CrowdDJ

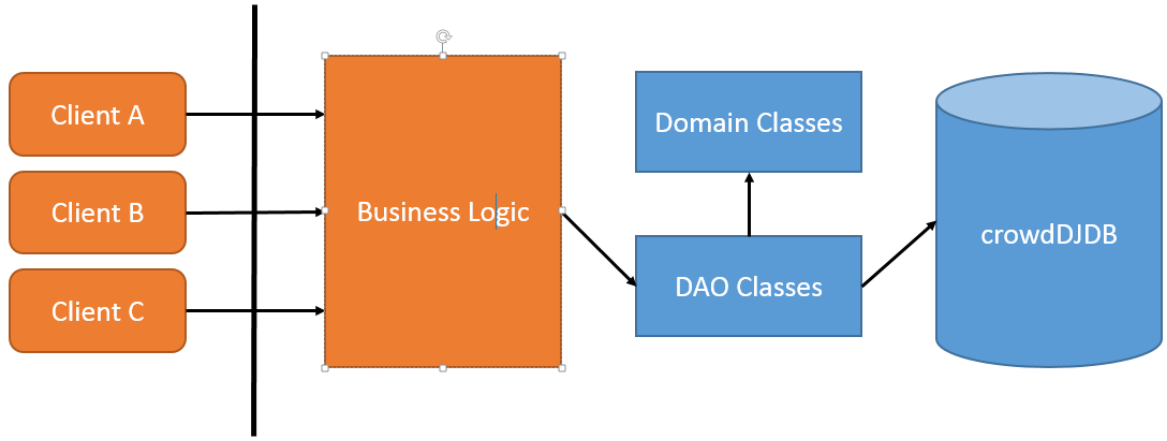
Lösungsbeschreibung

In der ersten Ausbaustufe wird sowohl die Datenbank, als auch die Datenzugriffsschicht implementiert. Hierfür wurde einfach eine „local DB“ verwendet und konfiguriert! Mittels C# wurden die Domänenklasse erstellt und Grundfunktionalitäten zwischen Datenbanken und Datenobjekten implementiert!

Systemarchitektur

Die Datenbank, sowie die „Businesslogic“-Schicht der Applikation befinden sich auf einem eigenen Server. Die Clients werden in den weiteren Ausbaustufen die Verbindung zur Logik über das Internet herstellen.

Die Logikschicht beinhaltet (neben der Programmlogik) die Domänenklassen, sowie die Datenzugriffsobjekt-Klassen, welche als einzige die Verbindung zur Datenbank herstellen können. Die Datenbank selbst beinhaltet lediglich Tabellen zur Persistierung der Daten der „CrowdDJ“ - Anwendung.



Hinweis: Blau wurde bei dieser Ausbaustufe implementiert!

Quellcode:

CrowdDJ.DAO.*

GuestDAO.cs:

```
    public class GuestDAO : IGuest
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;

        const string CmdInsert = @"INSERT INTO [dbo].[Guest] (userId, partyId) VALUES
                                   (@pUserId, @pPartyId)";
        const string CmdDelete = @"DELETE FROM [dbo].[Guest] WHERE userId = @pUserId
                                   AND partyId = @pPartyId";
        const string CmdSearch = @"SELECT * FROM [dbo].[Guest] WHERE userId = @pUserId
                                   AND partyId = @pPartyId";
        const string CmdSearchGuest = @"SELECT * FROM [dbo].[Guest] WHERE partyId =
                                   @pPartyId";
        const string CmdSearchGuestlist = @"SELECT u.userId, u.name, u.email,
                                u.isAdmin FROM [dbo].[Guest] g, [dbo].[USER] u WHERE
                                u.userId = g.userId AND g.partyId = @pPartyId";


       public bool AddGuest(Guest newGuest)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdInsert))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pUserId", newGuest.UserId));
                    cmd.Parameters.Add(new SqlParameter("@pPartyId",
                                                        newGuest.PartyId));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
        }

        public bool RemoveGuest(Guest removeGuest)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdDelete))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pUserId",
                                                       removeGuest.UserId));
                    cmd.Parameters.Add(new SqlParameter("@pPartyId",
                                                       removeGuest.PartyId));
                    if (cmd.ExecuteNonQuery() >= 1)
                        return true;
                    else
                        return false;
                }
            }
```

```
}

public bool PartyIsVisitedByGuest(int searchGuestId, int partyId)
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand cmd = new SqlCommand(CmdSearch))
        {
            connection.Open();
            cmd.Connection = connection;

            cmd.Parameters.Add(new SqlParameter("@pUserId", searchGuestId));
            cmd.Parameters.Add(new SqlParameter("@pPartyId", partyId));

            if (cmd.ExecuteNonQuery() == 1)
                return true;
            else
                return false;
        }
    }
}



public List<User> GetGuestlistForParty(int partyId)
{
    List<User> result = new List<User>();
    User user = null;
    int rUserId = 0;
    string rName = "";
    string rEmail = "";
    bool rIsAdmin = false;

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand cmd = new SqlCommand(CmdSearchGuestlist))
        {
            connection.Open();
            cmd.Connection = connection;
            cmd.Parameters.Add(new SqlParameter("@pPartyId", partyId));

            using (SqlDataReader rDr = cmd.ExecuteReader())
            {
                while (rDr.Read())
                {
                    if (!rDr.IsDBNull(0))
                    {
                        rUserId = rDr.GetInt32(0);
                        rName = rDr.GetString(1);
                        rEmail = rDr.GetString(2);
                        rIsAdmin = rDr.GetBoolean(3);
                        user = new User(rName, "", rEmail, rIsAdmin);
                        user.UserId = rUserId;
                    }
                    else
                    {
                        user = new User("", "", "", false);
                    }
                    result.Add(user);
                }
            }
        }
```

```
            }
            return result;
        }
    }
```

