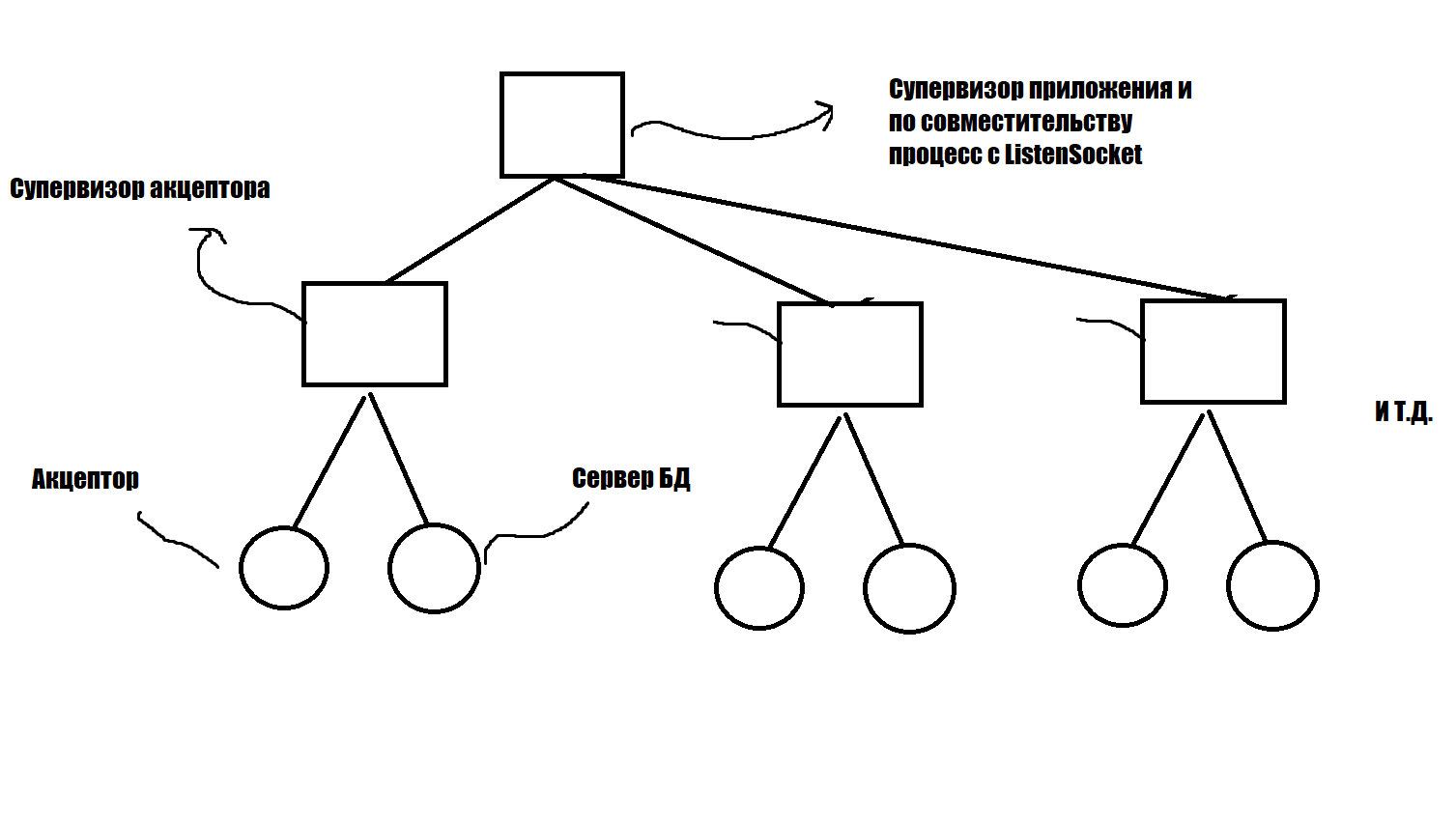
## Идеальная цель



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 error-handling в модулях:
   1. всё, что ниже контроллера отправляет коды ошибок,
   2. контроллер при получении исключения – делает throw.  
      Если ошибка критическая (не прошло сопоставление с образцом) то отправится error
   3. Во View обработка исключений будет делаться единообразно.
   4. Таким образом появится единая точка отлова ошибок и разделение кода на бизнес-логику и работу с ошибками. Подобная необходимость возникла, так как клиент в случае неполадок обязан быть уведомлён о них. Алгоритм обработки:
      1. Сначала пользователя уведомят об ошибке.
      2. Если ошибка критическая и внутрисерверная, то она отправится наверх.
4. 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-репозиторий:
   1. Вне зависимости от СУБД он предоставляет одинаковый интерфейс репозиториев акцепторам;
   2. Сам репозиторий сохраняет в состоянии соединение с БД;
   3. Функции для разных СУБД пишутся в своих модулях и предоставляют такие операции как:
      1. Работа с транзакциями
      2. Работа с соединениями
      3. Работа с каждой сущностью
   4. Тогда при смене СУБД:
      1. Сервер репозитория и его API не изменятся;
      2. Необходимо будет написать модули для новой СУБД по готовому образцу;
      3. Подменить импортируемый модуль в коде репозитория.