

# SPORTS CLUB MANAGER

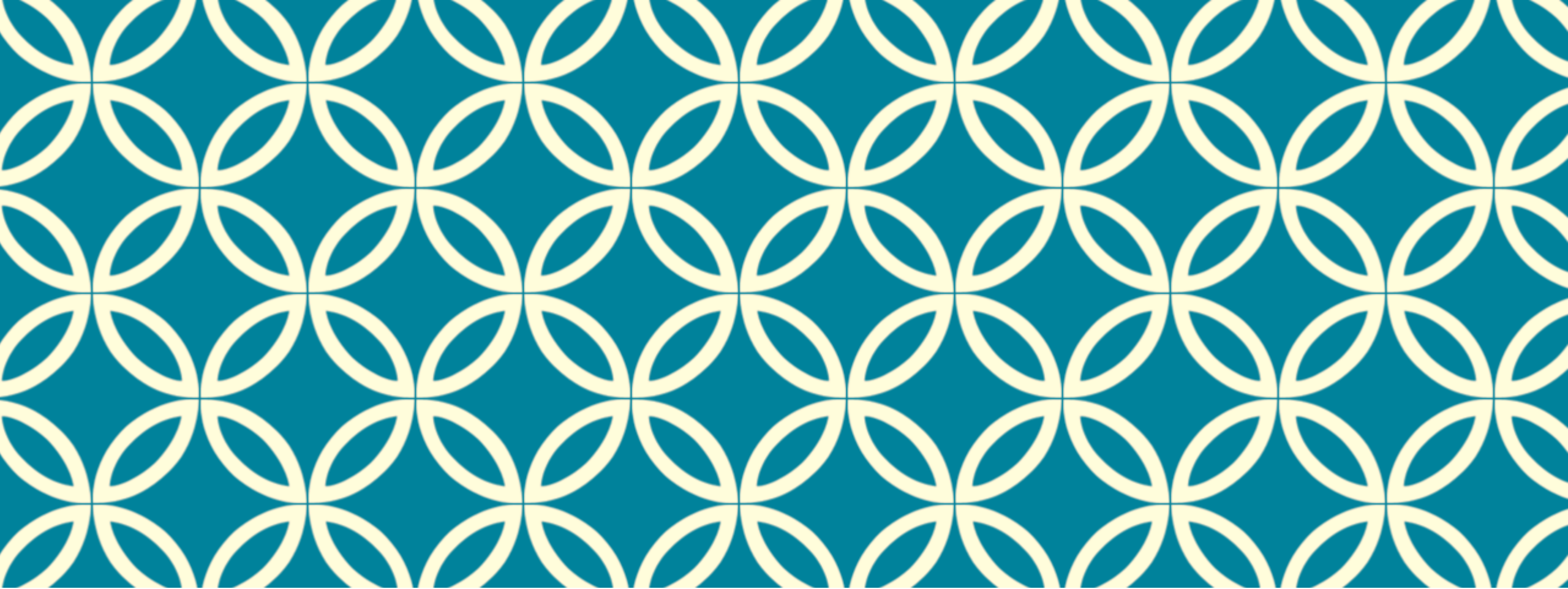
Eine wahre Geschichte

# 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

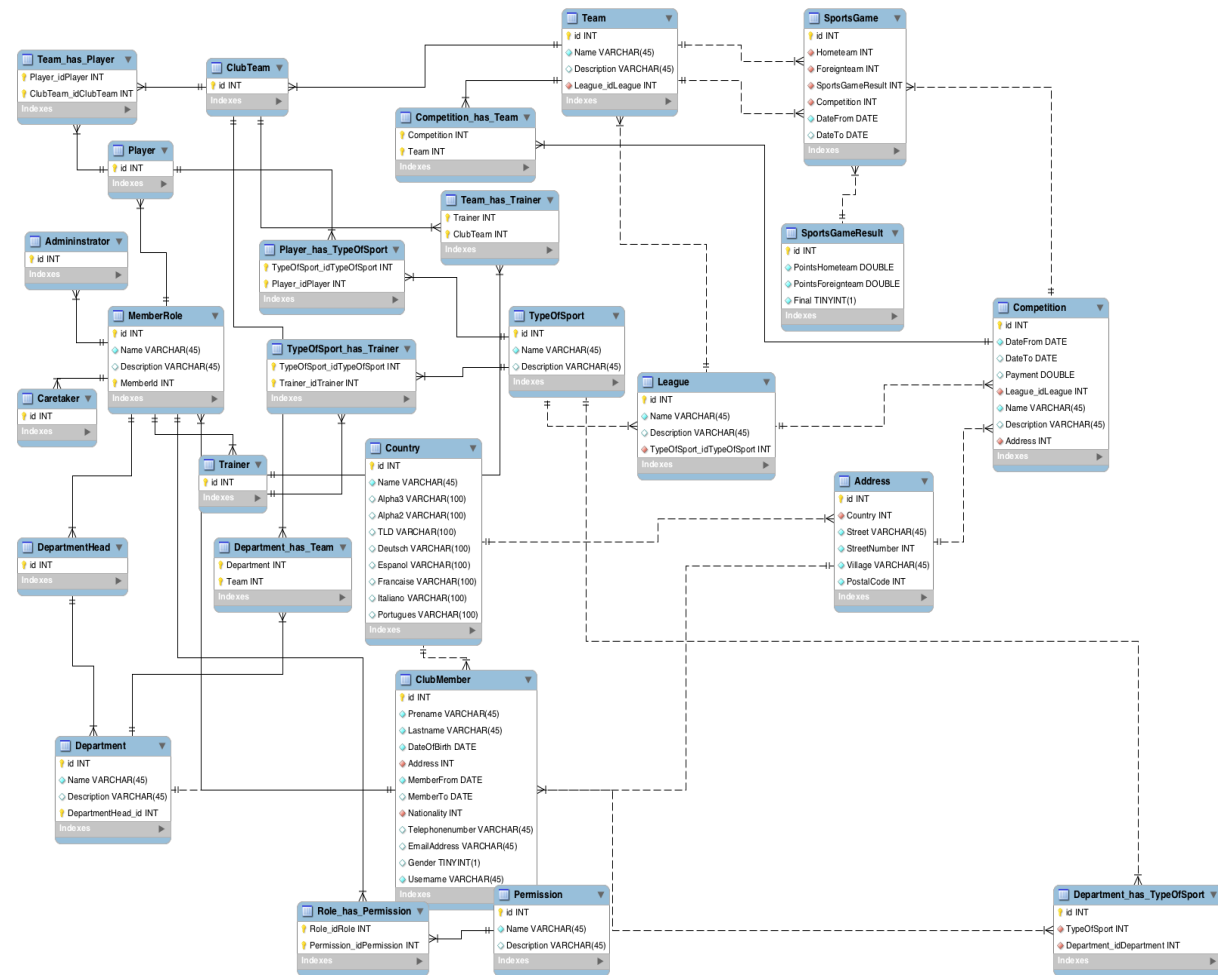
