Na tę chwilę to jest tylko żywy kod, trzeba będzie go jakoś obudować

Tworzenie funkcji SQL

* GetTransactionTypes()

Zwraca zawartość tabeli TransactionTypes

SELECT \* FROM TransactionTypes;

* GetTransactions()

Zwraca zawartość tabeli Transactions

SELECT \* FROM Transactions;

* GetCustomers()

Zwraca zawartość tabeli Customers

SELECT \* FROM Customers;

* GetEntrances()

Zwraca zawartość tabeli Entrances

SELECT \* FROM Entrances;

* GetExits()

Zwraca zawartość tabeli Exits

SELECT \* FROM Exits;

* GetServices()

Zwraca zawartość tabeli Services

SELECT \* FROM Services;

* GetWatches()

Zwraca zawartość tabeli Watches

SELECT \* FROM Watches;

* GetEntranceMethods()

Zwraca zawartość tabeli EntranceMethods

SELECT \* FROM EntranceMethods;

* GetDiscounts()

Zwraca zawartość tabeli Discounts

SELECT \* FROM Discounts;

* GetCharges()

Zwraca zawartość tabeli Charges

SELECT \* FROM Charges;

* GetVisits()

Zwraca zawartość tabeli Visits

SELECT \* FROM Visits;

* 1. Customers
* GetCustomerDiscount(int CustomerId)

Funkcja przyjmująca jako parametr identyfikator klienta, zwracająca jaką dany klient ma zniżkę. Wykorzystywana do obliczenia kwoty, jaką klient płaci przy wejściu na obiekt

SELECT Discounts.amount FROM Discounts WHERE discounts.id = @CustomerId;

* CreateCustomer(int CustomerId, varchar(50) FirstName, varchar(50) LastName, int DiscountId)

Funkcja tworząca nowego klienta. Służy do przechowywania danych o kliencie odwiedzającym obiekt w przeszłości

INSERT INTO CUSTOMERS (Id, FirstName, LastName, DiscountId)

VALUES (@Id, @FisrtName, @LastName, @Discount);

* 1. TransactionTypes
* GetTransactionType(int TransactionTypeId)

Funkcja zwracająca typ transakcji, jaką przeprowadził klient znajdując się w środku obiektu. Może to być “0 - wyjście lub 1 - wejście”. Na podstawie tych danych przy wyjściu klienta dopasowujemy wejścia i wyjścia klienta na dany obiekt, dzięki któremy możemy sprawdzić, czy np. klient nie oszukiwał i nie przeszedł pod bramką przy wejściu (nie będzie zgadzać się stosunek wejścia/wyjścia)

SELECT TransactionType.name FROM transactiontype WHERE transactiontype.id = @TransactionTypeId;

* 1. Transactions
* CreateTransaction(int ServiceId, int WatchId, int TransactionTypeId, datetime Time)

Funkcja tworząca nową transakcję. Jest to moment, kiedy klient przykłada zegarek do bramek przy wejściu na daną usługę. Transakcją jest zarówno wejście jak i wyjście na określoną strefę, czy to jest wyjście czy wejście determinuje TransactionTypeId.

INSERT INTO TRANSACTION (Id, Serviceid, TransactionTypeId, time)

VALUES (@Id, @ServiceId,@WatchId, @TransactionTypeId,sysdate);

* GetListOfTransactions(int WatchId)

Funkcja zwracająca listę wszystkich transakcji. Wykorzystywana do sprawdzenia, z jakich usług korzystał klient podczas swojego pobytu na obiekcie. Do identyfikacji klienta służy ID zegarka, z którego korzystał

SELECT \* FROM transaction WHERE watchid = @WatchId;

* DeleteTransactions(int WatchId)

Funkcja usuwająca wpis o zaistniałej transakcji przypisanej do danego zegarka. Jest konieczna, ponieważ rozważamy korzystanie z tego samego zegarka przez kilku klientów. W celu wejścia na obiekt z “czystym kontem” konieczne jest wyzerowanie transakcji przypisanych do zegarka

DELTE \* FROM transaction WHERE watchid = @WatchId;