PartyDAO.cs

```
    public class PartyDAO : IParty
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;

        const string CmdInsert = @"INSERT INTO [dbo].[Party] (name, location, host,
                                    partyBegin, partyEnd, isActive)
                                     VALUES (@pName, @pLocation, @pHost, @pBegin,
                                    @pEnd, @pIsActive)";
        const string CmdDelete = @"DELETE FROM [dbo].[Party] WHERE partyId =
                                                                    @pPartyId";
        const string CmdSearch = @"SELECT * FROM [dbo].[Party] WHERE partyId =
                                                                    @pPartyId";
        const string CmdSearchHost = @"SELECT * FROM [dbo].[Party] WHERE host =
                                                                    @pHost";
        const string CmdSelectAll = @"SELECT * FROM [dbo].[Party]";

        public bool AddParty(Party newParty)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdInsert))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pName", newParty.Name));
                    cmd.Parameters.Add(new SqlParameter("@pLocation",
                                                        newParty.Location));
                    cmd.Parameters.Add(new SqlParameter("@pHost", newParty.Host));
                    cmd.Parameters.Add(new SqlParameter("@pBegin",
                                                        newParty.PartyBegin));
                    cmd.Parameters.Add(new SqlParameter("@pEnd", newParty.PartyEnd));
                    cmd.Parameters.Add(new SqlParameter("@pIsActive",
                                                        newParty.IsActive));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
        }

        public bool RemovePartyWithId(int partyId)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdDelete))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pPartyId", partyId));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
```

```
        }

        public Party FindPartyById(int partyId)
        {
            Party party= null;
            int rPartyId = 0;
            string rName = "";
            string rLocation = "";
            string rHost = "";
            string rBegin;
            string rEnd;
            bool rIsActive = false;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdSearch))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pPartyId", partyId));

                    using (SqlDataReader rDr = cmd.ExecuteReader())
                    {
                        while (rDr.Read())
                        {
                            if (!rDr.IsDBNull(0))
                            {
                                rPartyId = rDr.GetInt32(0);
                                rName = rDr.GetString(1);
                                rLocation = rDr.GetString(2);
                                rHost = rDr.GetString(3);
                                rBegin = rDr.GetString(4);
                                rEnd = rDr.GetString(5);
                                rIsActive = rDr.GetBoolean(6);
                                party = new Party(rName, rLocation, rHost, rBegin,
                                                  rEnd, rIsActive);
                                party.PartyId = rPartyId;
                            }
                            else
                            {
                                party = new Party("", "", "", "", "", false);
                            }
                        }
                    }
                }
            }
            return party;
        }


        public List<Party> FindPartyWithHost(string hostName)
        {
            List<Party> result = new List<Party>();
            Party party = null;
            int rPartyId = 0;
            string rName = "";
            string rLocation = "";
            string rHost = "";
            string rBegin;
            string rEnd;
            bool rIsActive = false;
```

```
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdSearchHost))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pHost", hostName));

                    using (SqlDataReader rDr = cmd.ExecuteReader())
                    {
                        while (rDr.Read())
                        {
                            if (!rDr.IsDBNull(0))
                            {
                                rPartyId = rDr.GetInt32(0);
                                rName = rDr.GetString(1);
                                rLocation = rDr.GetString(2);
                                rHost = rDr.GetString(3);
                                rBegin = rDr.GetString(4);
                                rEnd = rDr.GetString(5);
                                rIsActive = rDr.GetBoolean(6);
                                party = new Party(rName, rLocation, rHost, rBegin,
                                                                    rEnd, rIsActive);
                                party.PartyId = rPartyId;
                            }
                            else
                            {
                                party = new Party("", "", "", "", "", false);
                            }
                            result.Add(party);
                        }
                    }
                }
            }
            return result;
        }

        public List<Party> GetAllParties()
        {
            List<Party> result = new List<Party>();
            Party party = null;
            int rPartyId = 0;
            string rName = "";
            string rLocation = "";
            string rHost = "";
            string rBegin;
            string rEnd;
            bool rIsActive = false;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdSelectAll))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    using (SqlDataReader rDr = cmd.ExecuteReader())
                    {
                        while (rDr.Read())
                        {
                            if (!rDr.IsDBNull(0))
                            {
```

```
                                rPartyId = rDr.GetInt32(0);
                                rName = rDr.GetString(1);
                                rLocation = rDr.GetString(2);
                                rHost = rDr.GetString(3);
                                rBegin = rDr.GetString(4);
                                rEnd = rDr.GetString(5);
                                rIsActive = rDr.GetBoolean(6);
                                party = new Party(rName, rLocation, rHost, rBegin,
                                                        rEnd, rIsActive);
                                party.PartyId = rPartyId;
                            }
                            else
                            {
                                party = new Party("", "", "", "", "", false);
                            }
                            result.Add(party);
                        }
                    }
                }
            }
            return result;
        }
    }
```

PartytweetDAO.cs

```
    public class PartytweetDAO : IPartytweet
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;

        const string CmdInsert = @"INSERT INTO [dbo].[Partytweet] (userId, partyId,
                                    message) VALUES (@pUserId, @pPartyId, @pMessage)";
        const string CmdGetTweetsForParty = @"SELECT * FROM [dbo].[Partytweet] WHERE
                                            partyId = @pPartyId";
        const string CmdGetAllTweets = @"SELECT * FROM [dbo].[Partytweet]";

        public bool AddTweet(Partytweet newTweet)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdInsert))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pUserId", newTweet.UserId));
                    cmd.Parameters.Add(new SqlParameter("@pPartyId",
                                                        newTweet.PartyId));
                    cmd.Parameters.Add(new SqlParameter("@pMessage",
                                                        newTweet.Message));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
        }
```

```
public List<Partytweet> GetTweetsForParty(int partyId)
 {
     List<Partytweet> result = new List<Partytweet>();
     Partytweet partytweet = null;
     int rUserId = 0;
     int rPartyId = 0;
     string rMessage = "";

     using (SqlConnection connection = new SqlConnection(connectionString))
     {
         using (SqlCommand cmd = new SqlCommand(CmdGetTweetsForParty))
         {
             connection.Open();
             cmd.Connection = connection;
             cmd.Parameters.Add(new SqlParameter("@pPartyId", partyId));

             using (SqlDataReader rDr = cmd.ExecuteReader())
             {
                 while (rDr.Read())
                 {
                     if (!rDr.IsDBNull(0))
                     {
                         rUserId = rDr.GetInt32(0);
                         rPartyId = rDr.GetInt32(1);
                         rMessage = rDr.GetString(2);
                         partytweet = new Partytweet(rUserId, rPartyId,
                                                     rMessage);
                     }
                     else
                     {
                         partytweet = new Partytweet(-1, -1, "");
                     }
                     result.Add(partytweet);
                 }
             }
         }
     }
     return result;
 }

 public List<Partytweet> GetAllTweets()
 {
     List<Partytweet> result = new List<Partytweet>();
     Partytweet partytweet = null;
     int rUserId = 0;
     int rPartyId = 0;
     string rMessage = "";

     using (SqlConnection connection = new SqlConnection(connectionString))
     {
         using (SqlCommand cmd = new SqlCommand(CmdGetAllTweets))
         {
             connection.Open();
             cmd.Connection = connection;

             using (SqlDataReader rDr = cmd.ExecuteReader())
             {
                 while (rDr.Read())
                 {
                     if (!rDr.IsDBNull(0))
                     {
                         rUserId = rDr.GetInt32(0);
                         rPartyId = rDr.GetInt32(1);
```

```
                            rMessage = rDr.GetString(2);
                            partytweet = new Partytweet(rUserId, rPartyId,
                                                        rMessage);
                        }
                        else
                        {
                            partytweet = new Partytweet(-1, -1, "");
                        }
                        result.Add(partytweet);
                    }
                }
            }
        }
        return result;
    }
}
```

