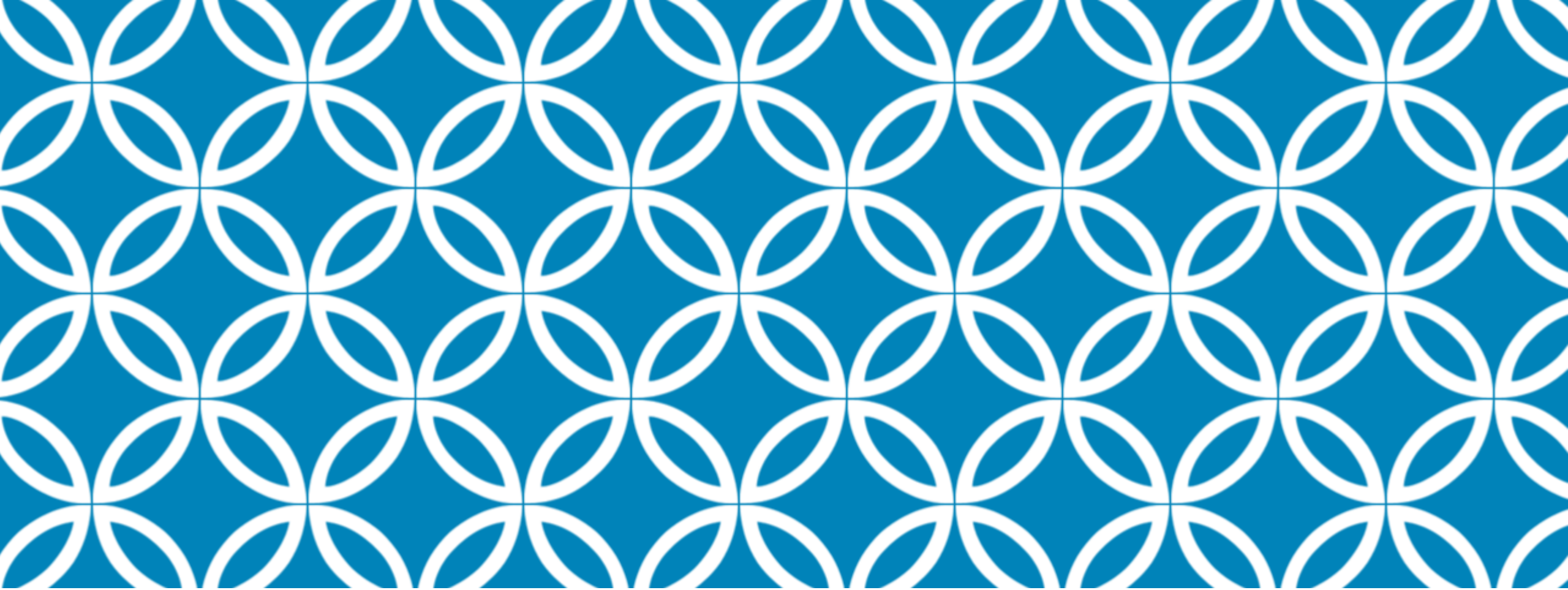


SPORTS CLUB MANAGER

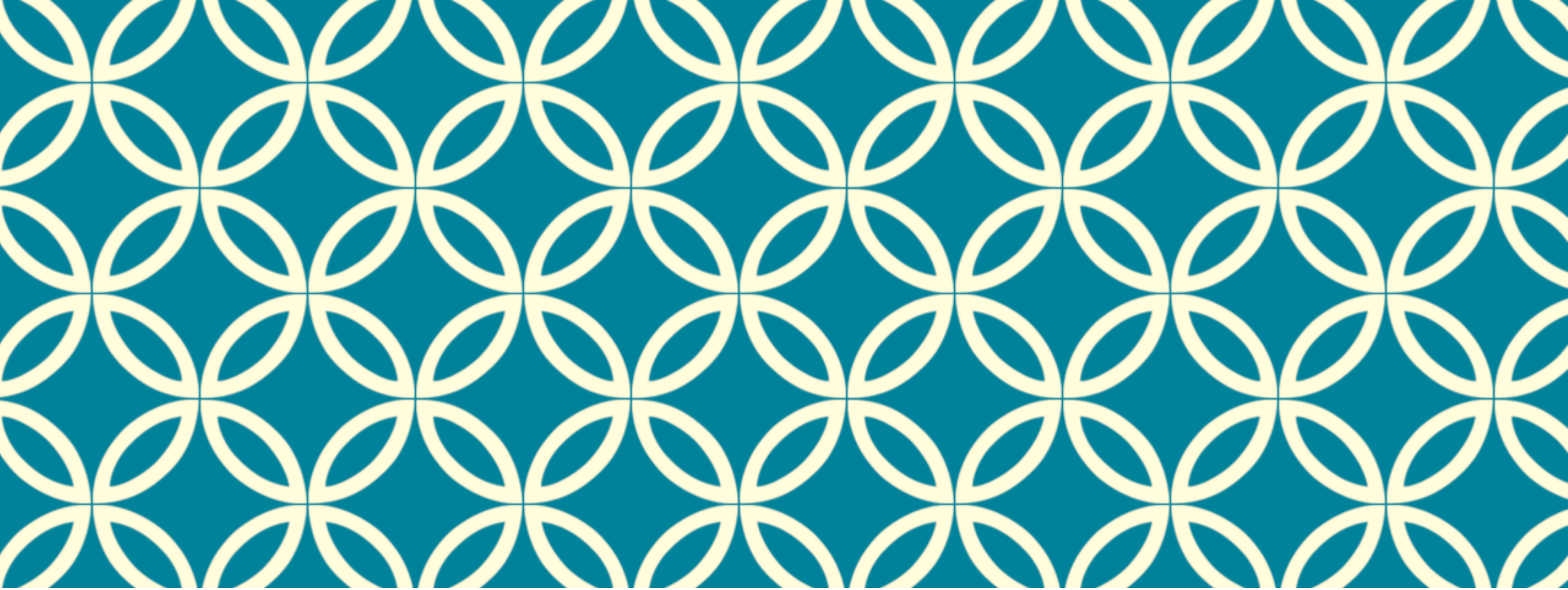
Eine wahre Geschichte

TEAM

- Christian Lins
- Dominik Gregotsch
- Markus Mohanty
- Thomas Schwarz
- Lucia Amann

AGENDA

- Datenbank und Persistenz
- RMI
- EJB
- Website
- Corba vs. Webservices
- GUI und Tests



