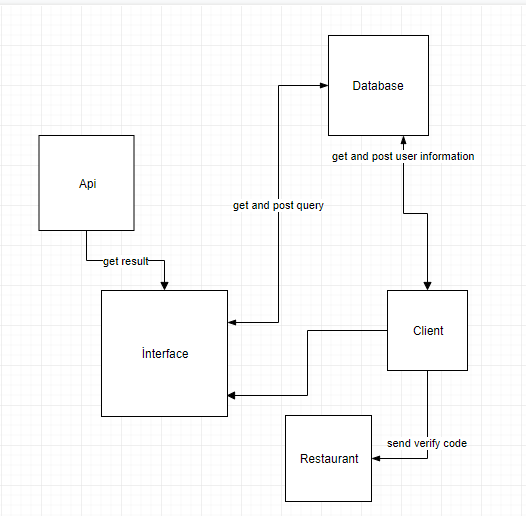
**3. SYSTEM DESIGN**

**3.1 Architectural Strategies**

* The design shall use Object Oriented Principles (OOP). The trade-off of increased code overhead and object message passing is considered justified by the gain of modularization of functionality, data encapsulation, communication through interfaces and re-use through polymorphism. In addition, the entire team is familiar with this paradigm and a design of this type will facilitate communication amongst the developers.
* Authentication will be limited to password checking on initial login and a session ID subsequently. This is considered sufficient to the low risk nature of the data.
* Application needs past restaurants for suggestions, so we save user’s restaurant history on DB. This action is goods for memory using.
* NeYesek?, will connect to Google Map Apis. Thus, the results of the restaurant will be instantaneous and we will not have to save the results in a DB.
* Level systems; when people use a suggestion, they will take point and have a level. These user informations save DB, so user informations are protected for changing.
* Discount codes valid for a certain period of time show user and send restaurant. This action is on client, when user click button, a code created and send.

**3.2 Architecture Design**



**3.2.2 Sub Components**

* Api; Google Maps Search Api.
* Database; LiteSQL.
* Client; Users’ Andoid Phones
* Restaurant; given discount restaurant