PlaylistDAO.cs

```
    public class PlaylistDAO : IPlaylist
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;

        const string CmdAddPlaylist = @"INSERT INTO [dbo].[Playlist] (playlistId,
                                        name) VALUES (@pPlaylistId, @pName)";
        const string CmdGetPlaylistForParty = @"SELECT * FROM [dbo].[Playlist] WHERE
                                          partyId = @pPartyId";
        const string CmdGetAllTracksInPlaylist = @"SELECT t.trackId, t.title,
                                 t.artist, t.url, t.length, t.genre, t.isVideo
                                FROM [dbo].[Playlist] pl,
                                [dbo].[Tracklist] tl, [dbo].[Track] t
                                WHERE pl.playlistId = @pPlaylistId AND
                                pl.playlistId = tl.playlistId
                                AND tl.trackId = t.trackId";
        const string CmdGetAllPlaylists = @"SELECT * FROM [dbo].[Playlist]";

        public bool AddPlaylist(Playlist newPlaylist)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdAddPlaylist))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pPlaylistId",
                                                       newPlaylist.PlaylistId));
                    cmd.Parameters.Add(new SqlParameter("@pName", newPlaylist.Name));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
        }

        public Playlist GetPlaylistForParty(int id)
        {
            Playlist result = null;
            int rPlaylistId = 0;
```

```
            string rName = "";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdGetPlaylistForParty))
                {
                    connection.Open();
                    cmd.Connection = connection;
                    cmd.Parameters.Add(new SqlParameter("@pPartyId", id));
                    using (SqlDataReader rDr = cmd.ExecuteReader())
                    {

                        while (rDr.Read())
                        {
                            if (!rDr.IsDBNull(0))
                            {
                                rPlaylistId = rDr.GetInt32(0);
                                rName = rDr.GetString(1);
                                result = new Playlist(rPlaylistId, rName);
                            }
                            else
                            {
                                result = new Playlist(-1, "");
                            }
                        }
                    }
                }
            }
            return result;
        }

        public List<Track> GetAllTracksInPlaylist(int playlistId)
        {
            List<Track> result = new List<Track>();
            Track track = null;
            int rTrackId = 0;
            string rTitle = "";
            string rArtist = "";
            string rUrl = "";
            int rLength = 0;
            string rGenre = "";
            bool rIsVideo = false;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdGetAllTracksInPlaylist))
                {
                    connection.Open();
                    cmd.Connection = connection;
                    cmd.Parameters.Add(new SqlParameter("@pPlaylistId", playlistId));
                    using (SqlDataReader rDr = cmd.ExecuteReader())
                    {
                        while (rDr.Read())
                        {
                            if (!rDr.IsDBNull(0))
                            {
                                rTrackId = rDr.GetInt32(0);
                                rTitle = rDr.GetString(1);
                                rArtist = rDr.GetString(2);
                                rUrl = rDr.GetString(3);
                                rLength = rDr.GetInt32(4);
                                rGenre = rDr.GetString(5);
                                rIsVideo = rDr.GetBoolean(6);
```

```
                            track = new Track(rTitle, rArtist, rUrl, rLength,
                                              rGenre, rIsVideo);
                            track.TrackId = rTrackId;
                        }
                        else
                        {
                            track = new Track("", "", "", 0, "", false);
                        }
                        result.Add(track);
                    }
                }
            }
        }

        return result;
    }

    public List<Playlist> GetAllPlaylists()
    {
        List<Playlist> result = new List<Playlist>();
        Playlist playlist = null;
        int rPlaylist = 0;
        string rName = "";

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            using (SqlCommand cmd = new SqlCommand(CmdGetAllPlaylists))
            {
                connection.Open();
                cmd.Connection = connection;
                SqlDataReader rDr = cmd.ExecuteReader();

                while (rDr.Read())
                {
                    if (!rDr.IsDBNull(0))
                    {
                        rPlaylist = rDr.GetInt32(0);
                        rName = rDr.GetString(1);
                        playlist = new Playlist(rPlaylist, rName);
                    }
                    else
                    {
                        playlist = new Playlist(-1, "");
                    }
                    result.Add(playlist);
                }
            }
        }

        return result;
    }
}
```