# MYSQL

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

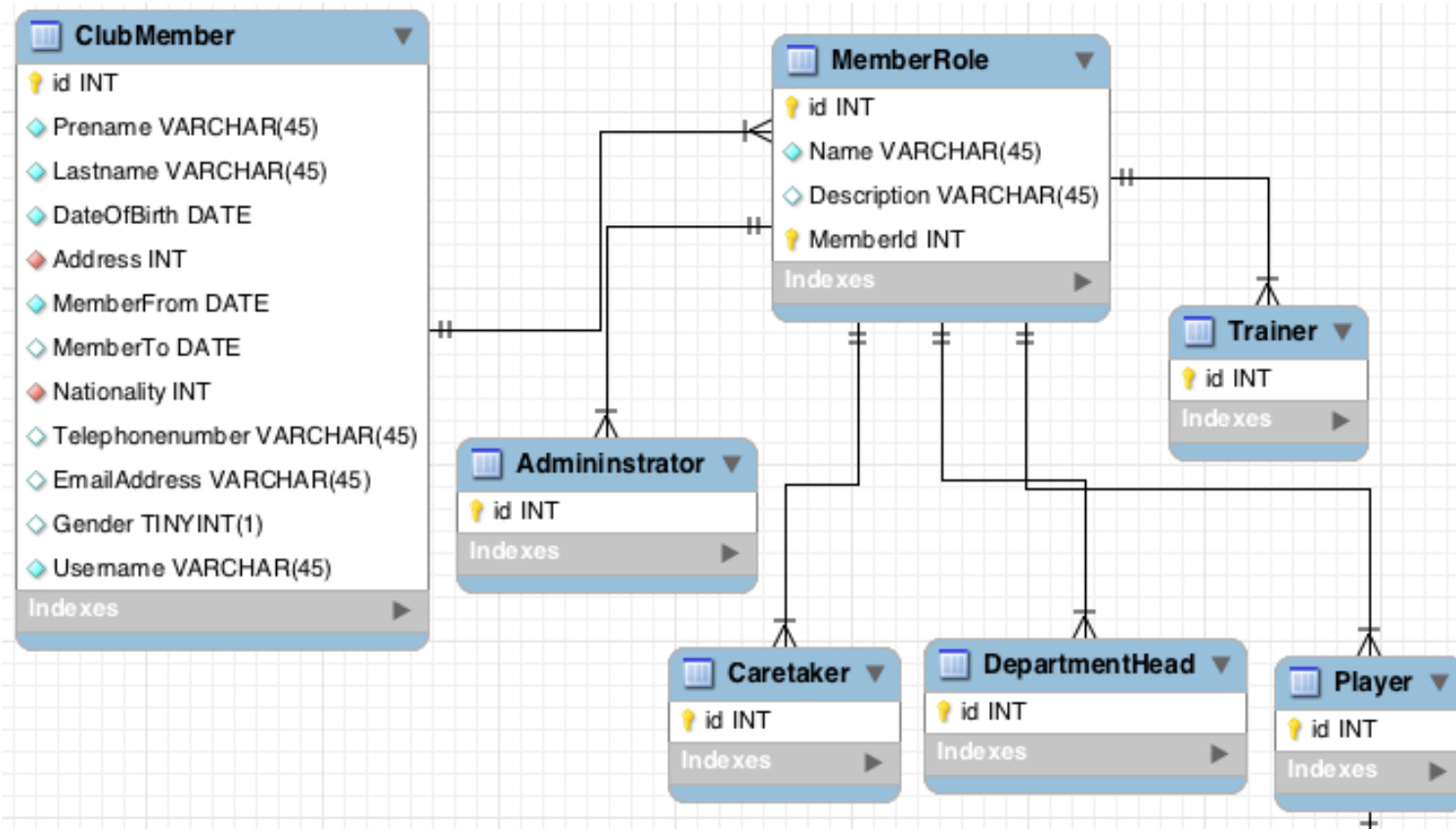
# HIBERNATE

- Freies Persistenz-Framework
- Caching
- Transaktions-Verwaltung integriert

