The site header contains menu navigation for product categories, store opening hours and contact phone numbers. The site’s footer contains contact information, links to social networks, office address, legal information, link to the store certificate. The information page of the store displays information about the company's business, terms of ordering goods, terms of payment for goods, terms of delivery of goods, contact details, legal address, location of the office on the map.

The project under development was implemented with all customer requirements in mind, hosted on a web hosting service and regularly maintained by the administration of the site.

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

DEVELOPMENT OF DATA TRANSFERRING SUBSYSTEM IN THE ONLINE BOOKMAKER PLATFORM

The title of the project is: developing online bookmaker platform, which is based on e-commerce platform – Hybris. The project was created for more convenient and comfortable methods which help users do bets without any problems like offline bookmaker have.

This application gives good possibilities for betting online, doing some payment transactions without any delays, tracking user’s bets and checking full schedule for all matches in online mode.

The main things in the common model of data are: matches, bets, playing teams and etc. Each type of the application is declared in special file: extention_name-item.xml. Also, there are described all inner types such as: Atomic Integer, Collection Type, Enum and Map. Beside of that, this file contains relations among models. Relations is the responsibility of Relation type which has special attributes.

The core of Hybris platform consists of Java-framework Spring. It means that all of the objects are created by Spring technologies with help special con-fig-files which create by XML format.

The architecture of project was implemented by standard pattern of web-developing: MVC. This application consists of several layers: DAO (Data Access Object) layer, Service layer, Facade layer, Controller, View. These levels are responsible for data transferring among levels from user view (browser, mobile application and etc.) to data base or some external system and vice versa:

– 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 de-velopers 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