TrackDAO.cs

```
    public class TrackDAO : ITrack
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;

        const string CmdInsert = @"INSERT INTO [dbo].[Track] (title, artist, url,
                                                              length, genre, isVideo)
                                                VALUES (@pTitle, @pArtist, @pUrl, @pLength,
                                                          @pGenre, @pIsVideo)";
        const string CmdDelete = @"DELETE FROM [dbo].[Track] WHERE trackId =
                                                                @pTrackId";
        const string CmdSearchTitle = @"SELECT * FROM [dbo].[Track] WHERE isVideo =
                                                                false";
        const string CmdSearchSongs = @"SELECT * FROM [dbo].[Track] WHERE isVideo =
                                                                false";
        const string CmdSearchVideos = @"SELECT * FROM [dbo].[Track] WHERE isVideo =
                                                                true";
        const string CmdSearchGenre = @"SELECT * FROM [dbo].[Track] WHERE genre =
                                                                @pGenre";
        const string CmdSelectAll = @"SELECT * FROM [dbo].[Track]";

        public bool AddTrack(Track newTrack)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdInsert))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pTitle", newTrack.Title));
                    cmd.Parameters.Add(new SqlParameter("@pArtist", newTrack.Artist));
                    cmd.Parameters.Add(new SqlParameter("@pUrl", newTrack.Url));
                    cmd.Parameters.Add(new SqlParameter("@pLength", newTrack.Length));
                    cmd.Parameters.Add(new SqlParameter("@pGenre", newTrack.Genre));
                    cmd.Parameters.Add(new SqlParameter("@pIsVideo",
                                                        newTrack.IsVideo));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
        }

        public bool RemoveTrackWithId(int id)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdDelete))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pTrackId", id));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
```

```
            }
        }
    }

    public List<Track> FindTrackWithTitle(string title)
    {
        List<Track> result = new List<Track>();
        Track track = null;
        int rTrackId = 0;
        string rTitle = "";
        string rArtist = "";
        string rUrl = "";
        int rLength = 0;
        string rGenre = "";
        bool rIsVideo = false;

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            using (SqlCommand cmd = new SqlCommand(CmdSearchTitle))
            {
                connection.Open();
                cmd.Connection = connection;
                cmd.Parameters.Add(new SqlParameter("@pTitle", title));
                SqlDataReader rDr = cmd.ExecuteReader();

                while (rDr.Read())
                {
                    if (!rDr.IsDBNull(0))
                    {
                        rTrackId = rDr.GetInt32(0);
                        rTitle = rDr.GetString(1);
                        rArtist = rDr.GetString(2);
                        rUrl = rDr.GetString(3);
                        rLength = rDr.GetInt32(4);
                        rGenre = rDr.GetString(5);
                        rIsVideo = rDr.GetBoolean(6);
                        track = new Track(rTitle, rArtist, rUrl, rLength, rGenre,
                                          rIsVideo);
                        track.TrackId = rTrackId;
                    }
                    else
                    {
                        track = new Track("", "", "", 0, "", false);
                    }
                    result.Add(track);
                }
            }
        }

        return result;
    }

    public List<Track> FindTracksInGenre(string genre)
    {
        List<Track> result = new List<Track>();
        Track track = null;
        int rTrackId = 0;
        string rTitle = "";
        string rArtist = "";
        string rUrl = "";
        int rLength = 0;
        string rGenre = "";
        bool rIsVideo = false;
```

```
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdSearchGenre))
                {
                    connection.Open();
                    cmd.Connection = connection;
                    cmd.Parameters.Add(new SqlParameter("@pGenre", genre));
                    SqlDataReader rDr = cmd.ExecuteReader();

                    while (rDr.Read())
                    {
                        if (!rDr.IsDBNull(0))
                        {
                            rTrackId = rDr.GetInt32(0);
                            rTitle = rDr.GetString(1);
                            rArtist = rDr.GetString(2);
                            rUrl = rDr.GetString(3);
                            rLength = rDr.GetInt32(4);
                            rGenre = rDr.GetString(5);
                            rIsVideo = rDr.GetBoolean(6);
                            track = new Track(rTitle, rArtist, rUrl, rLength, rGenre,
                                              rIsVideo);
                            track.TrackId = rTrackId;
                        }
                        else
                        {
                            track = new Track("", "", "", 0, "", false);
                        }
                        result.Add(track);
                    }
                }
            }

            return result;
        }

        public List<Track> FindVideos()
        {
            List<Track> result = new List<Track>();
            Track track = null;
            int rTrackId = 0;
            string rTitle = "";
            string rArtist = "";
            string rUrl = "";
            int rLength = 0;
            string rGenre = "";
            bool rIsVideo = false;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdSearchVideos))
                {
                    connection.Open();
                    cmd.Connection = connection;
                    SqlDataReader rDr = cmd.ExecuteReader();

                    while (rDr.Read())
                    {
                        if (!rDr.IsDBNull(0))
                        {
                            rTrackId = rDr.GetInt32(0);
                            rTitle = rDr.GetString(1);
```

```
                    rArtist = rDr.GetString(2);
                    rUrl = rDr.GetString(3);
                    rLength = rDr.GetInt32(4);
                    rGenre = rDr.GetString(5);
                    rIsVideo = rDr.GetBoolean(6);
                    track = new Track(rTitle, rArtist, rUrl, rLength, rGenre,
                                      rIsVideo);
                    track.TrackId = rTrackId;
                }
                else
                {
                    track = new Track("", "", "", 0, "", false);
                }
                result.Add(track);
            }
        }
    }

    return result;
}

public List<Track> FindSongs()
{
    List<Track> result = new List<Track>();
    Track track = null;
    int rTrackId = 0;
    string rTitle = "";
    string rArtist = "";
    string rUrl = "";
    int rLength = 0;
    string rGenre = "";
    bool rIsVideo = false;

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand cmd = new SqlCommand(CmdSearchSongs))
        {
            connection.Open();
            cmd.Connection = connection;
            SqlDataReader rDr = cmd.ExecuteReader();

            while (rDr.Read())
            {
                if (!rDr.IsDBNull(0))
                {
                    rTrackId = rDr.GetInt32(0);
                    rTitle = rDr.GetString(1);
                    rArtist = rDr.GetString(2);
                    rUrl = rDr.GetString(3);
                    rLength = rDr.GetInt32(4);
                    rGenre = rDr.GetString(5);
                    rIsVideo = rDr.GetBoolean(6);

                   track = new Track(rTitle, rArtist, rUrl, rLength, rGenre,
                          rIsVideo);
                    track.TrackId = rTrackId;
                }
                else
                {
                    track = new Track("", "", "", 0, "", false);
                }
                result.Add(track);
            }
```

```
                }
            }

            return result;
        }

        public List<Track> GetAllTracks()
        {
            List<Track> result = new List<Track>();
            Track track = null;
            int rTrackId = 0;
            string rTitle = "";
            string rArtist = "";
            string rUrl = "";
            int rLength = 0;
            string rGenre = "";
            bool rIsVideo = false;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdSelectAll))
                {
                    connection.Open();
                    cmd.Connection = connection;
                    SqlDataReader rDr = cmd.ExecuteReader();

                    while (rDr.Read())
                    {
                        if (!rDr.IsDBNull(0))
                        {
                            rTrackId = rDr.GetInt32(0);
                            rTitle = rDr.GetString(1);
                            rArtist = rDr.GetString(2);
                            rUrl = rDr.GetString(3);
                            rLength = rDr.GetInt32(4);
                            rGenre = rDr.GetString(5);
                            rIsVideo = rDr.GetBoolean(6);
                            track = new Track(rTitle, rArtist, rUrl, rLength, rGenre,
                                              rIsVideo);
                            track.TrackId = rTrackId;
                        }
                        else
                        {
                            track = new Track("", "", "", 0, "", false);
                        }
                        result.Add(track);
                    }
                }
            }

            return result;
        }
    }
```