# DATENBANKMODEL

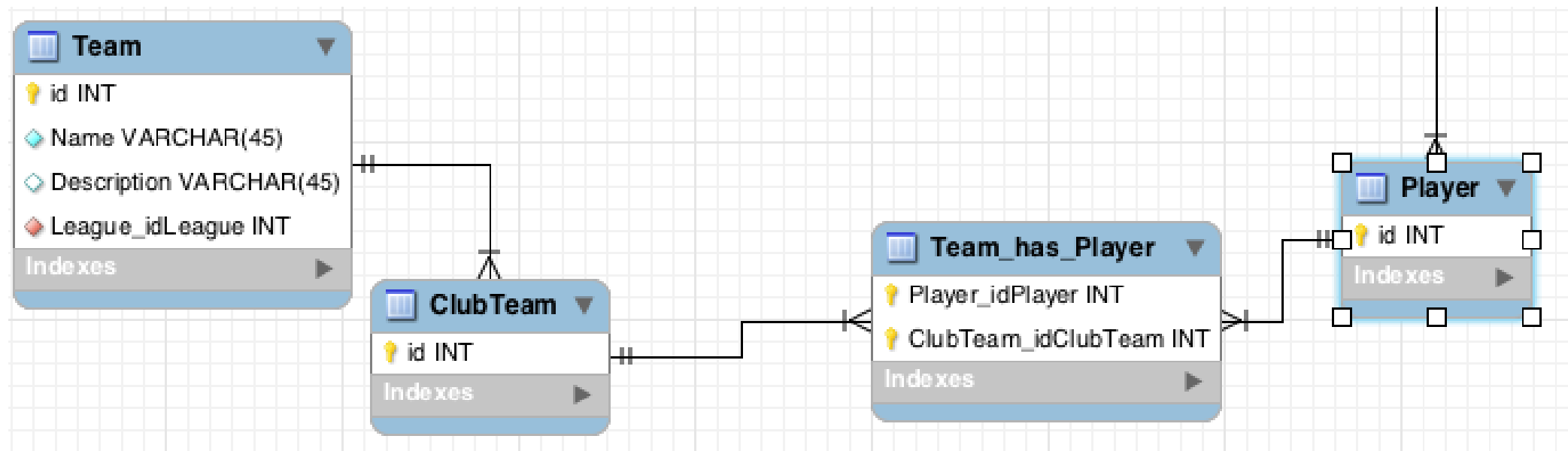


# ROLLEN

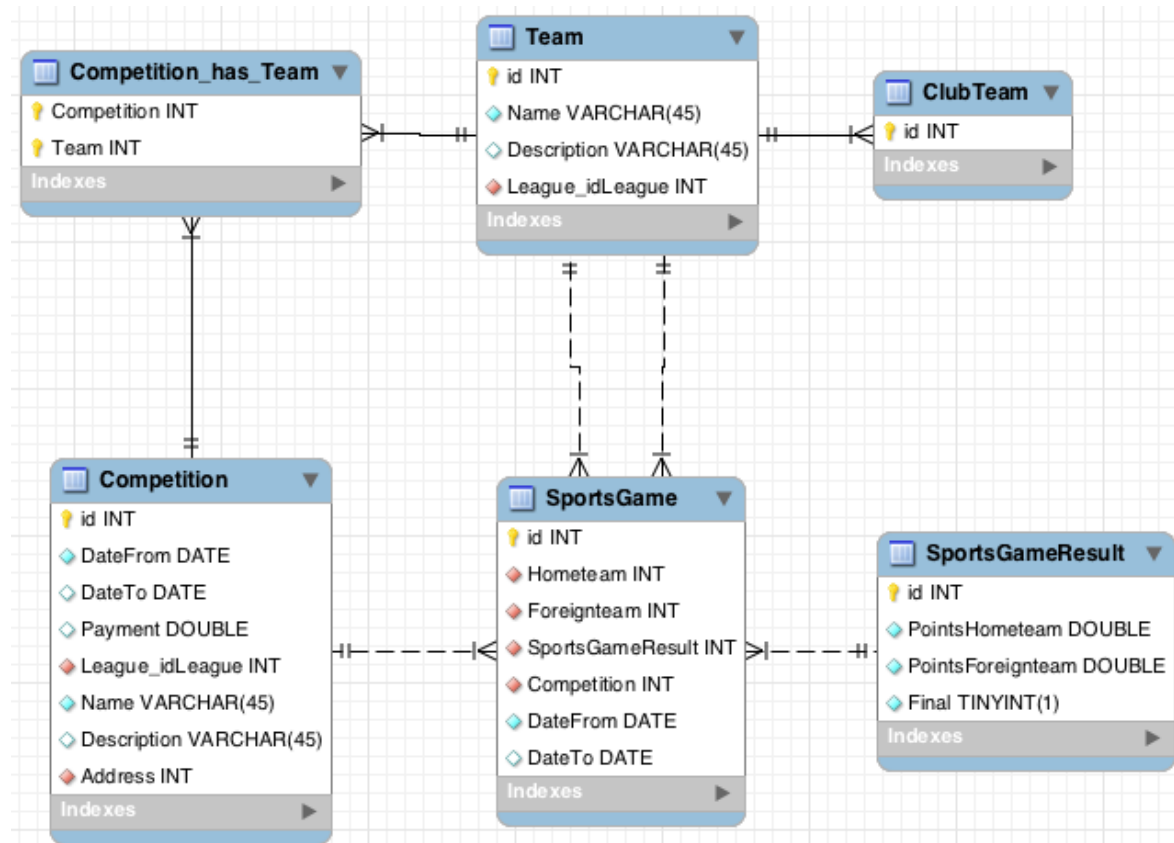




# TEAM



# WETTBEWERBE

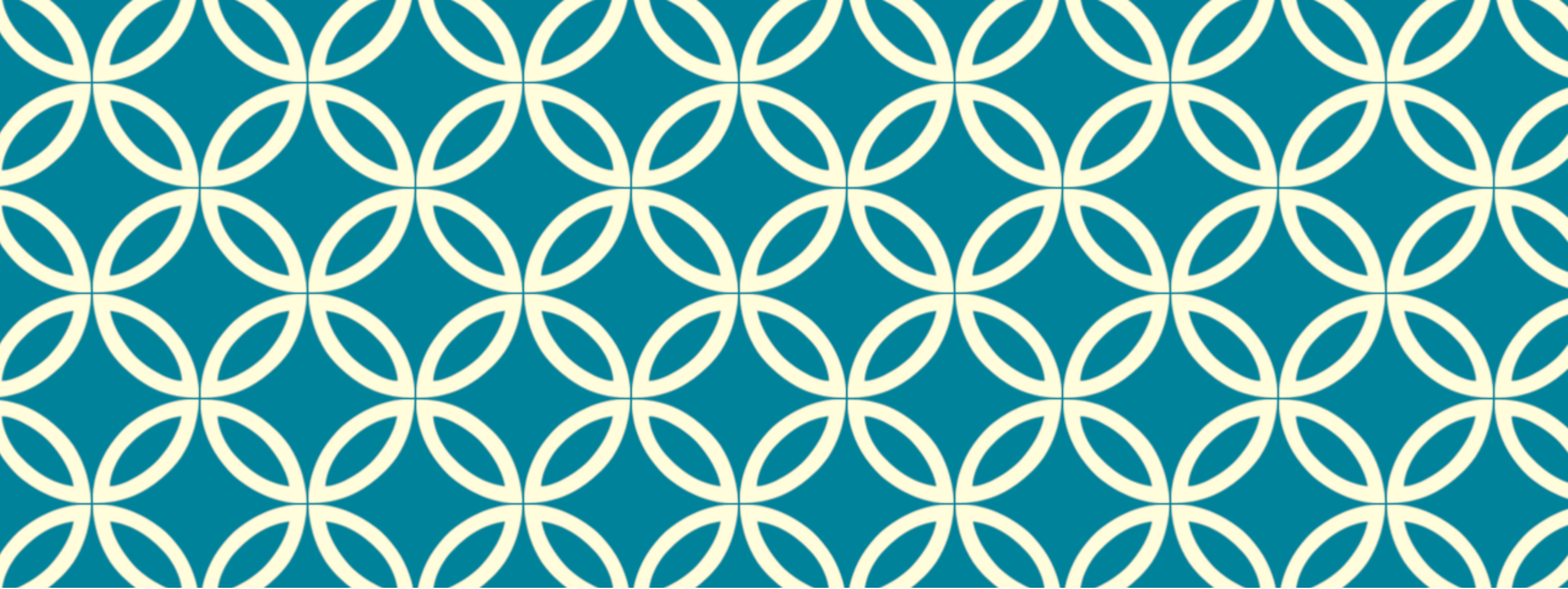


# PERSITENZ

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