PERSITENZ UND DATENBANK

Technologien und sonstige Hindernisse

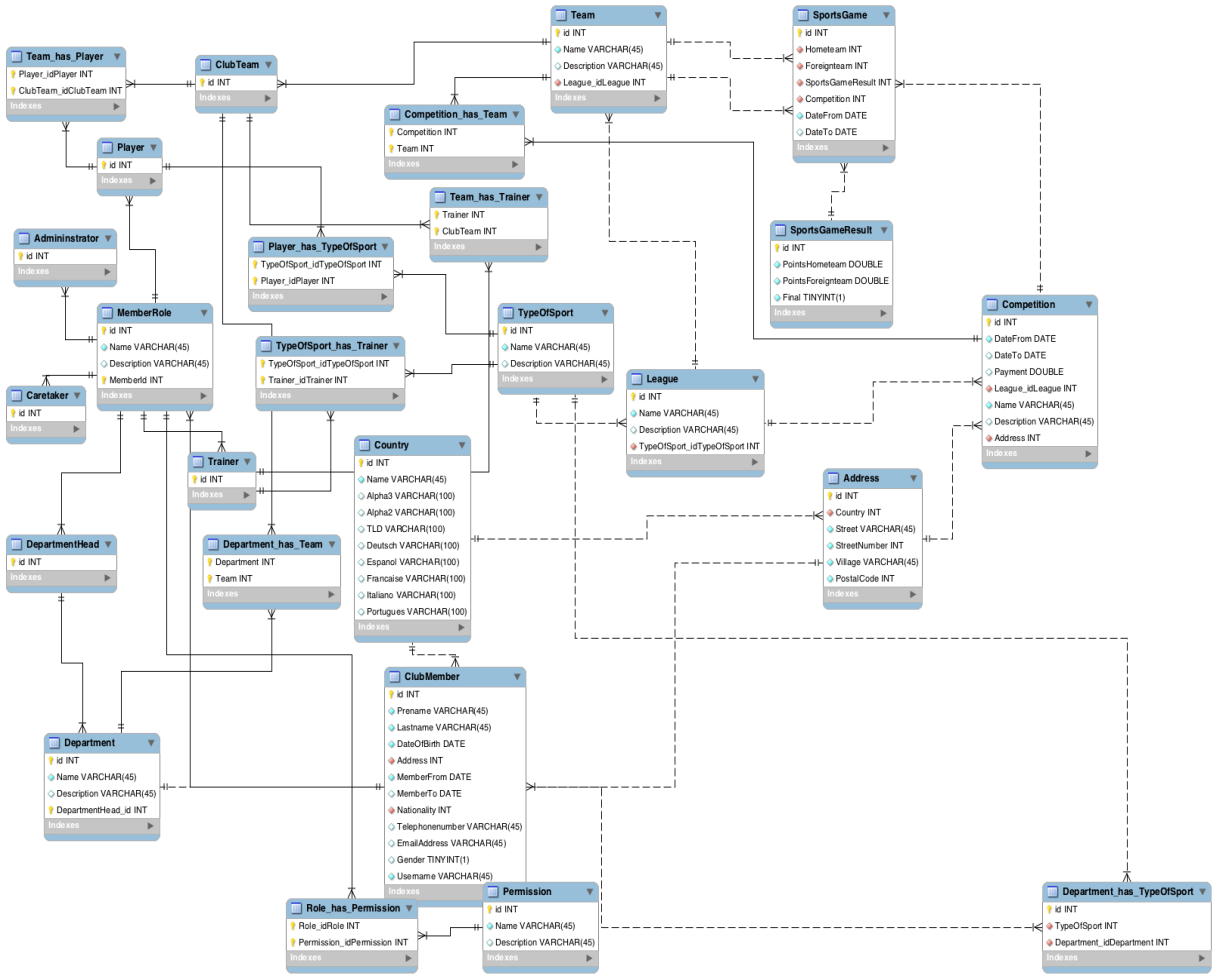
MYSQL

- Freies DBMS
- Ausreichend für kleine Projekte
- Schnelles Aufsetzen
- Via PhpMyAdmin gut verwaltbar