TracklistDAO.cs

```
    public class TracklistDAO : ITracklist
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;

        const string CmdAddTracklist = @"INSERT INTO [dbo].[Tracklist] (playlistId,
                                        userId, trackId) VALUES (@pPlaylistId,
                                        @pUserId, @pTrackId)";
        const string CmdGetTracksRecommendedByUser = @"SELECT t.trackId, t.title,
                                                    t.artist, t.url, t.length,
                                                     t.genre, t.isVideo
                                                     FROM [dbo].[Tracklist] tl,
                                                    [dbo].[Track] t
                                                     WHERE tl.userId = @pUserId AND
                                                    t.trackId = tl.trackId";
        const string CmdGetAllTracklists = @"SELECT * FROM [dbo].[Tracklist]";

        public bool AddTracklist(Tracklist newTracklist)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdAddTracklist))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pPlaylistId",
                                        newTracklist.PlaylistId));
                    cmd.Parameters.Add(new SqlParameter("@pUserId",
                                        newTracklist.UserId));
                    cmd.Parameters.Add(new SqlParameter("@pTrackId",
                                        newTracklist.TrackId));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
        }

        public List<Track> GetTracksRecommendedByUser(int userId)
        {
            List<Track> result = new List<Track>();
            Track track = null;
            int rTrackId = 0;
            string rTitle = "";
            string rArtist = "";
            string rUrl = "";
            int rLength = 0;
            string rGenre = "";
            bool rIsVideo = false;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdGetTracksRecommendedByUser))
                {
                    connection.Open();
                    cmd.Connection = connection;
                    cmd.Parameters.Add(new SqlParameter("@pUserId", userId));
```

```
                using (SqlDataReader rDr = cmd.ExecuteReader())
                {
                    while (rDr.Read())
                    {
                        if (!rDr.IsDBNull(0))
                        {
                            rTrackId = rDr.GetInt32(0);
                            rTitle = rDr.GetString(1);
                            rArtist = rDr.GetString(2);
                            rUrl = rDr.GetString(3);
                            rLength = rDr.GetInt32(4);
                            rGenre = rDr.GetString(5);
                            rIsVideo = rDr.GetBoolean(6);
                            track = new Track(rTitle, rArtist, rUrl, rLength,
                                              rGenre, rIsVideo);
                            track.TrackId = rTrackId;
                        }
                        else
                        {
                            track = new Track("", "", "", 0, "", false);
                        }
                        result.Add(track);
                    }
                }
            }
        }

        return result;
    }

    public List<Tracklist> GetAllTracklists()
    {
        List<Tracklist> result = new List<Tracklist>();
        Tracklist tl = null;
        int rUserId = 0;
        int rPlaylistId = 0;
        int rTrackId = 0;

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            using (SqlCommand cmd = new SqlCommand(CmdGetAllTracklists))
            {
                connection.Open();
                cmd.Connection = connection;
                using (SqlDataReader rDr = cmd.ExecuteReader())
                {
                    while (rDr.Read())
                    {
                        if (!rDr.IsDBNull(0))
                        {
                            rUserId = rDr.GetInt32(1);
                            rPlaylistId = rDr.GetInt32(0);
                            rTrackId = rDr.GetInt32(2);
                            tl = new Tracklist(rPlaylistId, rUserId, rTrackId);
                        }
                        else
                        {
                            tl = new Tracklist(-1, -1, -1);
                        }
                        result.Add(tl);
                    }
                }
            }
```

```
            }

            return result;
        }
    }
```

UserDAO.cs

```
    public class UserDAO : IUser
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;

        const string CmdInsert = @"INSERT INTO [dbo].[User] (email, password, isAdmin,
                                    name)
                                     VALUES (@pEmail, @pPassword, @pIsAdmin,
                                    @pName)";
        const string CmdDelete = @"DELETE FROM [dbo].[User] WHERE userId = @pUserId";
        const string CmdSearch = @"SELECT * FROM [dbo].[User] WHERE userId =
                                                @pUserId";
        const string CmdSelectAll = @"SELECT * FROM [dbo].[User]";

        public bool InsertUser(User user)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdInsert))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pEmail", user.Email));
                    cmd.Parameters.Add(new SqlParameter("@pPassword", user.Password));
                    cmd.Parameters.Add(new SqlParameter("@pIsAdmin", user.IsAdmin));
                    cmd.Parameters.Add(new SqlParameter("@pName", user.Name));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
        }

        public bool DeleteUser(int id)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdDelete))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pUserId", id));

                    if (cmd.ExecuteNonQuery() == 1)
                        return true;
                    else
                        return false;
                }
            }
        }
```

```
public List<User> GetAllUser()
{
    List<User> result = new List<User>();
    User user = null;
    int rUserId = 0;
    string rName = "";
    string rEmail = "";
    bool rIsAdmin = false;

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand cmd = new SqlCommand(CmdSelectAll))
        {
            connection.Open();
            cmd.Connection = connection;
            using (SqlDataReader rDr = cmd.ExecuteReader())
            {
                while (rDr.Read())
                {
                    if (!rDr.IsDBNull(0))
                    {
                        rUserId = rDr.GetInt32(0);
                        rEmail = rDr.GetString(1);
                        rIsAdmin = rDr.GetBoolean(3);
                        rName = rDr.GetString(4);
                        user = new User(rName, "", rEmail, rIsAdmin);
                        user.UserId = rUserId;
                    }
                    else
                    {
                        user = new User("", "", "", false);
                    }
                    result.Add(user);
                }
            }
        }
    }

    return result;
}

public User FindUserById(int id)
{
    User user = null;
    int rUserId = 0;
    string rName = "";
    string rEmail = "";
    bool rIsAdmin = false;

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand cmd = new SqlCommand(CmdSearch))
        {
            connection.Open();
            cmd.Connection = connection;

            cmd.Parameters.Add(new SqlParameter("@pUserId", id));

            SqlDataReader rDr = cmd.ExecuteReader();

            while (rDr.Read())
            {
```

```
                        if (!rDr.IsDBNull(0))
                        {
                            rUserId = rDr.GetInt32(0);
                            rEmail = rDr.GetString(1);
                            rIsAdmin = rDr.GetBoolean(3);
                            rName = rDr.GetString(4);
                            user = new User(rName, "", rEmail, rIsAdmin);
                            user.UserId = rUserId;
                        }
                    }
                }
            }
            return user;
        }
    }
```

VoteDAO.cs

```
    public class VoteDAO : IVote
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;

        const string CmdAddVote = @"INSERT INTO [dbo].[Vote] (userId, playlistId,
                                    trackId, TS_created)
                                     VALUES (@pUserId, @pPlaylistId, @pTrackId,
                                          @pTS_created)";
        const string CmdAlreadyVotedForTrack = @"SELECT * FROM [dbo].[Vote] WHERE
                                                   userId = @pUserId
                                                AND trackId = @pTrackId AND
                                                   playlistId = @pPlaylistId";
        const string CmdGetVotesForTrack = @"SELECT count(*) FROM [dbo].[Vote] WHERE
                                   trackId = @pTrackId AND playlistId = @pPlaylistId";

        public bool AddVote(Vote newVote)
        {
            if (!AlreadyVotedForTrack(newVote.TrackId, newVote.UserId,
                                                   newVote.PlaylistId))
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    using (SqlCommand cmd = new SqlCommand(CmdAddVote))
                    {
                        connection.Open();
                        cmd.Connection = connection;

                        cmd.Parameters.Add(new SqlParameter("@pPlaylistId",
                                                   newVote.PlaylistId));
                        cmd.Parameters.Add(new SqlParameter("@pUserId",
                                                   newVote.UserId));
                        cmd.Parameters.Add(new SqlParameter("@pTrackId",
                                                   newVote.TrackId));
                        cmd.Parameters.Add(new SqlParameter("@pTS_created",
                                                   newVote.TS_created));

                        if (cmd.ExecuteNonQuery() == 1)
                            return true;
                        else
                            return false;
                    }
                }
            }
            else
```

```
            {
                return false;
            }
        }

        public bool AlreadyVotedForTrack(int trackId, int userId, int playlistId)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdAlreadyVotedForTrack))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pUserId", userId));
                    cmd.Parameters.Add(new SqlParameter("@pPlaylistId", playlistId));
                    cmd.Parameters.Add(new SqlParameter("@pTrackId", trackId));

                    using (SqlDataReader rDr = cmd.ExecuteReader())
                    {
                        while (rDr.Read())
                        {
                            if (!rDr.IsDBNull(0))
                                return true;
                            else
                                return false;
                        }
                        return false;
                    }
                }
            }
        }

        public int GetVotesForTrack(int trackId, int playlistId)
        {
            int result = 0;
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(CmdGetVotesForTrack))
                {
                    connection.Open();
                    cmd.Connection = connection;

                    cmd.Parameters.Add(new SqlParameter("@pTrackId", trackId));
                    cmd.Parameters.Add(new SqlParameter("@pPlaylistId", playlistId));

                    using (SqlDataReader rDr = cmd.ExecuteReader())
                    {
                        while (rDr.Read())
                        {
                            result = rDr.GetInt32(0);
                        }
                        return result;
                    }
                }
            }
        }
    }
```

