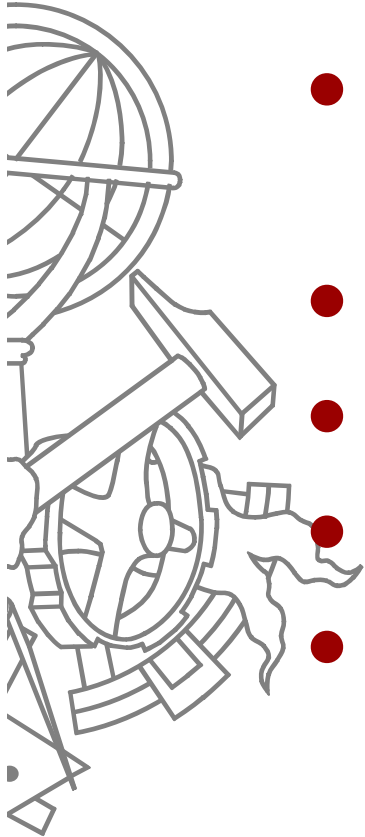
- Necessitamos de boa escalabilidade e capacidade de resposta
→ Lock Optimista
- O utilizador não pode perder as alterações que fez nos dados
→ Lock Pessimista
- Após detecção de conflito em Lock Optimista como corrigir?
- Implementar lock físico ou lógico?
 - timeout

Como tratar a interacção com o utilizador?

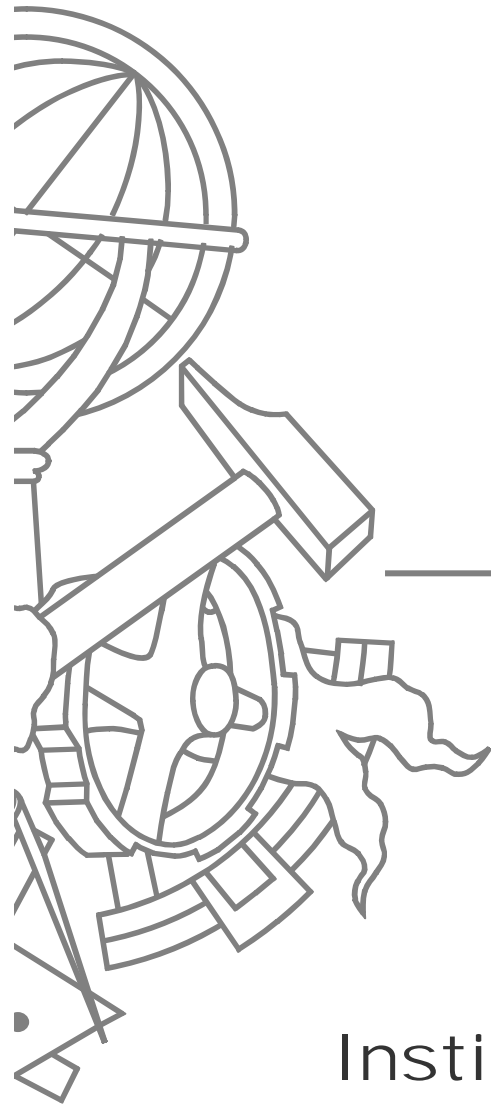


- Garantir separação entre a interacção com o utilizador e os dados da aplicação
→ Model – View – Controller
- Permitir várias visualizações dos mesmos dados
→ Model – View - Controller
- A apresentação é uma espécie de transformação dos dados do domínio
→ Transform View
- Permitir diferentes “skins” na mesma aplicação
→ Two Step Transform

Outras questões



- Gerir complexidade do projecto (nº de classes/métodos)
- Interoperabilidade de plataformas
- Integração de aplicações
- Multi-canal
- Segurança, autenticação, autorização e *single sign-on*
- Suporte para múltiplas BDs



Padrões de Aplicações Empresariais

Paulo Sousa

Engenharia da Informação
Instituto Superior de Engenharia do Porto