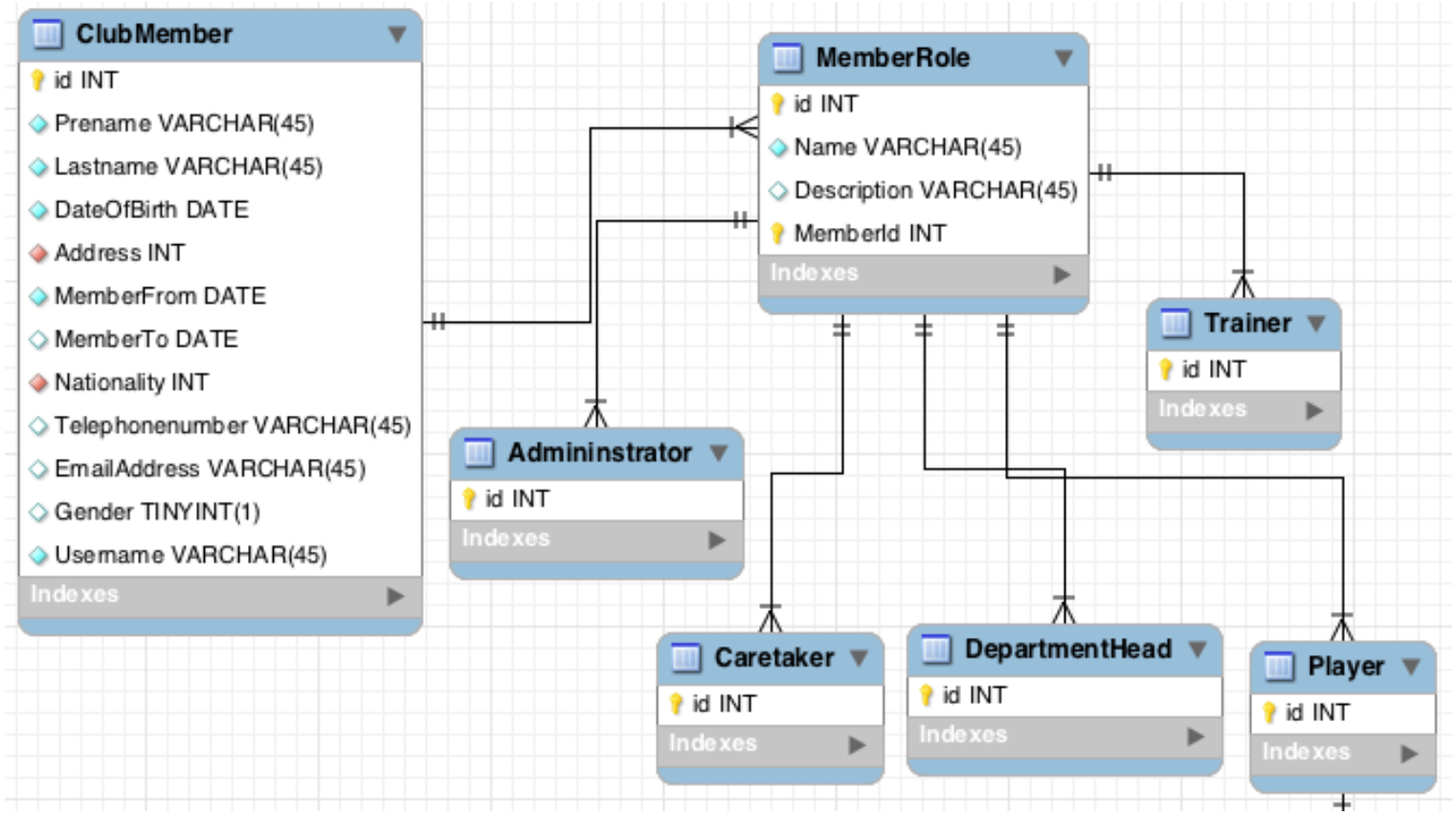
HIBERNATE

- Freies Persitenz-Framework
- Caching
- Transaktions-Verwaltung integriert

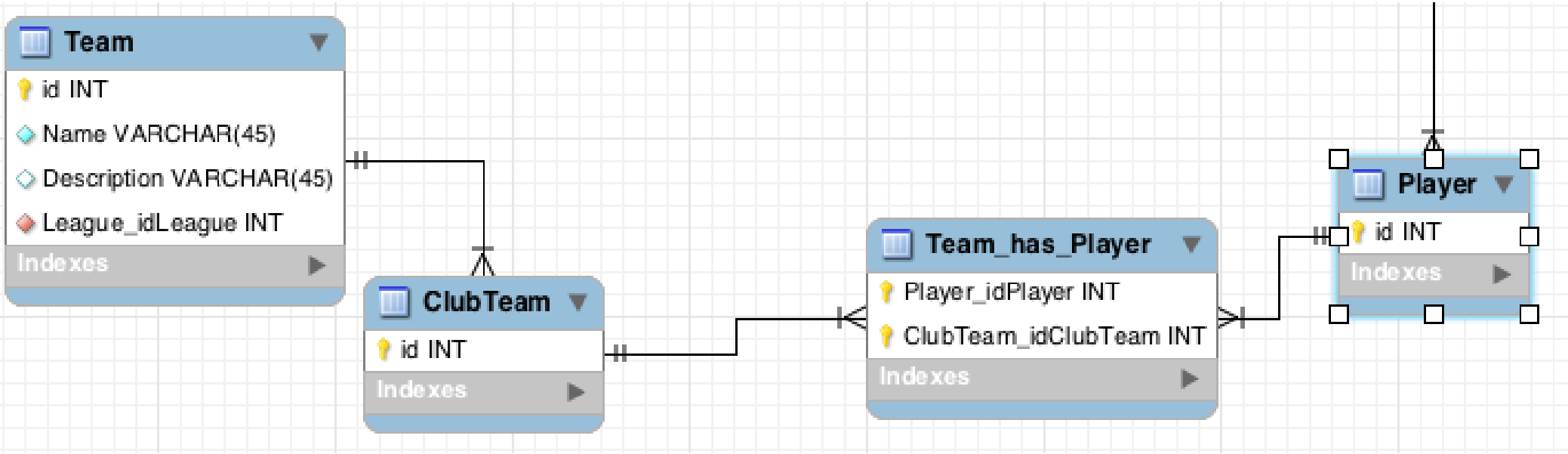
DATENBANKMODEL



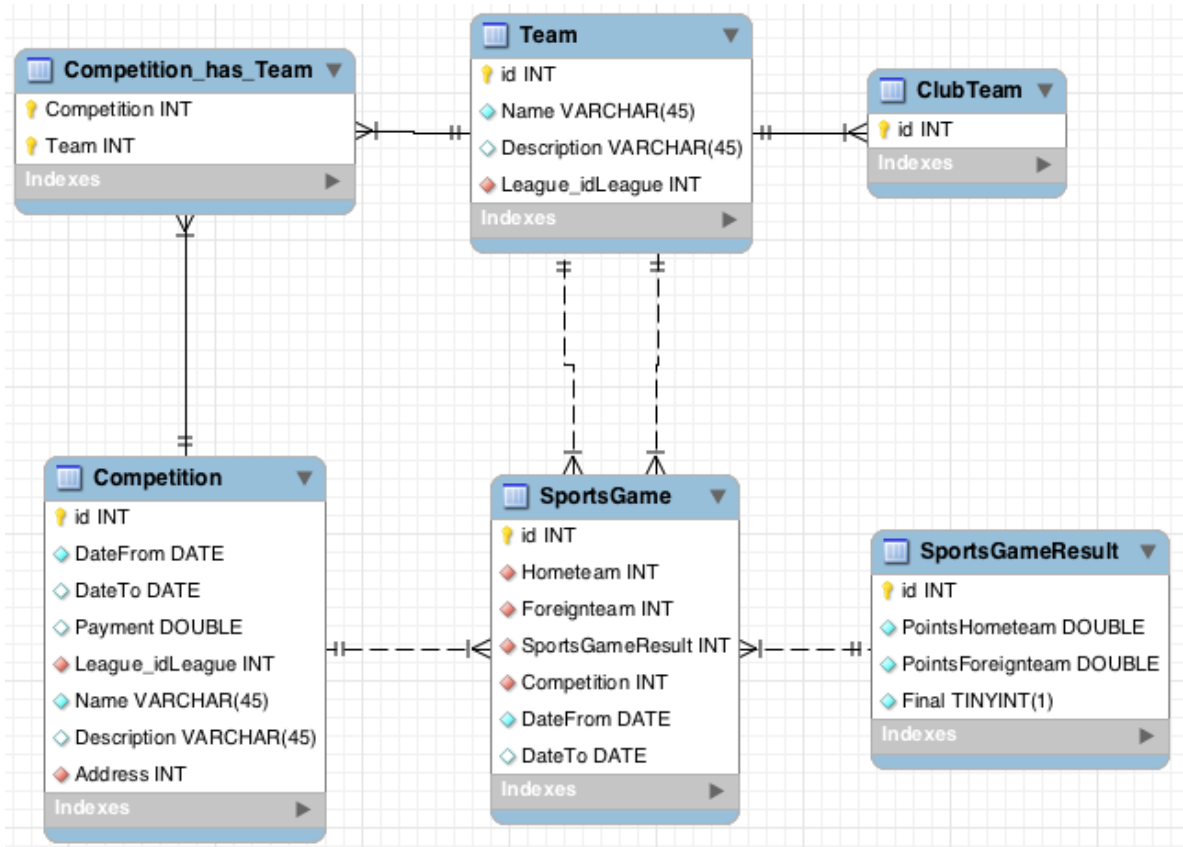
ROLLEN


ClubMember
id INT
Prename VARCHAR(45)
Lastname VARCHAR(45)
DateOfBirth DATE
Address INT
MemberFrom DATE
MemberTo DATE
Nationality INT
Telephonenumber VARCHAR(45)
EmailAddress VARCHAR(45)
Gender TINYINT(1)
Username VARCHAR(45)
Indexes
MemberRole
id INT
Name VARCHAR(45)
Description VARCHAR(45)
MemberId INT
Indexes
Trainer
id INT
Indexes
Admininstrator
id INT
Indexes
Caretaker
id INT
Indexes
DepartmentHead
id INT
Indexes
Player
id INT
Indexes

TEAM


Team
id INT
Name VARCHAR(45)
Description VARCHAR(45)
League_idLeague INT
Indexes
ClubTeam
id INT
Indexes
Team_has_Player
Player_idPlayer INT
ClubTeam_idClubTeam INT
Indexes
Player
id INT
Indexes

WETTBEWERBE

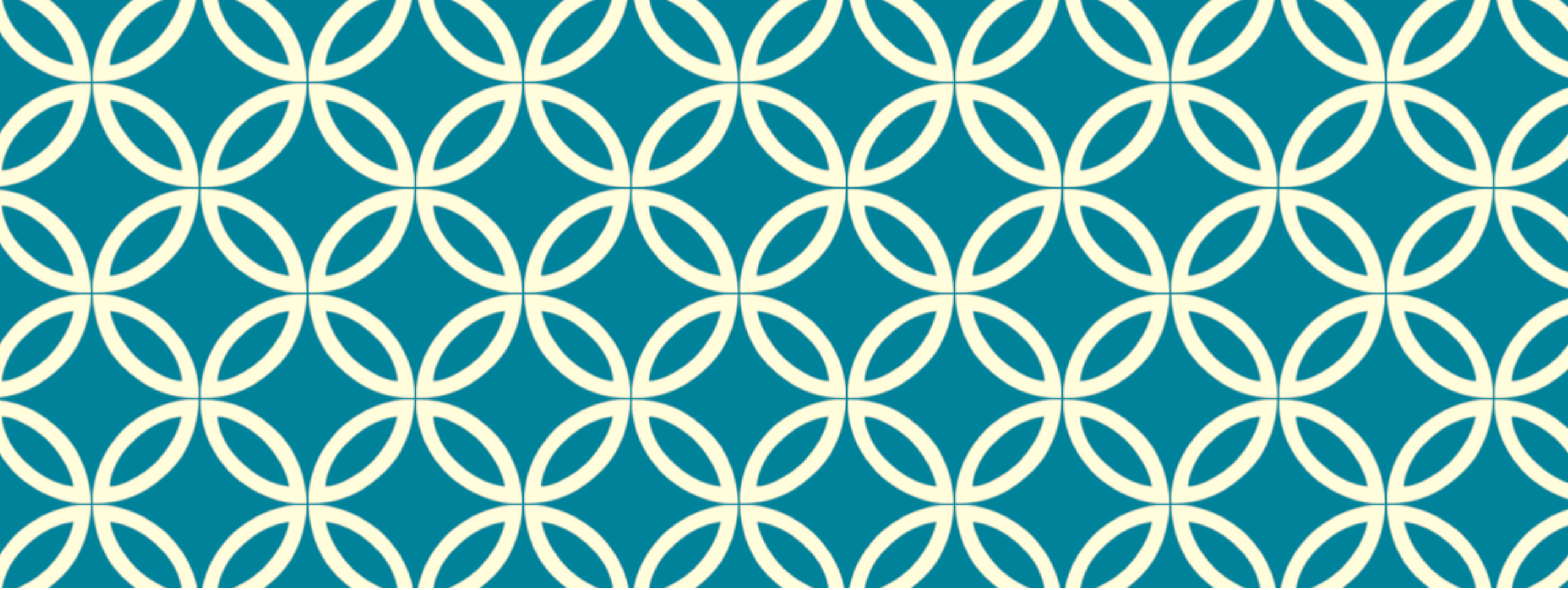


PERSITENZ

- Mittels Java Persistence 2.0
- Annotations
- Datenbank Facade
- Fetch und Save generisch

HINDERNISSE

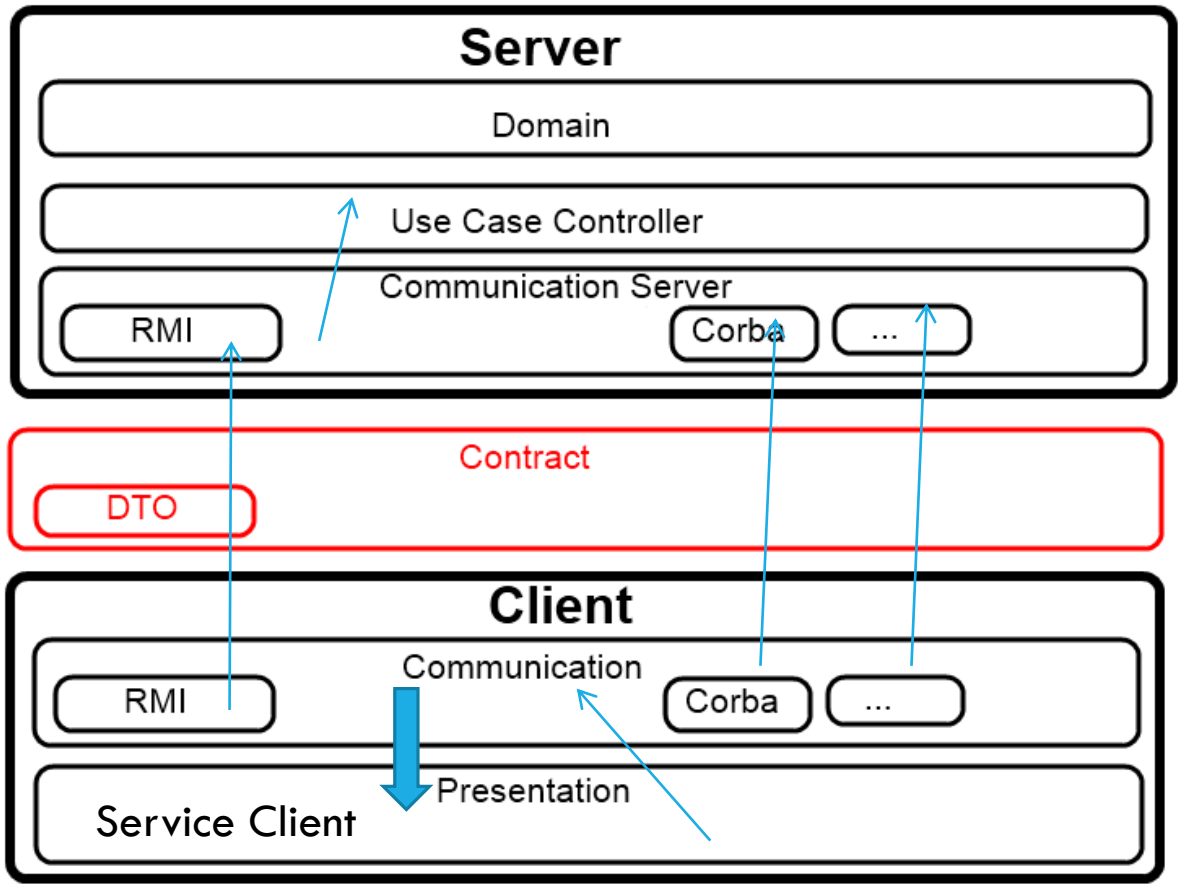
- Hibernate in Verbindung mit EJB
  - Libraries
  - Version
- Persitenzschicht auf DTOs mappen
  - Redundante Daten