CrowdDJ.DomainClasses.*

Guest.cs

```
    public class Guest
    {
        public Guest(int userId, int partyId)
        {
            PartyId = partyId;
            UserId = userId;
        }

        private int userId;
        public int UserId
        {
            get { return userId; }
            set { userId = value; }
        }

        private int partyId;
        public int PartyId
        {
            get { return partyId; }
            set { partyId = value; }
        }

    }
```

Party.cs

```
    public class Party
    {
        public Party(string name, string location, string host, string partyBegin,
string partyEnd, bool isActive)
        {
            Name = name;
            Location = location;
            Host = host;
            PartyBegin = partyBegin;
            PartyEnd = partyEnd;
            IsActive = isActive;
        }

        private int partyId;

        public int PartyId
        {
            get { return partyId; }
            set { partyId = value; }
        }


        private string name;
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        private string location;
        public string Location
        {
            get { return location; }
            set { location = value; }
```

```
        }

        private string host;
        public string Host
        {
            get { return host; }
            set { host = value; }
        }

        private string partyBegin;
        public string PartyBegin
        {
            get { return partyBegin; }
            set { partyBegin = value; }
        }

        private string partyEnd;
        public string PartyEnd
        {
            get { return partyEnd; }
            set { partyEnd = value; }
        }

        private bool isActive;
        public bool IsActive
        {
            get { return isActive; }
            set { isActive = value; }
        }


    }
```

Partytweet.cs

```
public class Partytweet
{
    public Partytweet(int userId, int partyId, string message)
    {
        UserId = userId;
        PartyId = partyId;
        Message = message;
    }

    private int userId;
    public int UserId
    {
        get { return userId; }
        set { userId = value; }
    }

    private int partyId;
    public int PartyId
    {
        get { return partyId; }
        set { partyId = value; }
    }

    private string message;
    public string Message
    {
        get { return message; }
        set { message = value; }
    }

}
```

Playlist.cs

```
public class Playlist
{
    public Playlist(int playlistId, string name)
    {
        PlaylistId = playlistId;
        Name = name;
    }

    private int playlistId;
    public int PlaylistId
    {
        get { return playlistId; }
        set { playlistId = value; }
    }

    private string name;
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
}
```

Track.cs

```
    public class Track
    {
        public Track(string title, string artist, string url, int length, string
genre, bool isVideo)
        {
            Title = title;
            Artist = artist;
            Url = url;
            Length = length;
            Genre = genre;
            IsVideo = isVideo;
        }

        private int trackId;
        public int TrackId
        {
            get { return trackId; }
            set { trackId = value; }
        }


        private string title;
        public string Title
        {
            get { return title; }
            set { title = value; }
        }

        private string artist;
        public string Artist
        {
            get { return artist; }
            set { artist = value; }
        }

        private string url;
        public string Url
        {
            get { return url; }
            set { url = value; }
        }

        private int length;
        public int Length
        {
            get { return length; }
            set { length = value; }
        }

        private string genre;
        public string Genre
        {
            get { return genre; }
            set { genre = value; }
        }

        private bool isVideo;
        public bool IsVideo
        {
            get { return isVideo; }
            set { isVideo = value; }
        }
```

```
    }
```

Tracklist.cs

```
    public class Tracklist
    {
        public Tracklist(int playlistId, int userId, int trackId)
        {
            PlaylistId = playlistId;
            UserId = userId;
            TrackId = trackId;
        }

        private int palylistId;
        public int PlaylistId
        {
            get { return palylistId; }
            set { palylistId = value; }
        }

        private int userId;
        public int UserId
        {
            get { return userId; }
            set { userId = value; }
        }

        private int trackId;
        public int TrackId
        {
            get { return trackId; }
            set { trackId = value; }
        }

    }
```

User.cs

```
    public class User
    {

        public User(string name, string password, string email, bool isAdmin)
        {
            Name = name;
            Email = email;
            Password = password;
            IsAdmin = isAdmin;
        }

        private int userId;
        public int UserId
        {
            get { return userId; }
            set { userId = value; }
        }


        private string name;
        public string Name
        {
            get { return name; }
            set { name = value; }
```

```
        }

        private string email;
        public string Email
        {
            get { return email; }
            set { email = value; }
        }

        private string password;
        public string Password
        {
            get { return password; }
            set { password = value; }
        }

        private bool isAdmin;
        public bool IsAdmin
        {
            get { return isAdmin; }
            set { isAdmin = value; }
        }


    }
```

Vote.cs

```
    public class Vote
    {
        public Vote(int userId, int playlistId, int trackId, string ts_created)
        {
            PlaylistId = playlistId;
            TrackId = trackId;
            TS_created = ts_created;
        }

        private int userId;
        public int UserId
        {
            get { return userId; }
            set { userId = value; }
        }

        private int playlistId;
        public int PlaylistId
        {
            get { return playlistId; }
            set { playlistId = value; }
        }

        private int trackId;
        public int TrackId
        {
            get { return trackId; }
            set { trackId = value; }
        }

        private string ts_created;
        public string TS_created
        {
            get { return ts_created; }
            set { ts_created = value; }
```

```
        }

    }
```

CrowdDJ.Interfaces.*

IGuest

```
    public interface IGuest
    {
        bool AddGuest(Guest newGuest);
        bool RemoveGuest(Guest removeGuest);
        bool PartyIsVisitedByGuest(int searchGuestId, int partyId);
        List<User> GetGuestlistForParty(int partyId);
    }
```

