Mobile zoo - Xamarin or native

Create a mobile application using Xamarin.

This application is aimed at waiters in a restaurant. Application will allow waiters to receive notifications about orders and customer requests, take orders, make payments, control interactive tables and more.

To understand what is interactive table and imagine typical use-cases in the restaurant please look at http://itrestaurant.net/table.

Application flow, main views and suggested design can be found in attachment. Explanation for views and concepts can be found below.

Application usage scenario

Main view show stream of messages that come from customers in the restaurant. Customers send them by using interactive tables. Processed messages must be removed. Message can be processed by waiter using other device or user can cancel it, so message can disappear without user interaction.

There are few types of messages waiter can receive:
1. Waiter call – it's a general call for waiter.
2. Payment by card request .
3. Payment by cash request .
4. Request for cutlery.
5. Call for taxi.
6. Food is ready.

Waiter call

• When waiter receives Waiter call message, they need a way to easily navigate to order view. This view shows currently ordered items, total price, edit buttons. Waiter can manually add new items to order, add nested orders or process payment.

• Payment by card request.

• It's shortcut message to "Pay by card" view. Waiter need to enter 4 last digits of credit card into app.

• Payment by cash request

・It's shortcut to "Pay by cash view". It must display input for cash amount and show change.

・Request for cutlery/Call for taxi/Food is ready

・Just button for accepting (and removing) message. It's waiters responsibility to execute the task.

Additional task

・Add table management system to this application. Main view for it must be a table map, from where user can navigate to different tables and seats.

・In Table view user can see messages that are attached to specific table/seat.

・From table view user can move to table management view, where he can put table to sleep, wake it up or change background. Backgrounds are stored on the server and not on the device.

・Suggested application architecture.

・Your application by design requires server to communicate with. Because it's a test task, we don't require you to write server communication. Implement all data access logic as implementation of IRestaurantService interface where your implementation will serve data from local memory instead of making actual connections.

Task Priorities

1. It must be a complete product. You may drop some features if you feel that you don't have enough time/experience.

2. Nice GUI, animations, and visual appearance is a priority for this task.

3. Application architecture and code quality.

4. Additional task.

Notes

・There is no upper limit on complexity of the application. But better make finished task than aim high and miss.

・You can use free public libraries. Using of libraries must be justified. We expect to be able to build and run this application. Please ensure that it's easy.

・You can use assets from Internet.

・Blind copying of sample code is easily detectable. We know how to google.

・This is not a commercial product, restrictions on commercial use don't apply to you.

・We expect final task in form of zip archive with source code and assets or any accessible repository.

・Projects will be tested on Win10, build 10586, VS 2015 Pro Update 1, or above.

Data model

・All types and their relations are described below. This chapter works as suggestion and explanation for this document. You are not required to follow this data model exactly.

・On design images seats are referenced in format "<Table><Seat>", where seat letter is (A,B,C,D).

Order

One order at the restaurant
* Table
* Seat
* Total
* Status (open, paid, closed)
* Items – array of OrderItem
* PayFor – array of Order – nested orders for payment grouping
* Messages

OrderItem

One item in the order
* Name
* Price
* Status (new, ready)
* Reference to MenuItem

MenuItem

One menu item at the restaurant. May be infinitely nested into categories using composite OOP pattern.
* Name
* Price
* Description
* Icon

Table

Represents single interactive table
* Status (active/sleeping)
* Wallpaper (image)
* Seats – array of Seat
* Number

Design