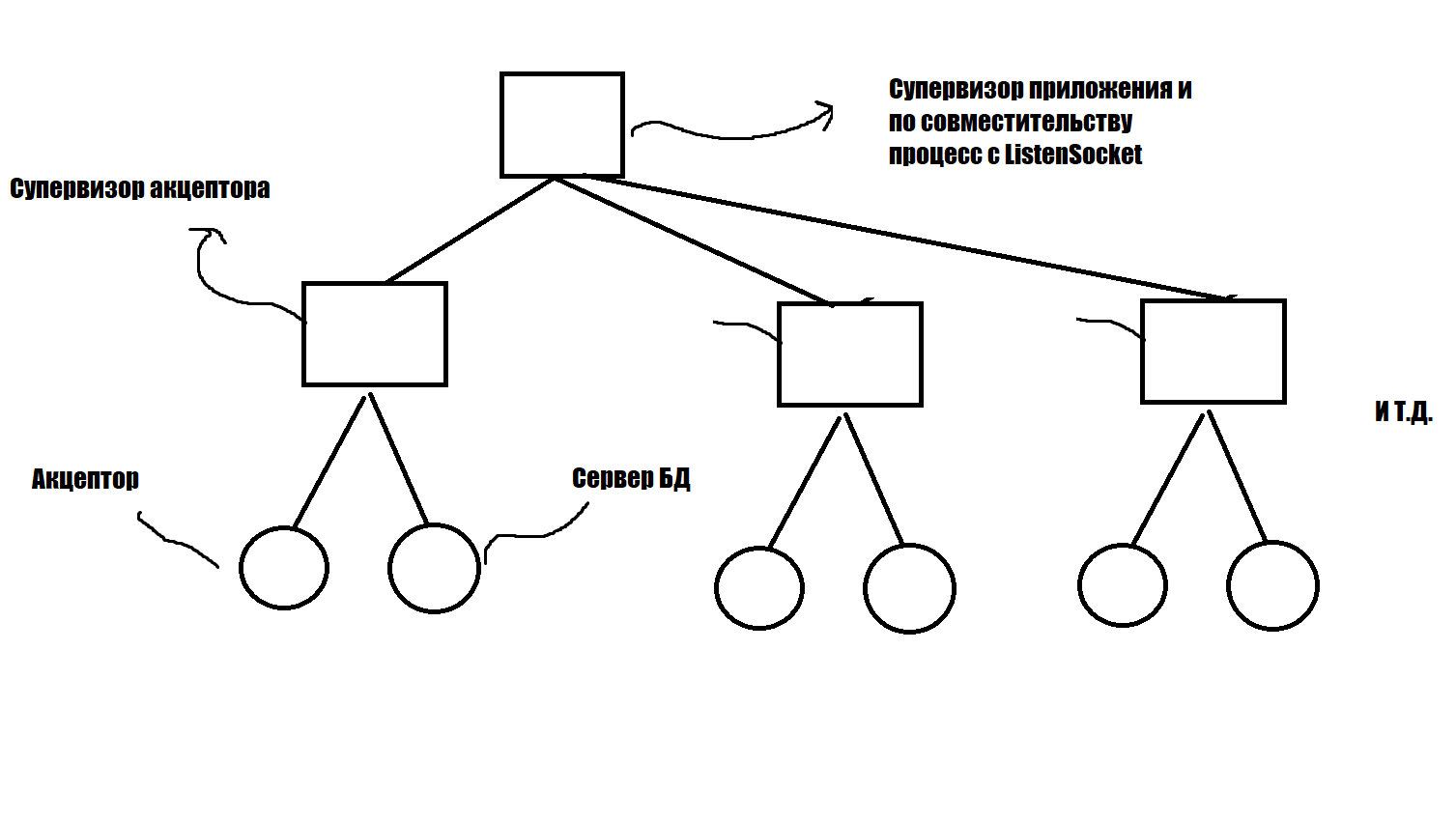
## Идеальная цель



Где:

* Супервизор приложения — читает конфиг, запускает gen\_tcp:connect(), запускает супервизоры акцепторов;
* Супервизор акцептора — подготавливает процессы, занимающиеся обработкой пользовательских запросов;
  + Акцептор — рабочий процесс обработки одного запроса. Содержит в состоянии Pid сервера БД — т.е. является клиентом БД;
  + Сервер БД — процесс-сервер, который предоставляет интерфейс доступа к данным приложения, абстрагируясь от СУБД.

## Поэтапная реализация

1. Пусть все акцепторы будут работать с одним сервером БД – Eredis;
2. Переписать функции модуля СХД для работы с Redis;
3. Переписать ListenSocket как супервизор, принимающий на вход сигнал стоп. Тогда дерево будет иметь примерно такой вид:

|  |
| --- |
| * Супервизор приложения   + Клиент Eredis   + Акцептор 1,   + Акцептор 2,   + …   + Акцептор N |

1. Написать супервизор акцептора. Он должен принимать stop-сигнал. Должен генерировать 1 акцептор и 1 сервер Eredis;
2. Переписать Супервизор приложения под новые нужды;
3. Написать обобщённый сервер БД:
   1. Вне зависимости от СУБД он предоставляет одинаковый интерфейс акцепторам.
   2. Реализация сервера меняется в зависимости от callback-функций, скормленных ему на генерации.
   3. Callback-функции для разных СУБД пишутся в своих модулях;
   4. Модули callback-функций имеют унифицированный характер, т.е. предоставляют одинаковый API.
   5. Тогда, если я захочу перейти с Redis на Mnesia или ODBC, то необходимо будет написать отдельный модуль с callback-функциями и в коде сервера поменять импорт одного модуля на другой.