RMI

Kommunikation und Implementierung

ARCHITEKTUR



BEISPIEL – START DES SERVERS

```
public class Server
{
    public static void main(String[] args) throws IOException
   {
            // start rmi-server-thread
            new Thread(new RmiServer(1099)).start();

            // start corba-server-thread
                new Thread(new CorbaServer()).start();
```

BEISPIEL – RMI SERVER

```
IRmiServiceFactory rmiServiceFactory = new RmiServiceClientFactory();
```

Remote Interface

Unicast Remote Object
- Instanziert RMI Use Case Controller
  - Z.B. „AddMatchResultRmiService“
  - Diese verweisen auf richtige „Server“-Use Case Controller

```
@Override
    public List<ICompetitionDto> getCompetitionList()
            throws RemoteException
    {
        return AddMatchResultsController.getInstance().getCompetitionList();
    }
```

BEISPIEL – CLIENT

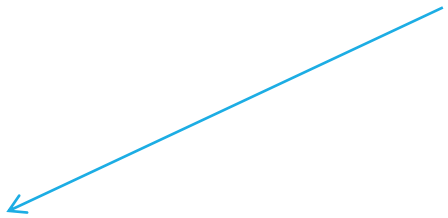
Allgemeine Use Case Controller Factories

```
public class ServiceClientFactory
{
    public static IUseCaseControllerFactory getRmiServiceClient(String host, int port)
            throws CommunicationProblemException
    {
        return new RmiUseCaseControllerFactory(host, port);
    }
```

Allgemeines Exception Handling

BEISPIEL – CLIENT

```
@Override
    public IAddMatchResultsController getAddMatchResultsController()
            throws ServiceNotAvailableException
    {
        try
        {
            return new
AddMatchResultsServiceMapper(rmiServiceClient.getAddMatchResultsService());
…
```



Mapping zwischen RMI-Services und Client Services
- Exception Handling

VOR-NACHTEILE

+ Hoher Abstraktionsgrad

+ Information Hiding

+ Einfache Erweiterbarkeit
  - Kommunikations-“Module“

- Kein einheitliches Interface für Use Case Controller

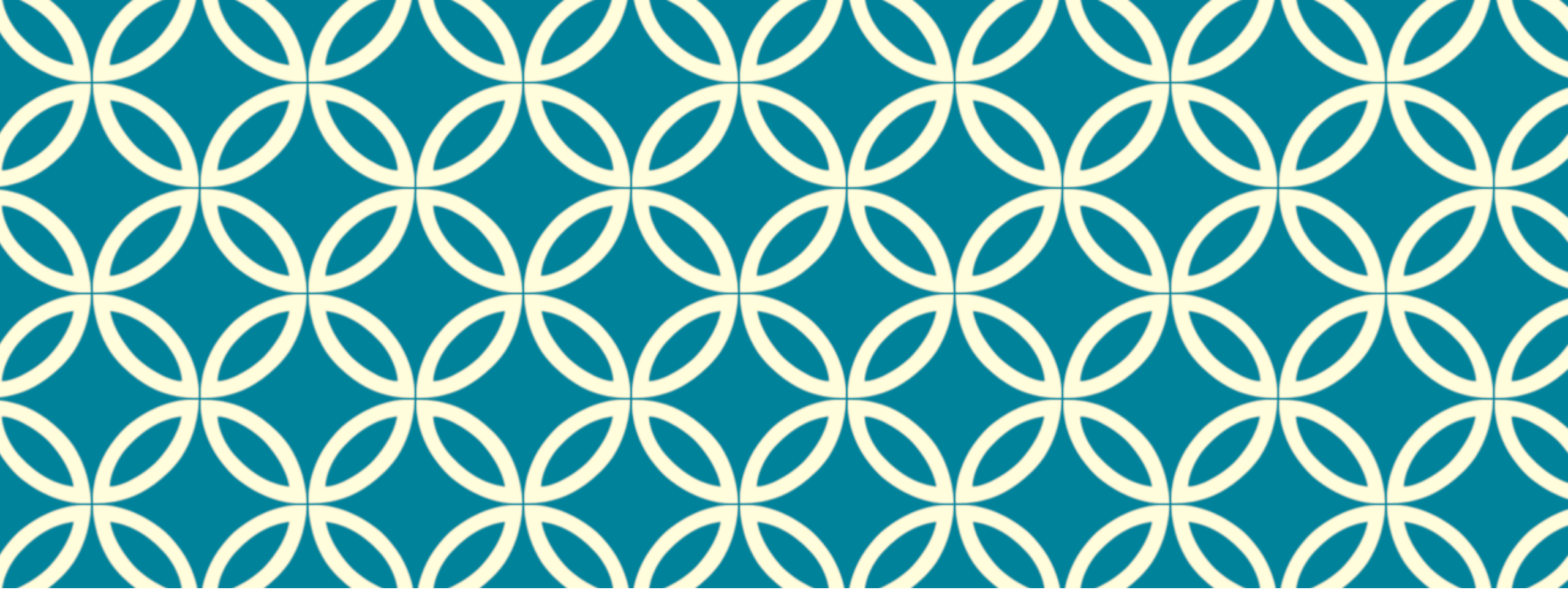
- Sehr viele Klassen und Schnittstellen
  - Pro Kommunikationsschicht „Duplikat“

- Aufwändige und fehleranfällige Use Case Controller Anpassungen

Fazit: für diese Aufgabe passend

FAZIT - RMI

- Klarer Aufbau
  - Keine Annotationen
  - Low Level
- Stabil
- Programmierung leicht fehleranfällig
- Probleme mit Netbeans



ENTERPRISE JAVA BEANS

Was ist EJB, Einsatz, Vor- und Nachteile, Umsetzung und Kritik

WAS IST EJB

- Standardisierte Komponente
- Mehrschichtige verteilte Softwaresysteme
- Einfache Bereitstellung
- Kann von mehreren Applikationen verwendet werden.

EINSATZ

Client

- Schnittstelle zur Datenbank
- Empfangen der Usecase Controller (Factory nicht möglich)