# HINDERNISSE

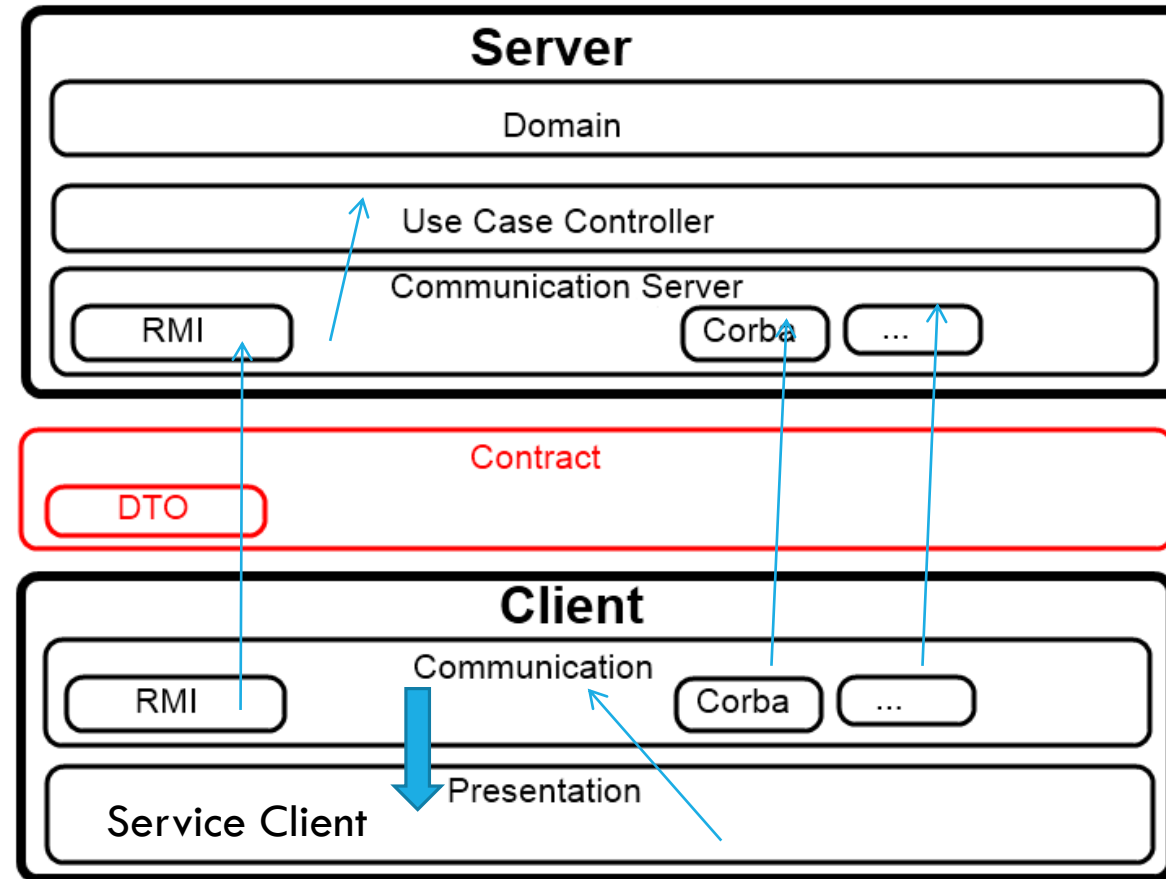
- Hibernate in Verbindung mit EJB
  - Libraries
  - Version
- Persistenzschicht auf DTOs mappen
  - Redundante Daten