IParty

```
    public interface IParty
    {
        bool AddParty(Party newParty);
        bool RemovePartyWithId(int partyId);
        Party FindPartyById(int partyId);
        //List<Party> FindPartyBegin(DateTime begin);
        //List<Party> FindPartyEnd(DateTime end);
        List<Party> FindPartyWithHost(string hostName);
        List<Party> GetAllParties();
    }
```

IPartytweet

```
    public interface IPartytweet
    {
        bool AddTweet(Partytweet newTweet);
        List<Partytweet> GetTweetsForParty(int partyId);
        List<Partytweet> GetAllTweets();
    }
```

IPlaylist

```
    public interface IPlaylist
    {
        bool AddPlaylist(Playlist newPlaylist);
        Playlist GetPlaylistForParty(int id);
        List<Track> GetAllTracksInPlaylist(int playlistId);
        List<Playlist> GetAllPlaylists();
    }
```

ITrack

```
    public interface ITrack
    {
        bool AddTrack(Track newTrack);
        bool RemoveTrackWithId(int id);
        List<Track> FindTrackWithTitle(string title);
        List<Track> FindTracksInGenre(string genre);
        List<Track> FindVideos();
        List<Track> FindSongs();
        List<Track> GetAllTracks();
    }
```

ITracklist

```
    public interface ITracklist
    {
        bool AddTracklist(Tracklist newTracklist);
        List<Track> GetTracksRecommendedByUser(int userId);
        List<Tracklist> GetAllTracklists();
```

```
    }
```

IUser

```
    public interface IUser
    {
        bool InsertUser(User user);
        bool DeleteUser(int id);
        List<User> GetAllUser();
        User FindUserById(int id);
    }
}
```

IVote

```
    public interface IVote
    {
        bool AddVote(Vote newVote);
        bool AlreadyVotedForTrack(int trackId, int userId, int playlistId);
        int GetVotesForTrack(int trackId, int partyId);
    }
}
```

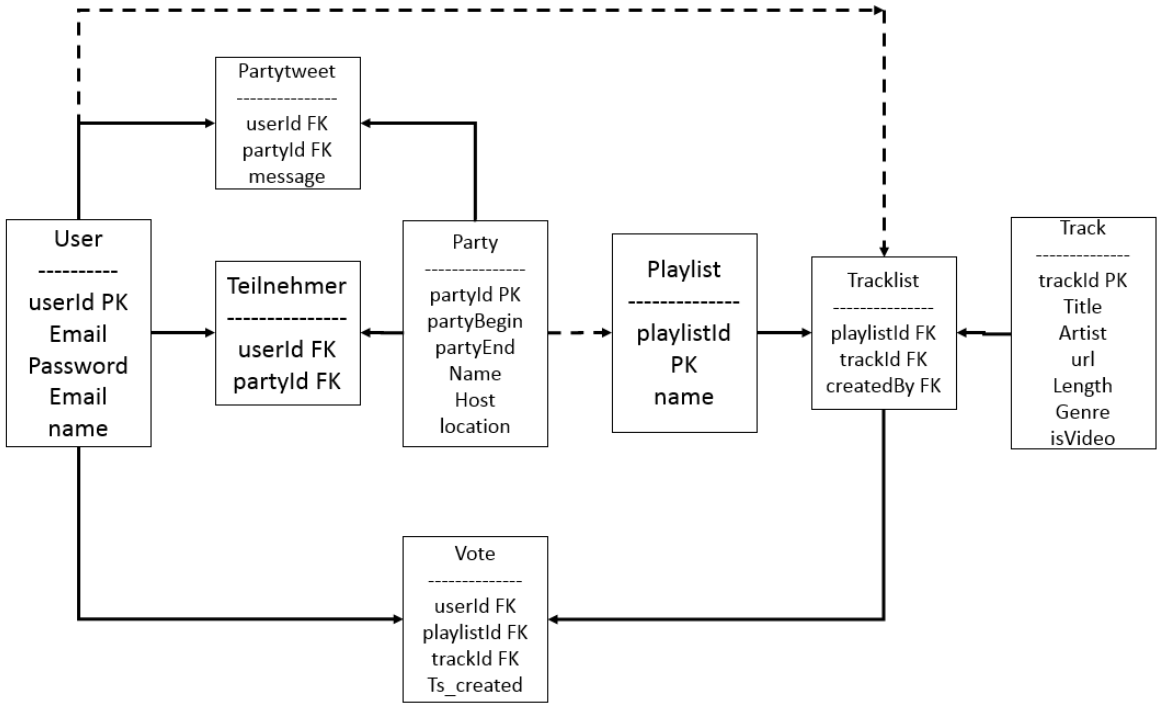
Program.cs

```
class Program
    {
        static void Main(string[] args)
        {
            BusinessLogic bl = new BusinessLogic();


            Random random = new Random();
            bool r;
            User user = null;
            for (int i = 13; i < 2000; i++)
            {
                if (random.Next(0, 1) == 0)
                    r = true;
                else
                    r = false;
                user = new User("Peter der " + i + ".", i.GetHashCode().ToString(),
                            "ichBinnnnn" + i + "@dar.at", r);
                bl.userDAO.InsertUser(user);
            }

            Party party = null;
            for (int i = 5; i < 2000; i++)
            {
                if (random.Next(0, 1) == 0)
                    r = true;
                else
                    r = false;
                party = new Party("Die " + i + ". wilde Hilde", "Am Berg " + i, "Susi
                                  von der " + i + ".",
                                 (i % 24).ToString() + ".00", (i %
                                  24).ToString() + ".00", r);
                bl.partyDAO.AddParty(party);
            }

            Track track = null;
            for (int i = 5; i < 2000; i++)
            {
                if (random.Next(0, 1) == 0)
```

```
                r = true;
            else
                r = false;
            track = new Track("Bloodbath Massacre " + i, i.ToString(), "www." + i
                              + ".com", (i % 300), "PoP", r);
            bl.trackDAO.AddTrack(track);
        }

        Guest guest = null;
        for (int i = 13; i < 500; i++)
        {
            guest = new Guest(i, i);
            bl.guestDAO.AddGuest(guest);
        }

        Partytweet partytweet = null;
        for (int i = 0; i < 230; i++)
        {
            partytweet = new Partytweet((i % 500) + 20, (i % 500) + 20, "Meine " +
                                        i + "-te Partey!");
            bl.partytweetDAO.AddTweet(partytweet);
        }

        Playlist playlist = null;
        for (int i = 6; i < 2000; i++)
        {
            playlist = new Playlist(i, "Das Beste im " + i + "er Pack");
            bl.playlistDAO.AddPlaylist(playlist);
        }

        Tracklist tracklist = null;
        for (int i = 5; i < 10; i++)
        {
            tracklist = new Tracklist(i, i + 20, i + 20);
            bl.tracklistDAO.AddTracklist(tracklist);

}}}}
```

Datenbank

Datenbankschema



Zur genierung der Tabelle wurden die Funktionalitäten von Visual Studio hergenommen!