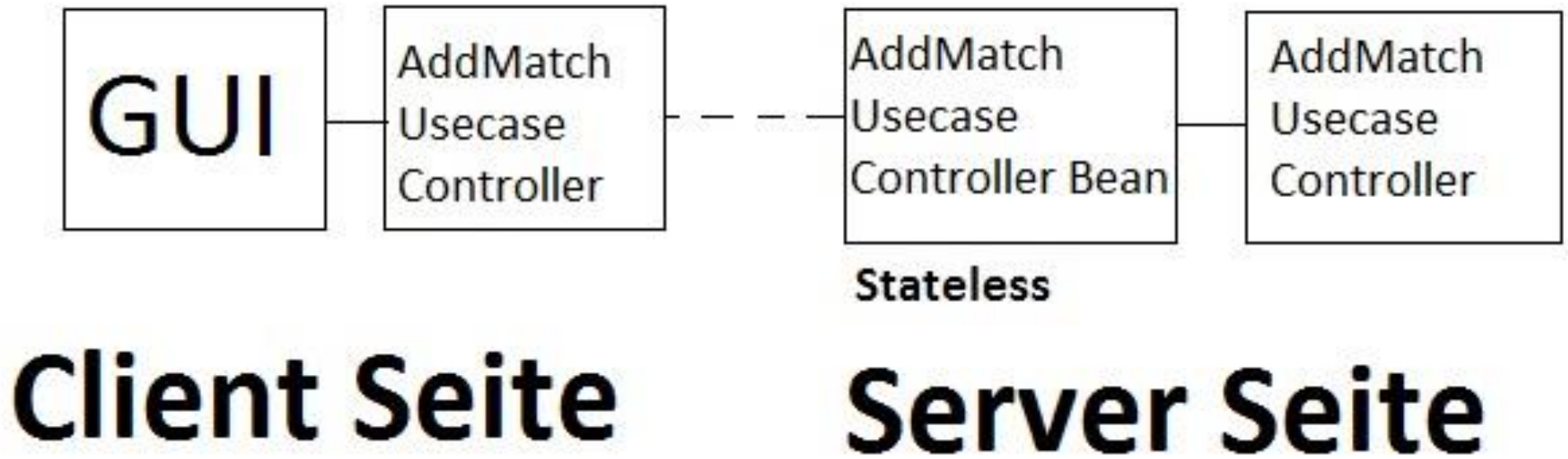
Website

- Schnittstelle zur Datenbank
- Empfangen der Usecase Controller

VOR-NACHTEILE

+ Spezifizierung
+ Enge Integration in J2EE
+ Skalierbarkeit
- Umfangreiche und komplexe Spezifikation
- Voluminöse Dokumentation
- Der Zeitaufwand für die Entwicklung erhöht sich
- EJB-Code ist komplexer
- Gefahr zu komplizierten Designs
- Spezifikationsänderungen
- Zugriff auf Ressourcenverwaltungssysteme

REALISIERUNG

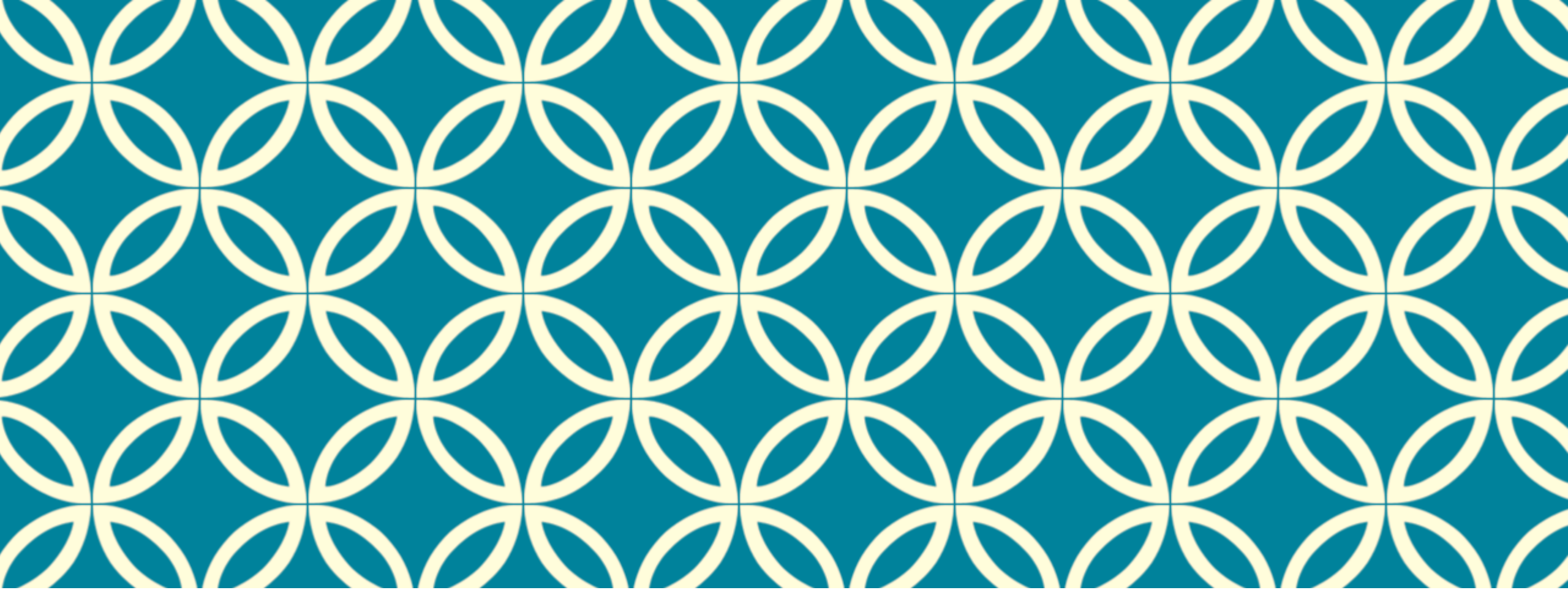


REALISIERUNG

- Versucht mit Factory
- Jeden Usecase Controller einzeln
- Stateless

FAZIT

- einfache Technik
- Factory funktioniert nicht
- Sehr mächtig
- Ohne Webserver nicht betreibbar
- Entwicklung nicht angenehm



WEBSITE

Einleitung, Designentscheidung, Technologien, Realisierung, Kritik

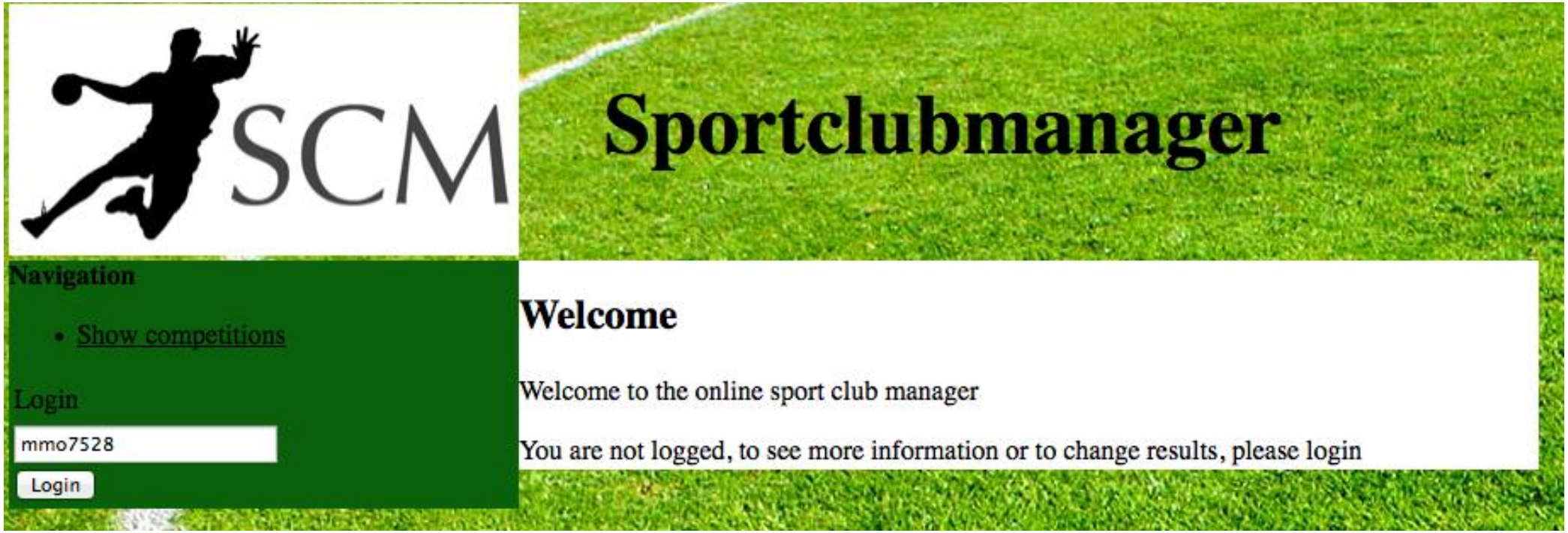
EINLEITUNG

- Darstellung ohne FAT Client
- Browser Kompatibilität
- Technologien
  - JSP
  - EJB
  - HTML
  - CSS