**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

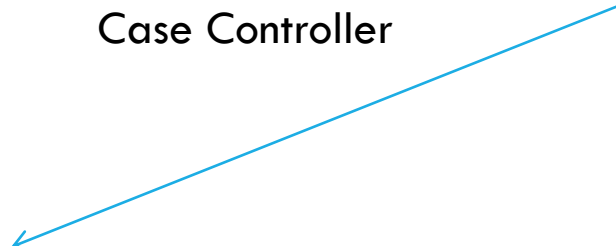
```
public List<ICompetitionDto> getCompetitionList()
```

```
throws RemoteException
```

```
{
```

```
    return AddMatchResultsController.getInstance().getCompetitionList();
```

```
}
```





# BEISPIEL — CLIENT

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

Allgemeine Use Case Controller Factories

Allgemeines Exception Handling

# BEISPIEL – CLIENT

@Override

```
public IAddMatchResultsController getAddMatchResultsController()
```

```
    throws ServiceUnavailableException
```

```
{
```

```
    try
```

```
{
```

```
        return new
```

```
AddMatchResultsServiceMapper(rmiServiceClient.getAddMatchResultsService());
```

```
...
```

Mapping zwischen RMI-Services und Client Services  
- Exception Handling



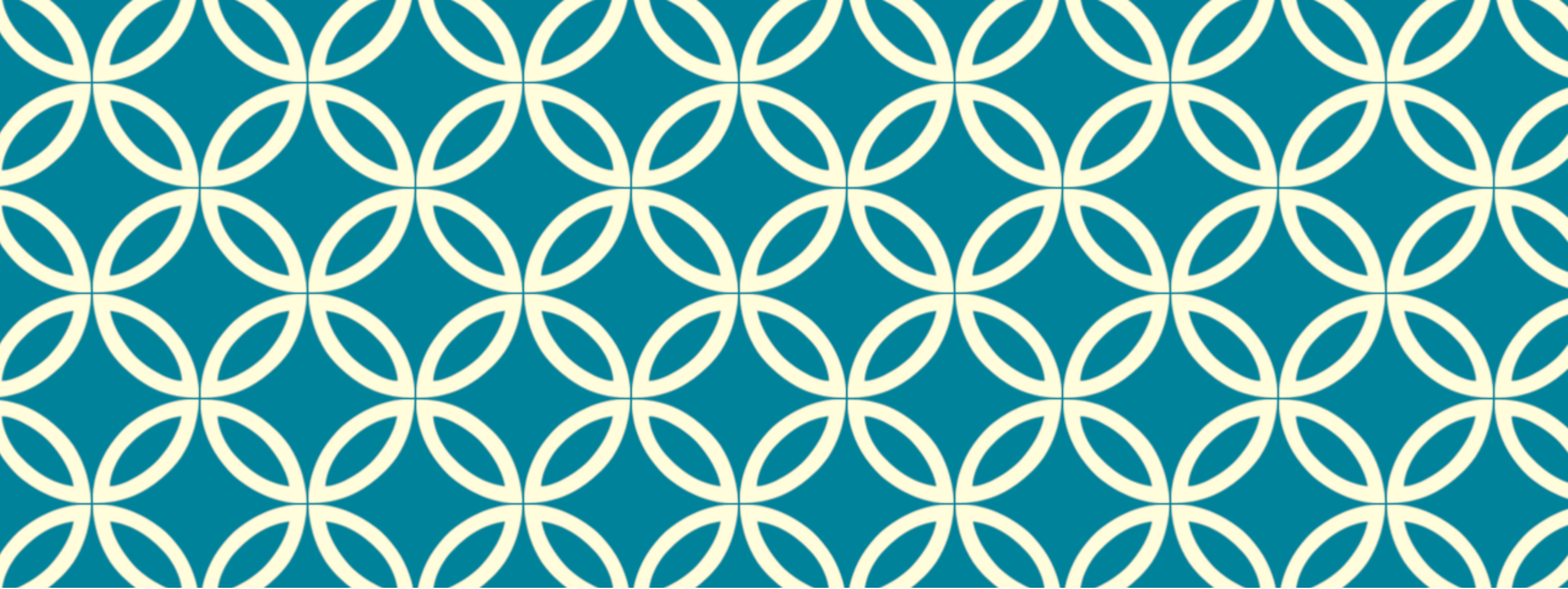
# VOR-NACHTEILE

- + 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

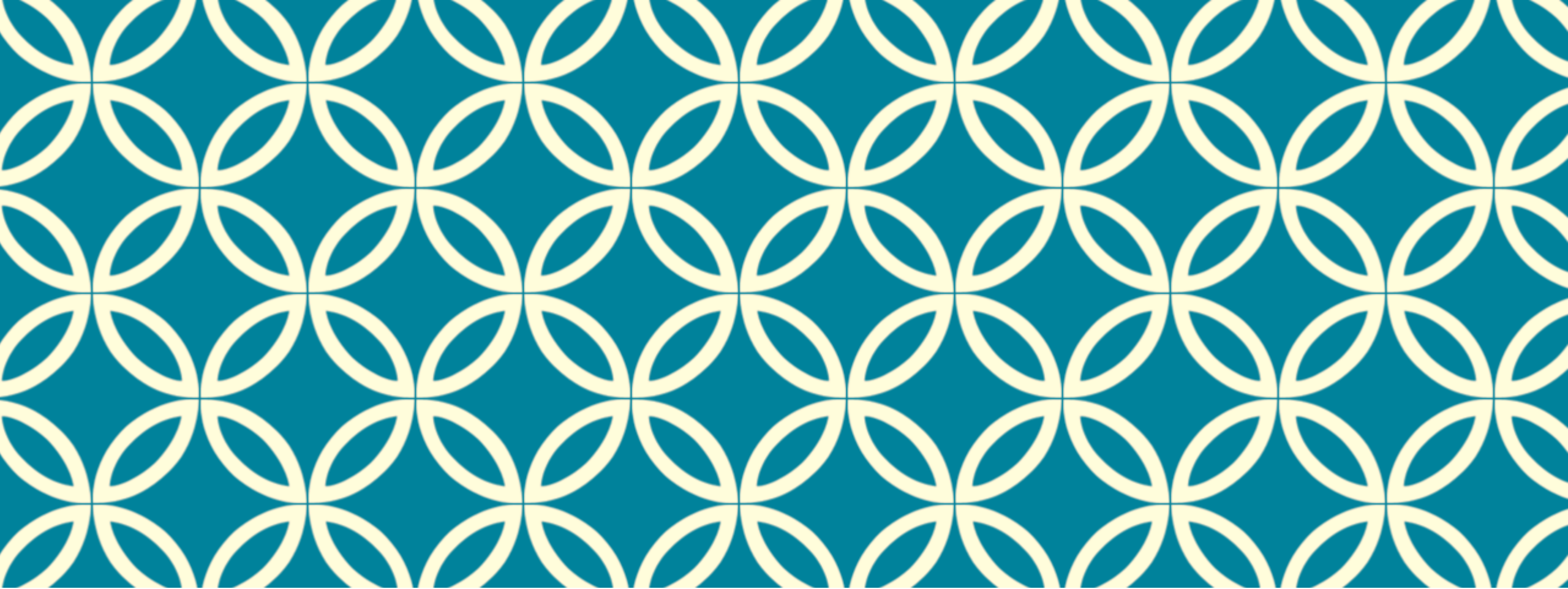
Hier grafik (muss no luga das i se schö lesbar herbring)

# REALISIERUNG

- Versucht mit Factory
- Jeden Usecase Controller einzeln
- Stateless

# FAZIT

- einfache Technik
- Veraltet
- 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

