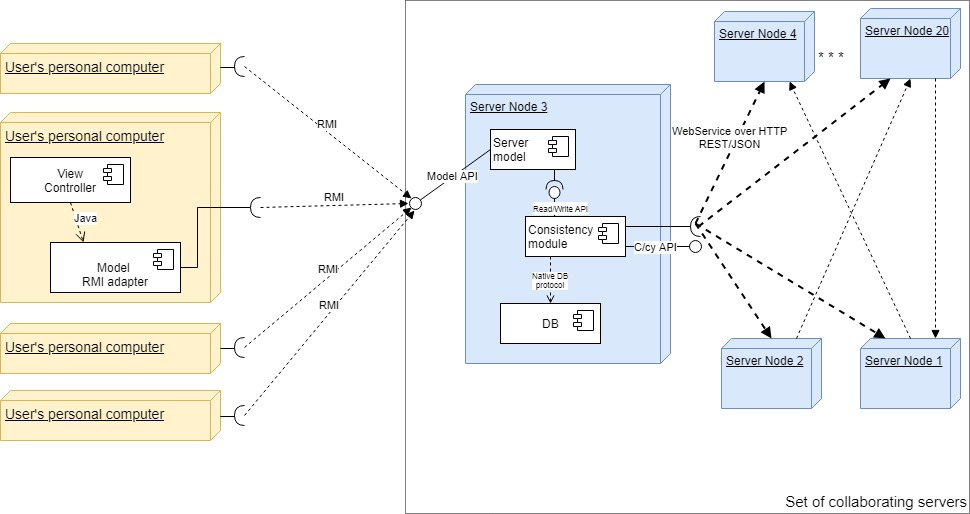
Pool Access System

# Component-deployment strategy



## Description

The primary goal of the deployment strategy is to provide (unbounded) scalability to the application. This is achieved by using a set of collaborating servers. The servers are peer (there is no hierarchy among servers). We assume that each server can serve zero to *n* concurrent users. Therefore, the system requires at least N/n servers, where N is the number of concurrent users. If N = 10’000 and n = 50, then there must be at least 200 servers. Each user is assigned to one of the servers so that each server has *n* registered users at most. To link client computers with the server, Java RMI (remote method invocation) is used. In turn, each server has its own replica of the database (DB holds data about customers and employees). Also, each server is connected to *y* other servers. The choice of *y* is dependent on the size of the database and the properties of the network (throughput and latency). In this example, we propose to set *y* equal to 6.

When some ‘write’ event happens (i. e. a swimmer checks-in, a swimmer’s profile is updated, or a new employee is added), the respective model class of the server calls the ‘write’ method of the Consistency module of the same server. The Consistency module inserts the data in the local database (DB) and sends the update request to *y* = 6 other servers. Each of these servers updates local database and retransmits the update request to another 6 servers. This relay continues until all servers (200) are updated. In this case, total coverage is achieved in just three steps (6^3 = 216 > 200). Although, the proposed architecture does not provide 100% consistency all the time, the inconsistency problem, in our opinion, will be tolerable. We think this way because the maximum period of inconsistency will be less than one second (under normal circumstances) which is absolutely acceptable from the business logic perspective.