«interface» Chlorine **IUserAccess** «Property» «Property» AddUser(string, string, string):bool FindUserByEmail(string):User + Id :int + Timestamp :string + PoolId :int Value :double IsE mailInUse (string): bool Datald :int Validate Password(string, string):bo RemoveUser(string) :void DeleteAllUsers() :void + Data() «property» + Data() :Data «property» + Pool() :Pool + Chlorine() :ICollection<Chlorine> + Humidity() :ICollection<Humidity> EditUserPassword(string, string) :bool EditUserEmail(string, string) :bool EditUserName(string, string):bool pH() :ICollection<pH> Temperature() :ICollection<Temperature: UserAccess «interface» PoolAccess IPo ol Ac cess AddUser(string, string, string) :bool ValidateName(string) :bool «Property» «Property» UserAccess :IUserAccess UserAccess : IUserAccess FindUserByEmail(string) :User IsE mailInUse (string) :bool AddPool(string, string, double) :bool PoolAccess(IUserAccess) Validate Password (string, string) :boo IsPoolNameAvailable(string, string):bool AddPool(string, string, double) :bool RemoveUser(string) :void FindSpecificPool(string, string):Pool RemovePool(string, string):bool IsPoolNameAvailable(string, string):bool FindSpecificPool(string, string):Pool DeleteAllUsers() :void EditUserPassword(string, string) :bool DeleteAllPools():void EditPoolName(string, string, string):bool RemovePool(string, string):bool DeleteAllPools():void EditUserEmail(string, string):bool EditUserName(string, string) :bool EditPoolVolume(string, string, double) :bo EditPoolUser(string, string, string) :bool EditPoolName(string, string, string) :bool EditPoolVolume(string, string, double) :boo FindAllPoolsOfUser(string) :List<Pool> EditPoolUser(string, string, string):bool FindAllPoolsOfUser(string) :List<Pool> Humidity «interface» Exception Exception IDataAccess PoolNotFoundExceptio «Property» UserNotFoundException «Property» ld :int Value :double PoolAccess : IPoolAccess Datald :int + CreateDataEntry(string, string, double, double, double, double) :bool + DeleteAllData() :void «property» + Data():Data GetChlorineValues(string, string, int):List<Tuple<SensorTypes, double>> GetTemperatureValues(string, string, int):List<Tuple<SensorTypes, double Exception Multiple Occourences Of Email Was Found Exception GetPhValues(string, string, int) :List<Tuple<SensorTypes, double>> GetHumidityValues(string, string, int) :List<Tuple<SensorTypes, double>> DataAccess **ISmartpoolDB** User «Property» «Property» «Property» PoolAccess : IPoolAccess + PoolAccess : IPoolAccess «Property» UserAccess :IUserAccess Value :doubl + Id :int + Firstname :string + Middelname :string DataAccess(IPoolAccess) CreateDataEntry(string, string, double, double, double, double) :bool DataAccess :IDataAccess Datald :int ClearEntireDatabase():void DeleteAllData() :void Lastname :string Email :string «property» GetChlorineValues(string, string, int) :List<Tuple<SensorTypes, double>> Data():Data GetTemperatureValues(string, string, int):List<Tuple<SensorTypes, double GetPhValues(string, string, int):List<Tuple<SensorTypes, double>> Password :string GetHumidityValues(string, string, int) :List<Tuple<SensorTypes, double>> + Use r() SmartpoolDB «property» + Pool() :ICollection<Pool:</pre> «Property» Db Conte xt PoolAccess : IPoolAccess Temperature Pool DatabaseContext UserAccess : IUserAccess DataAccess :IDataAccess «Property» «Property» DatabaseContext() ld :int SmartpoolDB (IDataAccess) On Model Creating (DbM odel Builder) :voi Value :double Name :string ClearEntireDatabase():vo Datald :int Volume :double UserSet() :DbSet<User> UserId :int PoolSet() :DbSet<Pool> «property» + Data():Data pHSet() :DbSet<pH> + Pool() Chlorine Set(): Db Set<Chlorine> Temperature Set(): DbSet<Temperature «property» + User() :User + Data() :ICollection<Data> HumiditySet():DbSet<Humidity>DataSet():DbSet<Data>