BILD

# TECHNOLOGIEN

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

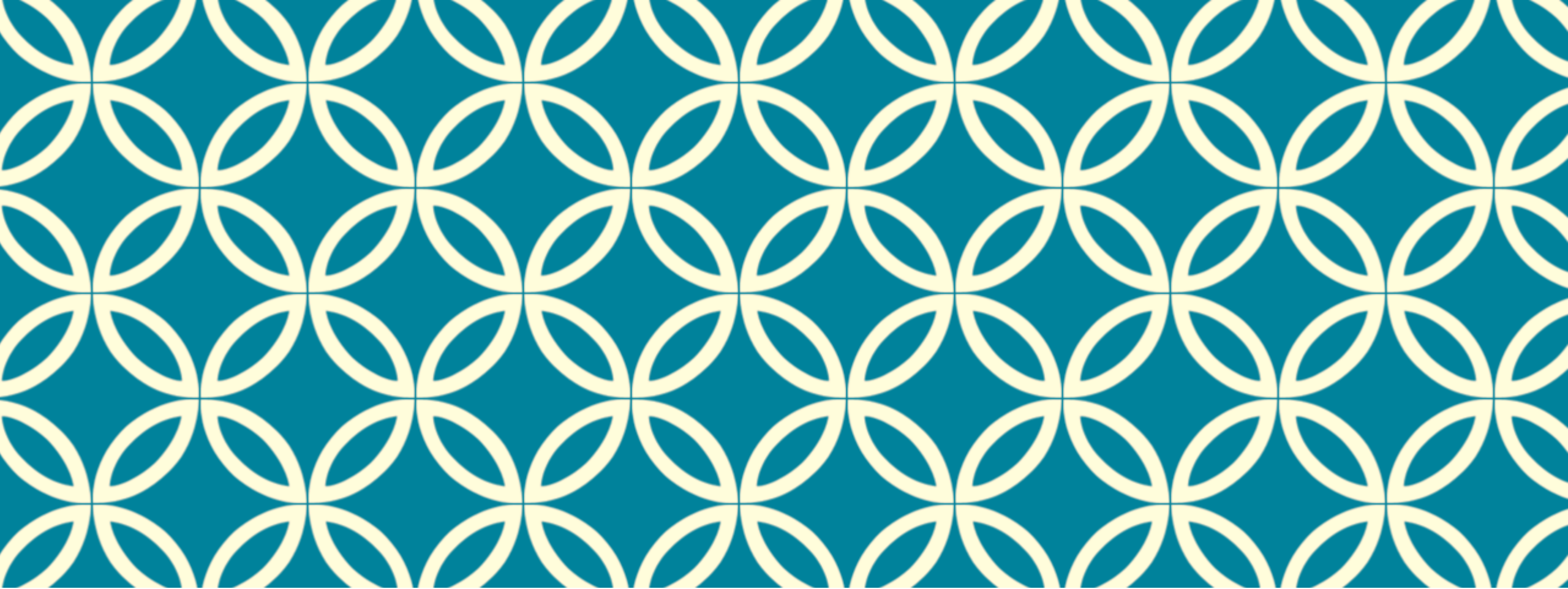


# TECHNOLOGIEN

- 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();
```

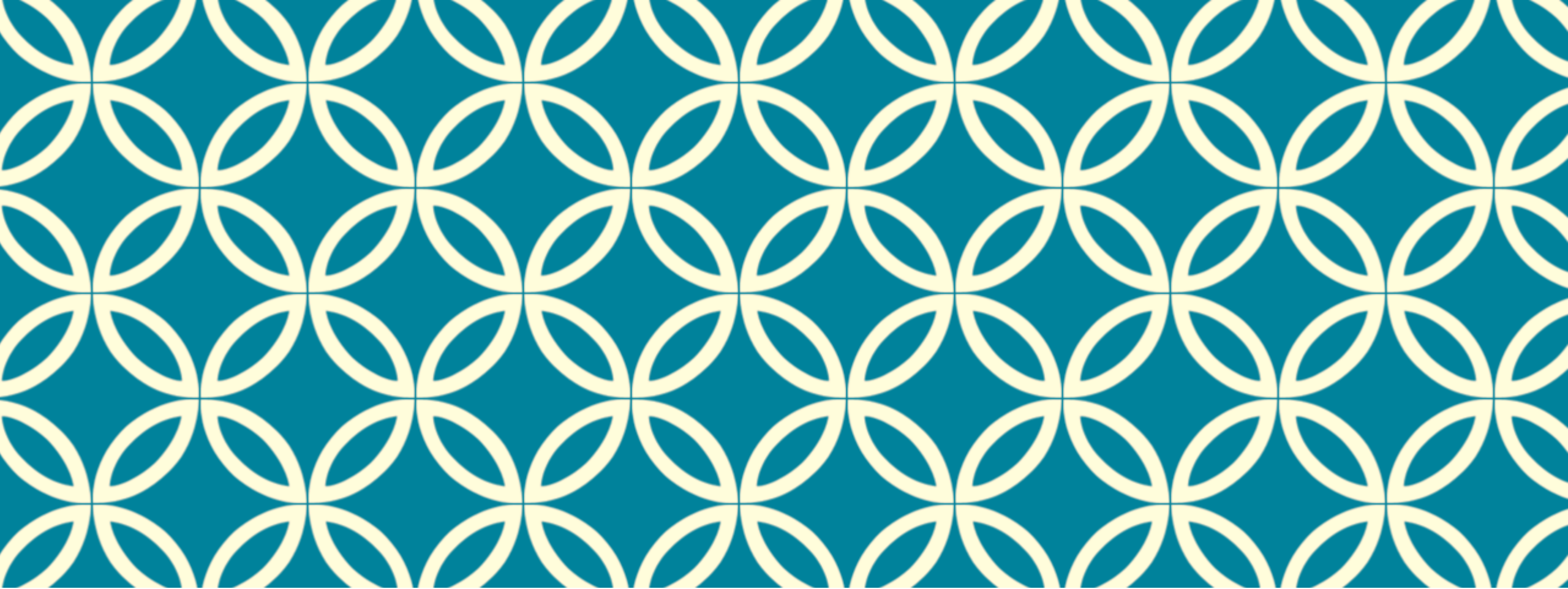
# FAZIT

## 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

Sports Club Manager

Member Competition

Search Member

th Search

Add New Member

Add To Team

Membership Nr.	First Name	Last Name	Birth Date	Gender
1	Thomas	Schwarz	1988-01-29	male
6	Thomas	Feilhauer	1965-11-06	male

First Name Thomas

Last Name Feilhauer

Mail

Phone

Birth Date 06.11.1965

Gender ☐ female ☒ male

Nationality Austria

Street + Nr. Hochschulstrasse 1

City Dornbirn

Post Code 6850

Country Austria

UserName tf-test

Entry Date 06.11.1992

Membership Nr. 6

Role ☒ Admin ☐ caretaker ☒ Department Head ☒ Trainer ☒ Player

Sport Handball Select Sport(s)

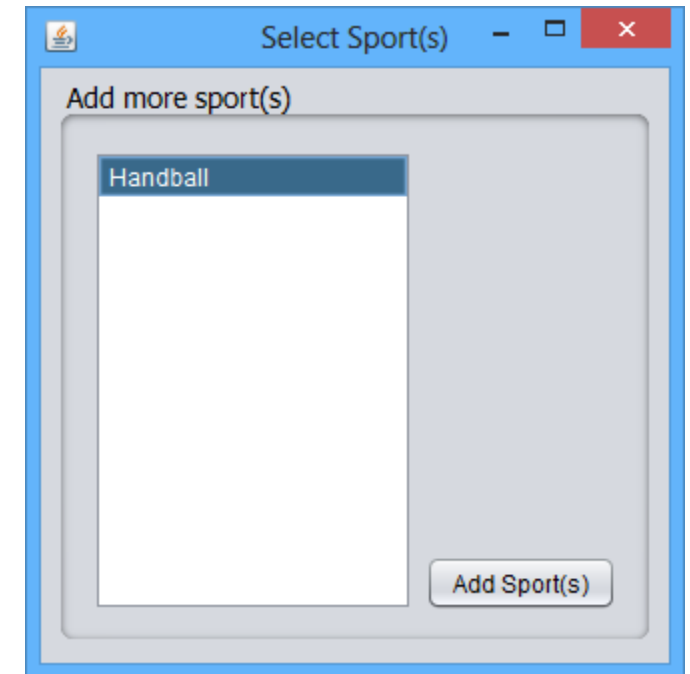