DESIGNENTSCHEIDUNG

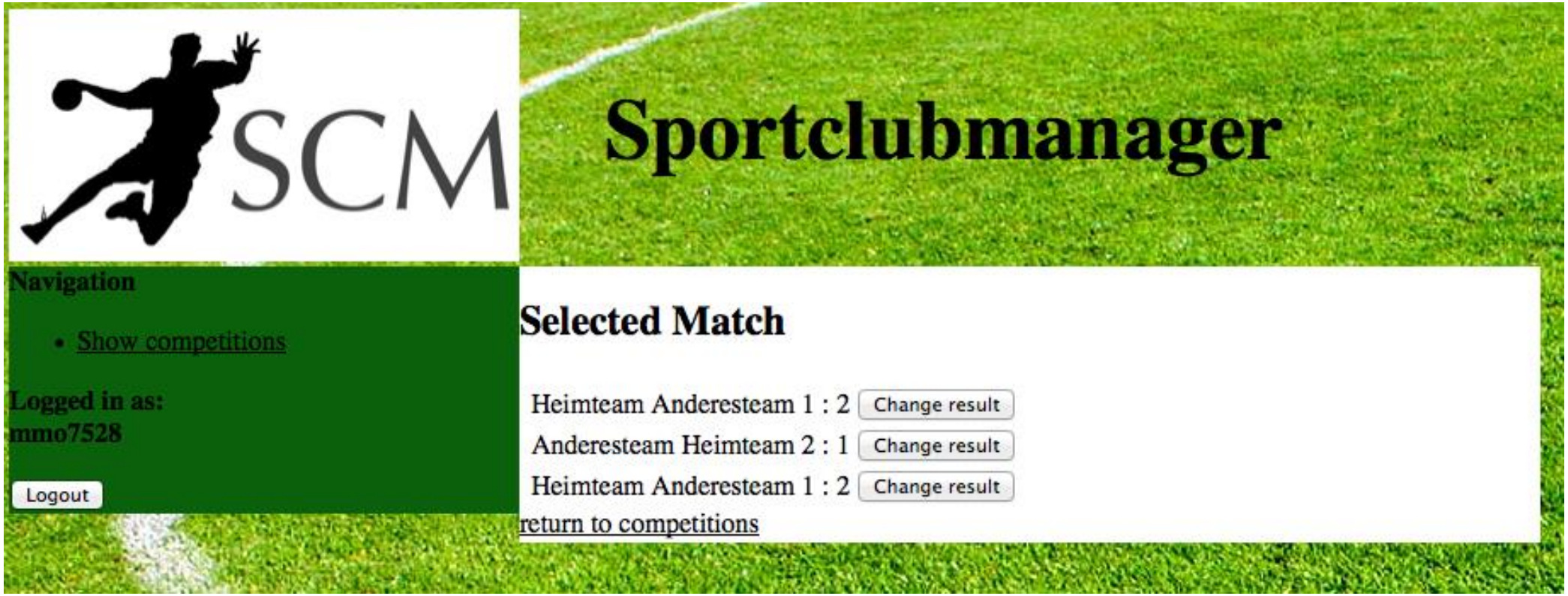
- Übersichtlich
- Bedienerfreundlich
- In jedem Browser darstellbar
- Funktionen unterstützen

DESIGNENTSCHEIDUNG



DESIGNENTSCHEIDUNG

DESIGNENTSCHEIDUNG



DESIGNENTSCHEIDUNG

TECHNOLOGIEN

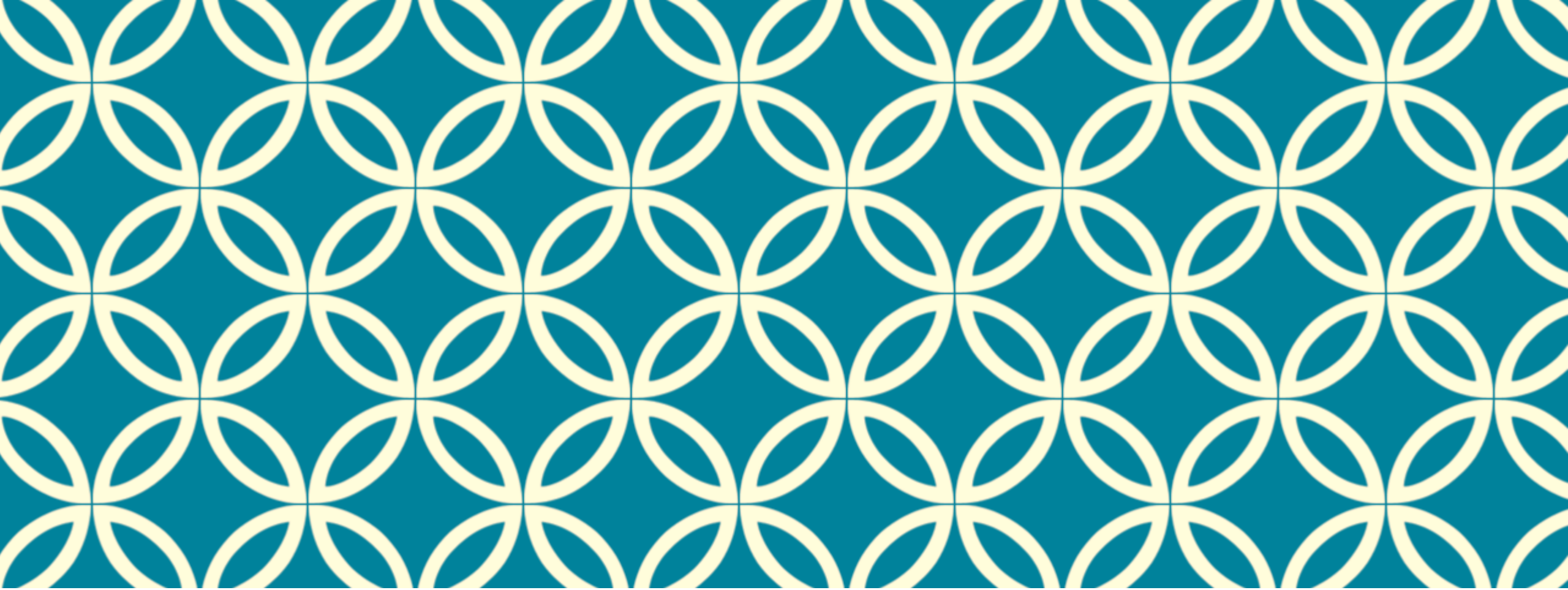
- EJB
  - Austausch der Daten
  - Bearbeitung der eingaben/ausgaben
- JSP
  - Bearbeitung der Daten zur visuellen darstellung
  - Weiter delegieren

TECHNOLOGIEN

- HTML
  - Formatierung
  - Darstellung
- CSS
  - Formatierung

REALISIERUNG

- Mischung vieler Technologien
- Zusammengeführt zu einem Großen
- Vorhandene EJB Komponente verwendet
- JSP hinzugefügt
- HTML hinzugefügt
- Keine Änderung am bestehenden Code



CORBA VS. WEBSERVICE

Beide Technologien bearbeiten das selbe Feld

Was ist besser?

INTERFACEDEFINITION

Corba IDL

```
interface MatchresultDataprovider
    {
        MatchresultListCorba getMatchresults(
                            in string typeOfSport,
                            in string league,
                            in string compdate);

     boolean isMatchresultFinal(
                            in MatchresultCorba result);
    };
```

Webservices

```
@WebService(name = "MatchSvc")
public class MatchService
{
    public List<MatchresultWs> getMatches(String
competitiondate, String league, String typeOfSport)
    {
…
     }
}
```

DTO DEFINITION

Corba IDL

```
struct MatchresultCorba
    {
        long Id;
        string Date;
        string hometeam;
        string foreignteam;
        double pointsHometeam;
        double pointsForeignteam;
    };
```

Webservices

```
@XmlType(propOrder = { "id", "date", …})
public class MatchresultWs
{
…

    @XmlElement(name = "id", required = true)
    public int getId()
    {
```

SERVER-IMPLEMENTIERUNG

Corba

Stubs müssen erzeugt werden IDLJ

Implementierung der Logik per Vererbung

Webservices

Logik wird direkt in die Definition programmiert

SERVER-DEPLOYMENT

Corba

```
Runtime.getRuntime().exec("orbd -
ORBInitialPort 2050");

String[] args1 = new String[] {"-
ORBInitialPort", "2050"};

ORB orb = ORB.init(args1, null);

… magic…

orb.run();
```

Webservices

```
Endpoint.publish("http://localhost:8080
/services", new MatchService())
```

Veröffentlichung als .war

CLIENT-STUBS ERZEUGEN

Corba

Commandozeile IDLJ …

Kein update möglich

Webservices

In VisualStudio „AddServiceReference“ und dann den Pfad eingeben

Update möglich

CLIENT-ANBINDUNG

Corba

```
String[] args1 = new String[] {"-ORBInitialPort", "2050"};

ORB orb = ORB.init(args1, null);

org.omg.CORBA.Object objRef =
orb.resolve_initial_references("NameService");

NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);

MatchresultDataprovider
matchresultDataprovider =
MatchresultDataproviderHelper.narrow(ncRef.res
olve_str("HelloObject"));
```

Webservices

```
ServiceReference1.MatchSvcClient client
= new
ServiceReference1.MatchSvcClient();
```