Erstellungsskripte

Guest

```
CREATE TABLE [dbo].[Guest] (
    [userId]  INT NOT NULL,
    [partyId] INT NOT NULL,
    CONSTRAINT [GuestToUserFK] FOREIGN KEY ([userId]) REFERENCES [dbo].[User]
([userId]),
    CONSTRAINT [GuestToPartyFK] FOREIGN KEY ([partyId]) REFERENCES [dbo].[Party]
([partyId])
);
```

Party

```
CREATE TABLE [dbo].[Party] (
    [partyId]    INT          IDENTITY (1, 1) NOT NULL,
    [name]       VARCHAR (50) NOT NULL,
    [location]   VARCHAR (50) NOT NULL,
    [host]       VARCHAR (50) NOT NULL,
    [partyBegin] VARCHAR (50) NOT NULL,
    [partyEnd]   VARCHAR (50) NOT NULL,
    [isActive]   BIT          NOT NULL,
    CONSTRAINT [PK_Party] PRIMARY KEY CLUSTERED ([partyId] ASC)
);
```

Partytweet

```
CREATE TABLE [dbo].[PartyTweet] (
    [userId]  INT  NOT NULL,
    [partyId] INT  NOT NULL,
    [message] TEXT NOT NULL,
    CONSTRAINT [PartytweetToUserFk] FOREIGN KEY ([userId]) REFERENCES [dbo].[User]
([userId]),
    CONSTRAINT [PartytweetToPartyFk] FOREIGN KEY ([partyId]) REFERENCES [dbo].[Party]
([partyId])
);
```

Playlist

```
CREATE TABLE [dbo].[Playlist] (
    [playlistId] INT          NOT NULL,
    [name]       VARCHAR (50) NOT NULL,
    CONSTRAINT [PK_Playlist] PRIMARY KEY CLUSTERED ([playlistId] ASC),
    CONSTRAINT [PlaylistToPartyFk] FOREIGN KEY ([playlistId]) REFERENCES [dbo].[Party]
([partyId])
);
```

Track

```
CREATE TABLE [dbo].[Track] (
    [trackId] INT          IDENTITY (1, 1) NOT NULL,
    [title]   VARCHAR (50) NOT NULL,
    [artist]  VARCHAR (50) NOT NULL,
    [url]     VARCHAR (50) NOT NULL,
    [length]  INT          NULL,
    [genre]   VARCHAR (50) NULL,
    [isVideo] BIT          NOT NULL,
    CONSTRAINT [PK_Track] PRIMARY KEY CLUSTERED ([trackId] ASC)
);
```

Tracklist

```
CREATE TABLE [dbo].[Tracklist] (
    [playlistId] INT NOT NULL,
    [userId]     INT NOT NULL,
    [trackId]    INT NOT NULL,
    CONSTRAINT [tracklistToPlaylistFK] FOREIGN KEY ([playlistId]) REFERENCES
[dbo].[Playlist] ([playlistId]),
    CONSTRAINT [TracklistToUserFk] FOREIGN KEY ([userId]) REFERENCES [dbo].[User]
([userId]),
    CONSTRAINT [tracklistToTrackFK] FOREIGN KEY ([trackId]) REFERENCES [dbo].[Track]
([trackId])
);
```

User

```
CREATE TABLE [dbo].[User] (
    [userId]   INT          IDENTITY (1, 1) NOT NULL,
    [email]    VARCHAR (50) NOT NULL,
    [password] VARCHAR (50) NOT NULL,
    [isAdmin]  BIT          NOT NULL,
    [name]     VARCHAR (50) NOT NULL,
    CONSTRAINT [PK_User] PRIMARY KEY CLUSTERED ([userId] ASC)
);
```

Vote

```
CREATE TABLE [dbo].[Vote] (
    [userId]     INT           NOT NULL,
    [playlistId] INT           NOT NULL,
    [trackId]    INT           NOT NULL,
    [TS_created] VARCHAR (50) NOT NULL,
    CONSTRAINT [VoteToPlaylistFk] FOREIGN KEY ([playlistId]) REFERENCES
[dbo].[Playlist] ([playlistId]),
    CONSTRAINT [VoteToUserFk] FOREIGN KEY ([userId]) REFERENCES [dbo].[User]
([userId]),
    CONSTRAINT [VoteToTrackFk] FOREIGN KEY ([trackId]) REFERENCES [dbo].[Track]
([trackId])
);
```

Abfragen

	userId	email	password	isAdmin	name
1	13	www@email.at	aaa	0	Werner
2	14	sssssssss@email.at	aaass	1	Hans
3	15	www@email.at	aaa	0	Werner
4	16	sssssssss@email.at	aaass	1	Hans
5	17	www@email.at	aaa	0	Werner

	partyId	name	location	host	partyBegin	partyEnd	isActive
1	5	Swagerino	Hgb	Werner	22.11.2013 18.00	22.11.2013 24.00	0
2	6	Swageri...	Linz	Schr...	22.11.2013 18.00	22.11.2013 24.00	1
3	7	Swagerino	Hgb	Werner	22.11.2013 18.00	22.11.2013 24.00	0
4	8	Swageri...	Linz	Schr...	22.11.2013 18.00	22.11.2013 24.00	1
5	9	Swagerino	Hgb	Werner	22.11.2013 18.00	22.11.2013 24.00	0

	trackId	title	artist	url	length	genre	isVideo
1	5	Dancing	Bobby	ww.ww.ww.	180	Metallerino	0
2	6	Partoy	Mo...	ww.ss.ww.	231	Metallerino	1
3	7	Dancing	Bobby	ww.ww.ww.	180	Metallerino	0
4	8	Partoy	Mo...	ww.ss.ww.	231	Metallerino	1
5	9	Blood	5	www.5.com	5	PoP	1

	playlistId	name
1	5	the dream is real
2	6	Das Beste im ...
3	7	Das Beste im ...
4	8	Das Beste im ...
5	9	Das Beste im ...

	playlistId	userId	trackId
1	5	15	6
2	5	15	6
3	5	15	8
4	5	25	25
5	6	26	26

	userId	partyId	message
1	13	5	YOLO VOLL SWAGGA DIE PARTY!
2	15	5	YOLO VOLL sheeeet DIE PARTY!

Abfrage erfolgreich ausgeführt um 23:08:19. | (LocalDB)\v11.0 (11.0 SP1) | OLLYWOOD\Ollywood.II (53) | D:\UNI\SEM5\SWK\PROJEK... | 00:00:00 | 8243 Zeilen