Team as Trainer Heimteam Select Team(s)

Team as Player Heimteam Select Team(s)

Apply Changes

# 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

The screenshot shows a software window titled "Sports Club Manager" with two tabs: "Member" and "Competition". The "Competition" tab is active. On the left side of the window, there is a vertical sidebar with four buttons: "Show Result", "Add Result", "Create Competition", and "Change Team".

The main area of the window is divided into two sections:

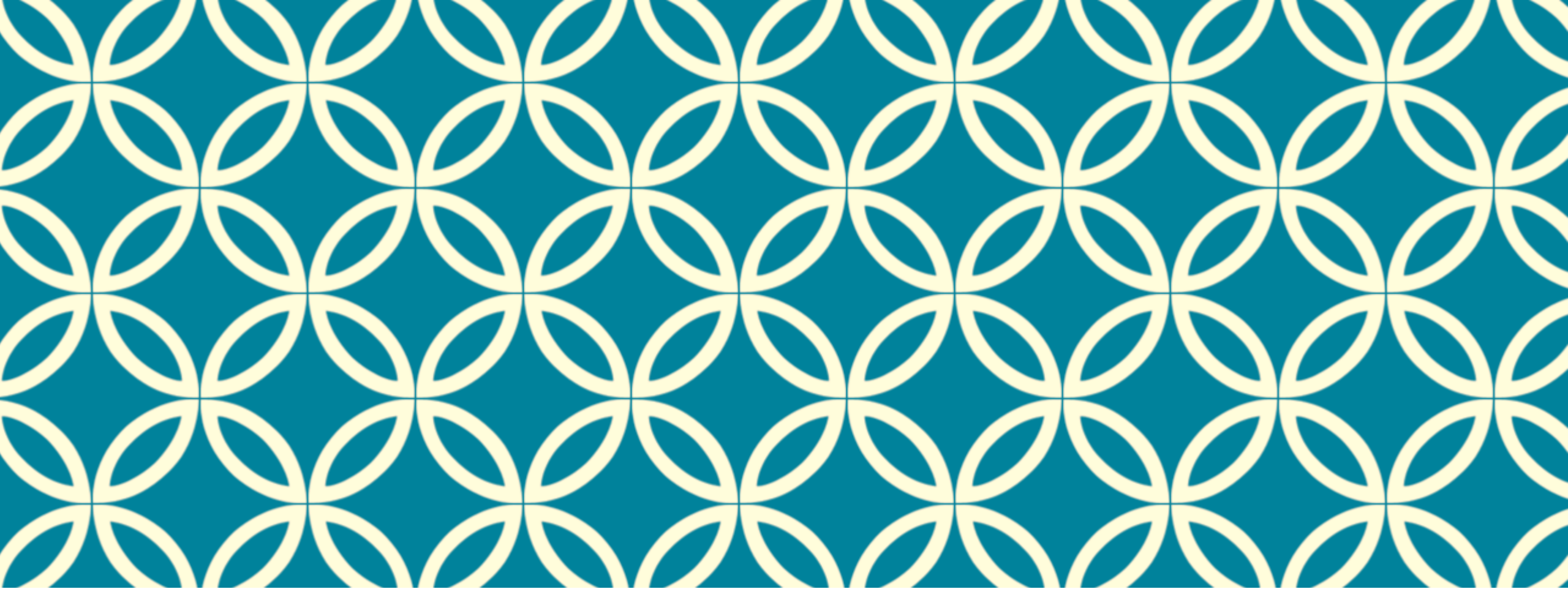
- Set Competition Details:** This section contains a form for creating a competition. It includes:
  - A "Sport" dropdown menu set to "Handball".
  - A "Select Teams" list with "Heimteam" and "Anderesteam" options.
  - A "Title" text field containing "HandballTurnier".
  - "From" and "To" date pickers set to "18.01.2013" and "20.01.2013" respectively.
  - A "Location" text field containing "FHV Dornbirn".
  - "Post Code" and "City" text fields containing "6850" and "Dornbirn".
  - A "Country" text field containing "Austria".
  - A "Fee" text field containing "5".
  - A "Confirm Data" button.
- Set Matches:** This section is for defining the match structure. It features:
  - Two team selection lists on the left, each containing "Heimteam" and "Anderesteam".
  - A "vs." label between the lists.
  - An "Add Match" button.
  - Two empty rectangular boxes for match details.
  - Another "vs." label between the boxes.
  - A "Create Competition" button at the bottom right.

# ERWEITERUNGEN

- Wöchentliche Anpassungen:
  - Mitglied zu Team hinzufügen
  - Login mit LDAP, Zugriffserlaubnis?
  - EJB
- Wettkampfergebnisse (Corba)
- Einladungen JMS
- WebClient

# TESTS & PROBLEME

- Kompatibilität
- Exception Handling 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