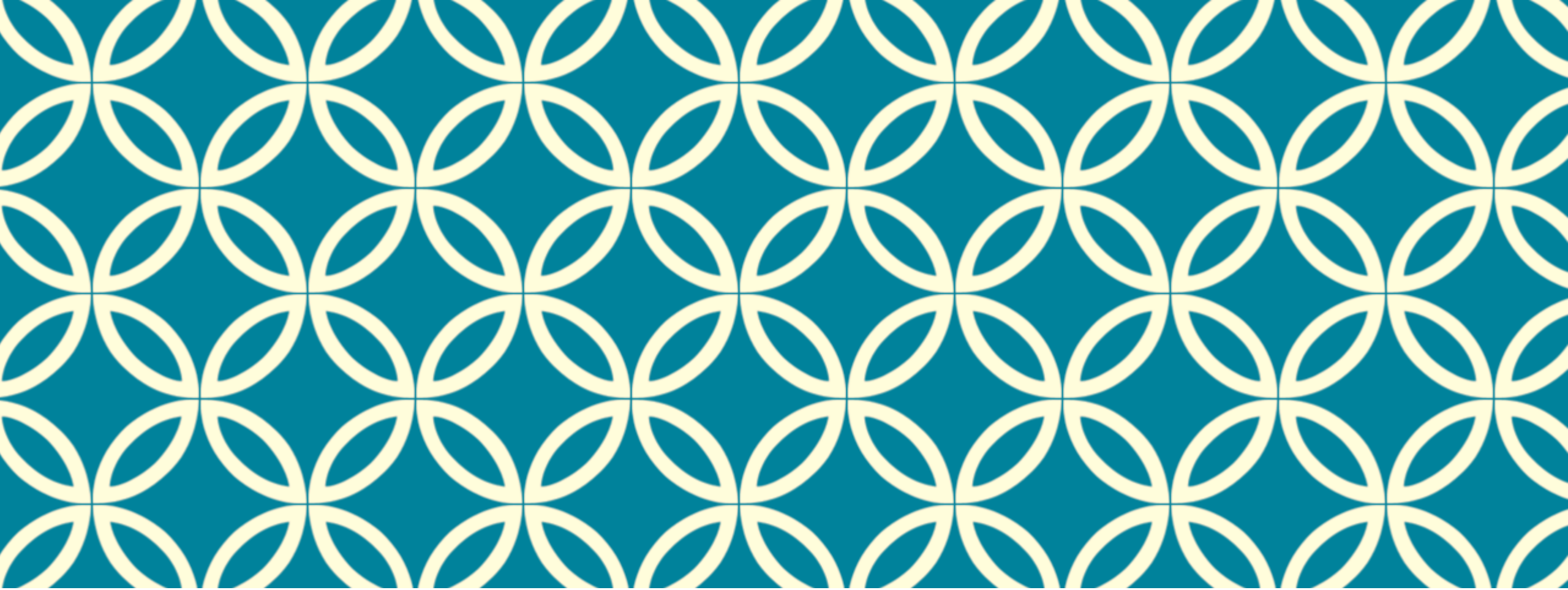
FAZIT

Corba

+ Plattformunabhängig
- Commandozeile
- IDL File und Syntax

Webservices

+ Kann alles was auch Corba kann
+ aktueller
+ Besseres Tooling, da kein IDL
+ Einfache Portierung von Corba
- Braucht einen Webserver (Glassfish, IIS, usw.)
- Für die Anbindung muss der Service aktiv sein



GUI UND TESTS

Die unendliche Geschichte

ALLGEMEINES ZUR OBERFLÄCHE

- Swing – GUI
- möglichst wenig Logik
  - UseCase Controllern
- IUseCaseControllerFactory
  - vereinfacht Technologie Anpassung
- Actions vermitteln

USECASE CONTROLLER FACTORY

```
public interface IUseCaseControllerFactory
{
    IAddMatchResultsController getAddMatchResultsController()
            throws ServiceNotAvailableException;

    IChangeCompetitionTeamController getChangeCompetitionTeamController()
            throws ServiceNotAvailableException;
    …
}
```

LAYOUT

- Basis Einstiegspunkt
- Grob Navigation über Tabs
  - Erweiterbarkeit
  - Übersichtlichkeit
- *KISS - Keep it simple stupid*

HAUPTBILDSCHIRM

GRUNDLEGENDE FUNKTIONALITÄTEN

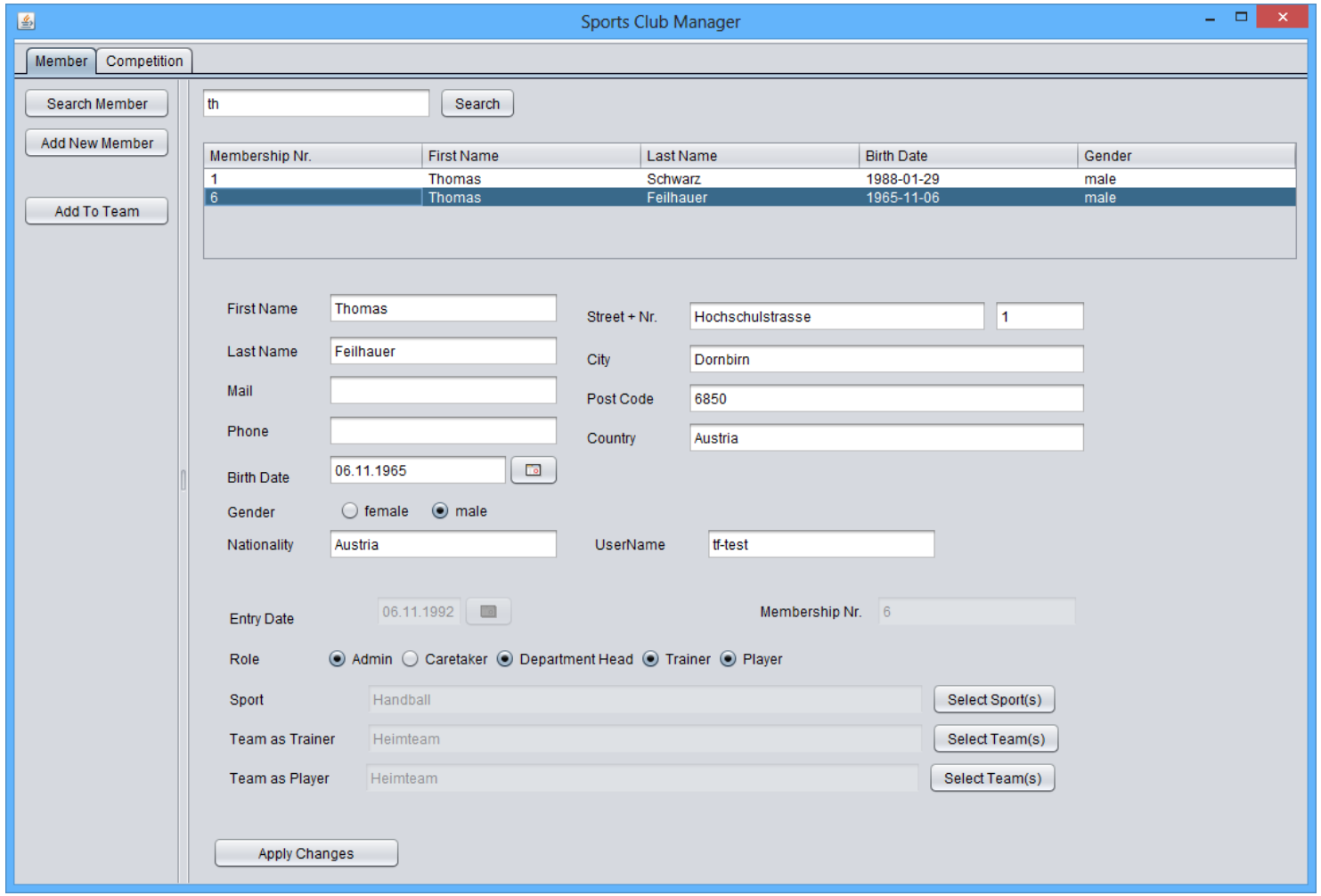
Login

Berechtigungsprüfung

- Mitglied
  - Suchen/ Ändern
  - Neu anlegen
  - Zu Team hinzufügen

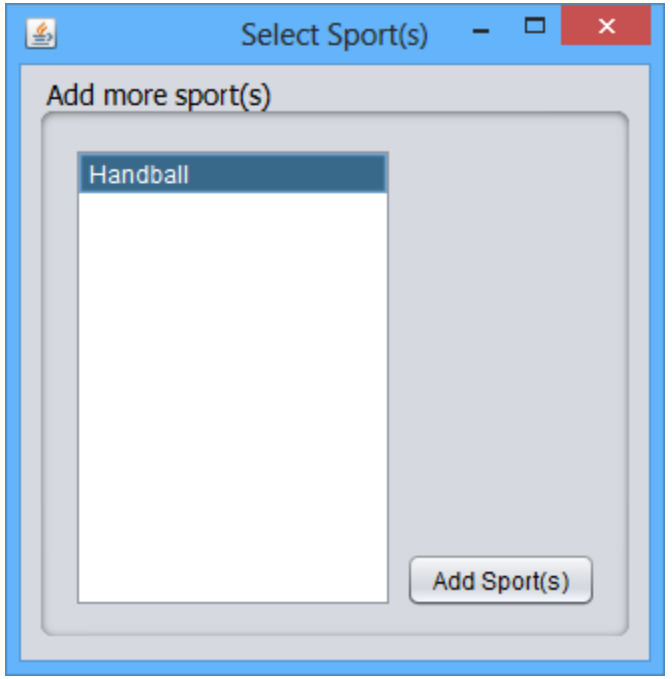
- Wettkampf
  - Neu anlegen
  - Resultate eingeben
  - Team festlegen
  - Anzeigen