* 1. Services
* GetPricePerHour(int ServiceId)

Funkcja zwracająca koszt usługi za godziny. Potrzebna do uiszczania opłaty podczas wejścia na obiekt

SELECT Services.priceperhour FROM services WHERE services.serviceid = @ServiceId;

* GetPricePerMinute(int ServiceId)

Funkcja zwracająca koszt usługi za minutę. Potrzebna do uiszczania opłaty podczas wyjścia z obiektu - mnożona przez ilość minut spędzonych na dodatkowo płatnej strefie

SELECT Services.priceperminute FROM services WHERE services.serviceid = @ServiceId;

* GetServiceName(int ServiceId)

Funkcja zwracająca nazwę usługi po ID. Potrzebna w celu podsumowania klientowi, na jakich usługach spędzał czas.

SELECT Services.name FROM services WHERE services.serviceid = @ServiceId;

* GetFreeServicesId(int EntranceMethodId)

Funkcja zwracająca listę płatnych stref, na które klient ma darmowy wstęp dzięki danej karcie benefit

???? Tego nie wiem

* 1. Watches
* GetCustomerId(int WatchId)

Funkcja zwracająca ID klienta, do którego jest przypisany zegarek. Wykorzystywana w celu identyfikacji klienta przypisanego do danego zegarka

SELECT Watches.Customerid FROM watches WHERE watches.watchid = @WatchId;

* GetServiceId(int WatchId)

Funkcja zwracająca usługę, na jaką wykupione jest wejście w danym zegarku. Przykład: Klient na wejściu wykupił godzinę na basenie oraz godzinę na saunie. Dzięki takiej informacji podczas wejścia na strefę siłowni system wie, żeby nie naliczać opłaty za minuty

SELECT Watches.Serviceid FROM watches WHERE watches.watchid = @WatchId;

* 1. Discounts
* GetDiscountAmount(int DiscountId)

Funkcja zwracająca wielkość zniżki. Dzięki niej można wyliczyć cenę biletu

SELECT Discounts.amount FROM discounts WHERE discounts.discountid = @DiscountId;

* 1. Charges
* CreateCharge(int Amount, int WatchId)

Funkcja dodająca nowe obciążenie do danego zegarka.

INSERT INTO CHARGES (Amount, WatchId)

VALUES (@Amount, @WatchId);

* GetSumOfWatchCharges(int WatchId)

Funkcja zwracająca sumę wszystkich obciążeń przypisanych dodanego zegarka. Jest niezbędna do rozliczenia klienta z odwiedzanych przez niego stref oraz opóźnień przy jego wyjściu

SELECT SUM(Charge.amount) FROM charge Where charge.WatchId =@WatchId;

* 1. Entrances
* CreateEntrance(int WatchId, int EntranceMethodId, datetime Time, int CustomerId)

Funkcja wywoływana podczas wejścia nowego klienta na obiekt.

INSERT INTO ENTRANCES (WatchId, EntranceMethodId,Time, CustomerId)

VALUES (@WatchId,@EntranceMethodId, sysdate,@CustomerId,);

* GetEntranceTime(int WatchId)

Funkcja zwracająca godzinę, o której klient wszedł na obiekt. Służy do rozliczania klienta ze spóźnień

SELECT Entrances.TIME FROM Entrances WHERE entrances.Watchid = @WatchId;

* 1. Exits
* CreateExit(int WatchId, datetime Time)

Funkcja wywoływana podczas wyjścia klienta z obiektu.

INSERT INTO EXITS (WatchId, Time)

VALUES (@WatchId, sysdate);

* GetExitTime(int WatchId)

Tak samo jak funkcja GetEntranceTime służy do rozliczenia spóźnienia

SELECT Exits.TIME FROM Exits WHERE Exits.Watchid = @WatchId;

* 1. Visits
* CreateVisit(int WatchId, int ServiceId, int CustomerId, decimal Amount)

Funkcja służąca do zapisywania wizyt

INSERT INTO Visits (WatchId, ServiceId, CustomerId, Amount)

VALUES (@WatchId, @ServiceId, @CustomerId, @Amount);