Frameworks de persistência são utilizados para realizar o mapeamento de dados de aplicações desenvolvidas no paradigma orientado a objetos para permitir seu armazenamento em bancos de dados relacionais e por isto são também denominados *frameworks* de mapeamento objeto-relacional. Diferentes *frameworks* em diferentes linguagens apresentam recursos e soluções semelhantes, bem como características e limitações próprias. O objetivo desta pesquisa de iniciação científica é comparar, de forma qualitativa, diferentes *frameworks* de mapeamento objeto-relacional através da implementação, em cada *framework*, de problemas baseados em padrões de análise e na comparação crítica dos resultados obtidos. Esta comparação busca identificar como cada *framework* implementa certos mapeamentos, qual a facilidade de aprender a usar este *framework*, e se existem limitações para o que é possível de ser reproduzido no contexto dos padrões selecionados. Para isso foram escolhidos os três *frameworks* mais difundidos de três linguagens diferentes: para a linguagem Java foi escolhido o JPA, para a linguagem Python o SQLAlchemy, e para a linguagem Ruby o ActiveRecord, que até o presente momento ainda não foi estudado. Para cada *framework* foi implementado o mesmo conjunto de padrões de análise especificados em UML: *Account, Transaction* e *Address Book;* e suas evoluções para os padrões *Multilegged Transaction*, *Summary Accounts* e *Party*. Os padrões de análise foram selecionados da literatura pela sua simplicidade e difusão de uso, representando soluções típicas para problemas recorrentes em sistemas de informação. A partir destas implementações foram realizadas algumas comparações que, até o presente momento, permitiram concluir que a grande maioria dos recursos encontrados nos *frameworks* estudados possuem implementações e usos similares, apresentando poucas diferenças na forma como que fazem o mapeamento objeto-relacional. Por exemplo, embora haja diferenças na forma de realizar este mapeamento – seja a partir de anotações(no caso de Java+JPA) ou utilizando *mappers* e definições de tabelas (no caso Pyton+SQLAlchemy), estas abordagens se aproximam quando se analisa a forma com que cada atributo de uma tabela é definido, e propriedades como chaves primárias, chaves estrangeiras e domínios seguem padrões similares, o que torna ainda mais direto o aprendizado de um segundo *framework* para quem já conhece um primeiro. A pesquisa ainda está em andamento e por esta razão ainda não apresenta resultados definitivos quanto às limitações de cada *framework* e os impactos causados pelo uso de uma linguagem tipada e compilada como o Java em comparação à uma linguagem dinâmica e interpretada como o Python.