MITGLIED SUCHEN/ÄNDERN

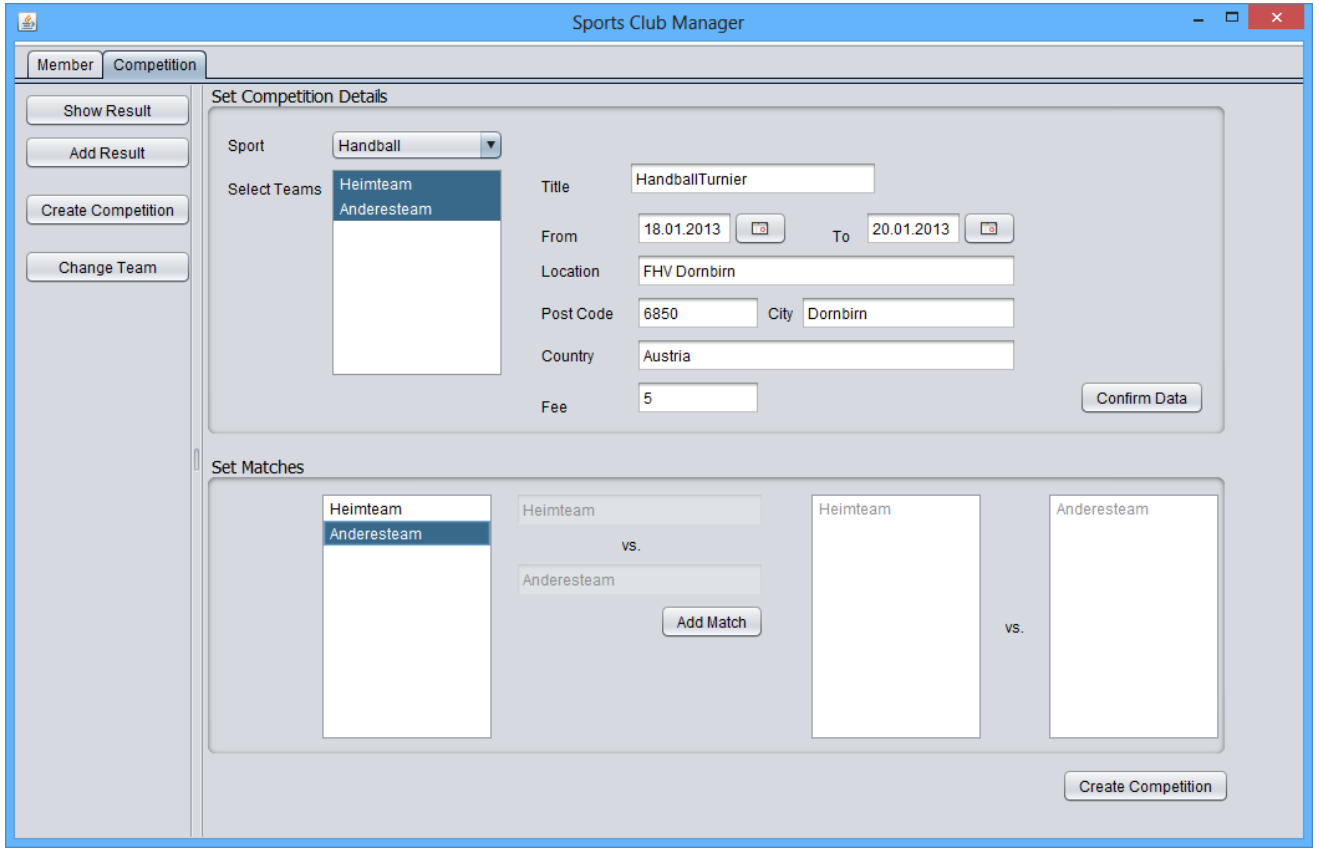


OBSERVER SOLUTION

```
public class MembershipDataPanel  extends JPanel  implements SelectedSportsValue, SelectedPlayerTeamsValue, SelectTrainerTeamsValue {

        private void btnAddSportActionPerformed(ActionEvent evt) {

                SelectSportsHelper helper = new SelectSportsHelper(controller.getAllSports(), selectedSports, this);
        }
}

public class SelectSportsHelper {

        private void btnAddSports(ActionEvent evt){…

                selectedSportsValue.sportSelected(selSports);
        }
}

public interface SelectedSportsValue {

    void sportSelected(List<ITypeOfSportDto> sport);

    List<IClubTeamDto> getClubTeamsBySport(ITypeOfSportDto sport);

}
```



WETTKAMPF ANLEGEN



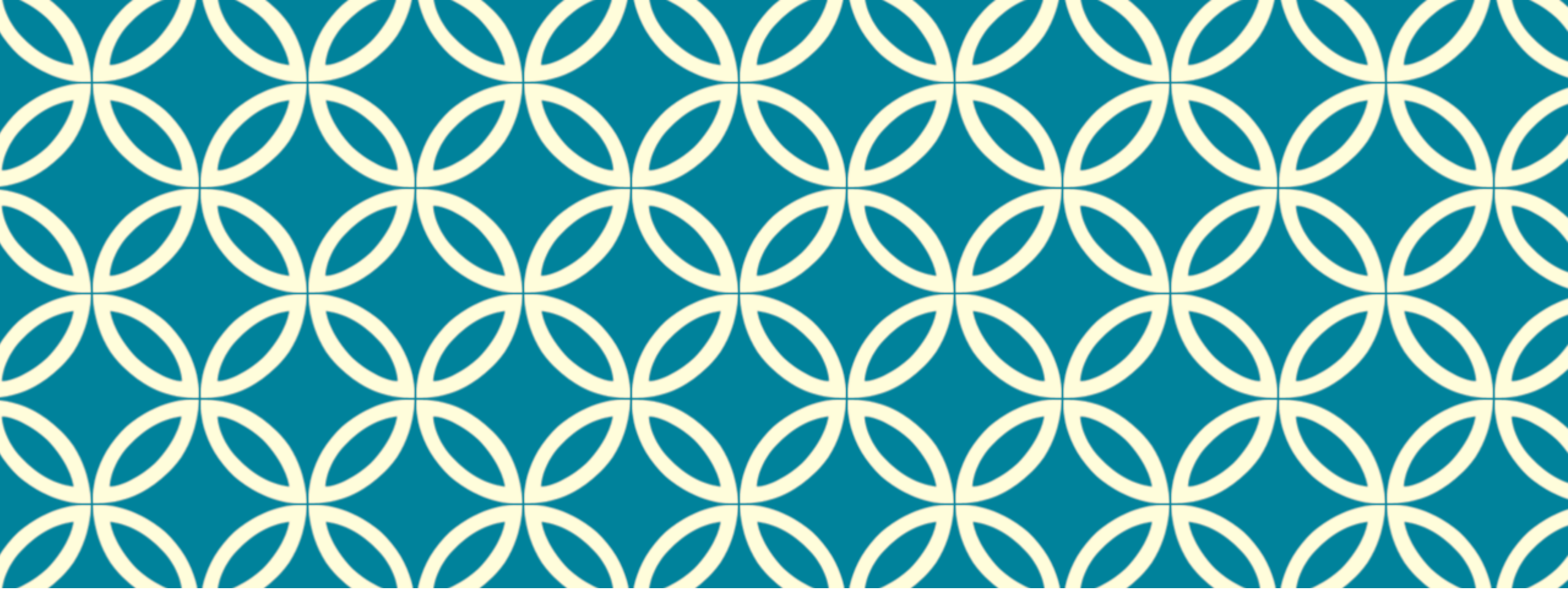
ERWEITERUNGEN

- Wöchentliche Anpassungen:
  - Mitglied zu Team hinzufügen
  - Login mit LDAP, Zugriffserlaubnis?
  - EJB

  - Wettkampfergebnisse (Corba)
  - Einladungen JMS
  - WebClient

TESTS & PROBLEME

- Kompatibilität
- Exception Handeling verbessert
- Aufwändig
  - Fehler -> mehrere Schichten
- Wiederverwendbare Forms
- Unit Tests
- Datenbank Testapplikation



LESSONS LEARNED

Das Ende

LESSONS LEARNED

- Analyse und genaue Aufteilung der Aufgabenstellung
- Konflikte durch verschiedene Technologien
- Hibernate vs. Technologie Anpassungen
- Zeitmanagement essentiell

ZEIT

- 8 Timeboxen
- ca. 11 Wochen
- ~ 600 h

ENDE

- Noch Fragen?
- Danke