– DAO Layer – the level is used for working with data base. Requests to DB are performed by special technology for requesting – Flexible Search. These queries look like SQL queries, but have particular property for the plat-form. After that, DAO layer fills models by data from DB;

– Service Layer – the level is responsible for validation data, which come from DAO layer.

Eventually, application's flow goes to upper levels, where data are converted for user's view.

For realization bookmaker platform generally Hybris platform was used. It gives a lot of tools and possibilities for flexible developing to developers for realization the same decisions. The data base is included in Hybris platform. Also, for developing the IntelliJ IDEA IDE was used.

K.N. Susla (Francisk Skorina Gomel State University, Gomel),
Scientific adviser **V.D. Liauchuk**, Ph.D. in technics, associate professor

ARCHITECTURE OF DATA TRANSFERRING SUBSYSTEM IN THE ONLINE BOOKMAKER PLATFORM

The architecture of the application is based on MVC web-pattern, which consists of levels such as: DAO, Service, Facade, Controller and view.

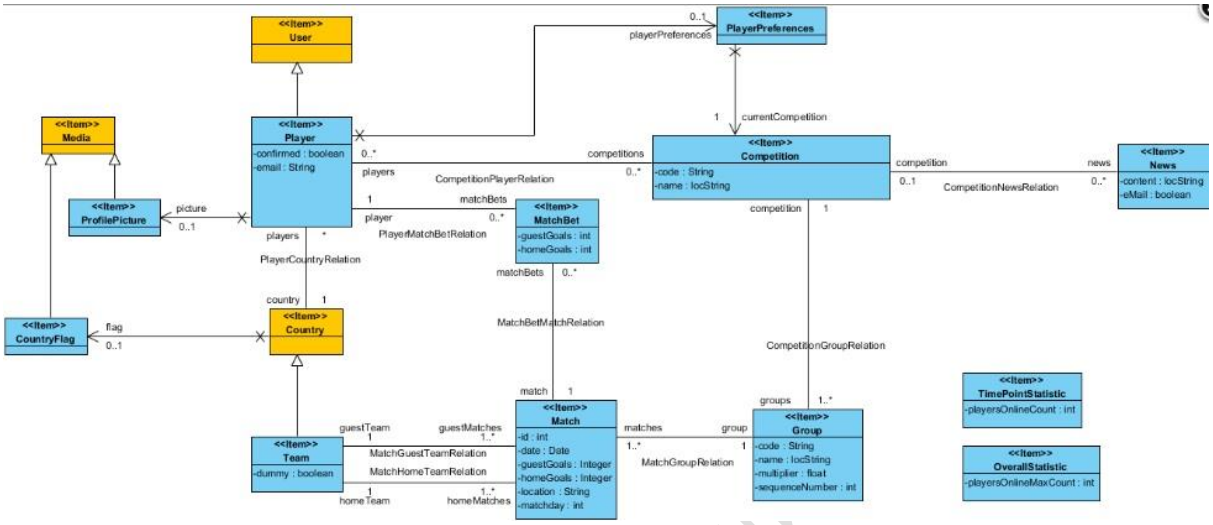
The DAO layer is responsible for working with data base. Queries for re-quests to DB are used by FlexibleSearch technology. The sample of that query is demonstrated on picture below.

```
@Override
public List<MatchModel> findMatchById(final CompetitionModel competition, final int id)
{
    final StringBuilder builder = new StringBuilder();
    builder.append("SELECT {m:").append(MatchModel.PK).append("} ");
    builder.append("FROM {").append(MatchModel._TYPECODE).append(" AS m} ");
    builder.append("WHERE ").append("{m:").append(MatchModel.ID).append("}").append("=?id ");
    builder.append("AND ").append("{m:").append(MatchModel.GROUP).append("}").append(" IN ");
    builder.append("({{ ");
    builder.append("  SELECT {g:").append(GroupModel.PK).append("}");
    builder.append("  FROM {").append(GroupModel._TYPECODE).append(" AS g} ");
    builder.append("  WHERE ").append("{g:").append(GroupModel.COMPETITION).append("}").append("=?comp ");
    builder.append(" }})");

    final FlexibleSearchQuery query = new FlexibleSearchQuery(builder.toString());
    query.setNeedTotal(true);
    query.addQueryParameter("id", Integer.valueOf(id));
    query.addQueryParameter("comp", competition);

    return flexibleSearchService.<~> search(query).getResult();
}
```

Picture 1 – Query for finding matches by Id

This method gets some match by id from data base and converts result into object of suitable model type (MatchModel).

Also, the application's architecture contains models which interact among other. There is standard kit of models for implementation simple online book-maker platform.



Picture 2 – Main schema of data models

For testing application, the couple of test-cases were written for DAO and Service layers which consist of Unit-tests and Integration-tests.

M.S. Zaletsin (Francisk Skorina Gomel State University, Gomel),
Scientific adviser **V.D. Liauchuk**, Ph.D. in technics, associate professor

THE CONCEPT OF A CLOUD OBJECT STORAGE

Today it is difficult to argue with the fact that technologies based on cloud computing are in high demand and are actively developing. Private «clouds» are intended for use within the company. They may belong to the enterprise itself or be hosted by the provider.

The first deployment model provides more control and more security, because the infrastructure components and customers are situated in the same organization. Each detail is optimally adapted to the needs of this company. However, this approach to IT infrastructure implies a significant resource cost.

In the second case, cloud storage is provided as a web service, for example, Amazon S3, Microsoft Azure Blob Storage, Google Cloud Storage. These services are based on the object model.