RU Baked?

Kyle Rapps | Charles London | Nick Pieros
Scott Jeffrey | Justin Pilla | Ryan Smith

RU Baked Preview

RU Baked? Goals

- Develop an application that will run on mobile devices.
- Provide a convenient and quick interface for users to order goods.
- Application powered by a RESTful service
- A database to save user information in order to provide a more streamlined experience.
- Provide a digital venue for merchants to sell their goods.

RU Baked? Features

Buyer Features

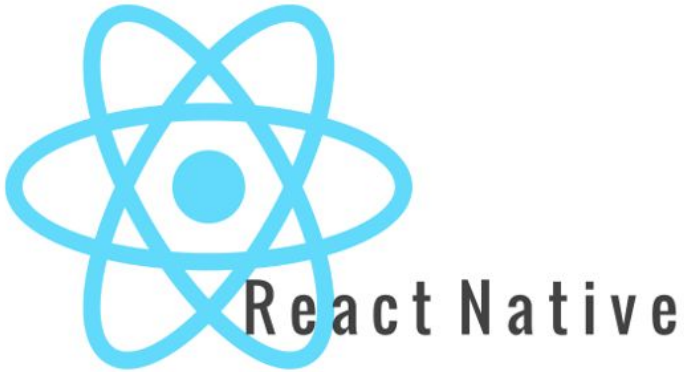
- SMS notification of orders
- Rating system of merchants to ensure quality of service
- All digital payment platform for ease of access

Merchant Features

- Hours easily modifiable to accommodate last minute schedule changes
- Uniform format for merchants ensures quality experience for customers
- Can set delivery options if available
- Build Your Own Menus

React Native

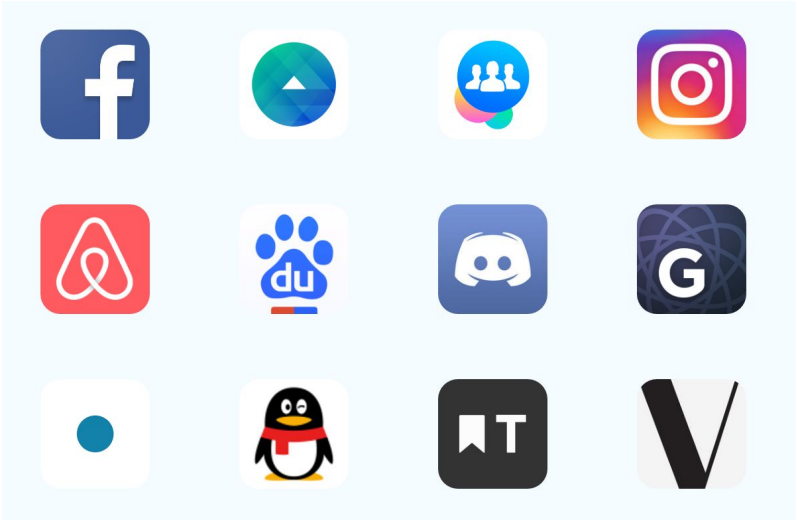
- Facebook Developed Technology
- Cross Platform Development using React
- Write Once, run on iOS and Android
- No need for recompiling new code, supports hot reloading
- Easy to implement Javascript (ES6)
- Ver. 0.35 in September, Ver 0.39 in December



React Native is a Growing Platform

Many popular services are moving to React-Native or are build on the platform including:

- Facebook
- Instagram
- Airbnb
- Discord
- Soundcloud Pulse
- And many more: https://goo.gl/rNUi09



Cross Platform Development with React Native:

Below is a sample explaining the different ways components will be interpreted by Android or iOS

```
import React, { Component } from 'react';
import { Image, ScrollView, Text } from 'react-native';

class AwkwardScrollingImageWithText extends Component {
  render() {
    return (
      <ScrollView>
        <Image source={{uri: 'https://i.chzbgr.com/full/7345954048/h7E2C65F9/'}} />
        <Text>
          On iOS, a React Native ScrollView uses a native UIScrollView.
          On Android, it uses a native ScrollView.

          On iOS, a React Native Image uses a native UIImageView.
          On Android, it uses a native ImageView.

          React Native wraps the fundamental native components, giving you
          the performance of a native app, plus the clean design of React.
        </Text>
      </ScrollView>
    );
  }}
```

iOS vs Android Development

Android development:

- Nuclide IDE (very buggery, not recommended)
- Any JavaScript editor

iOS development:

- Deco IDE (recommended)
- Any JavaScript editor

Run Android app:

- react-native run-android
- Android Studio

Run iOS

- react-native run-ios
- Xcode

RU Baked? Navigation Example

```
_navigateRegistration(name, type){
   this.props.navigator.push({
     component: RegisterScene,
     passProps: {
       name: name      },
     type: type
   })
```

RU Baked? Render Example

```
class InformationInput extends Component{
 constructor(props){
   super(props);
   this.state = {text: ''};
 }
 render(){
   return (
         <TextInput style = {styles.textinput}
           onChangeText = { (text) => this.setState({text})}
           onSubmitEditing = {dismissKeyboard}
           secureTextEntry = {this.props.secure}
          />
       );
```

NPM

Package manager for Javascript, required for React Native

Must run ‘cd /.../PROJECT && npm install’ on any new project

Npmjs.com hosts a wide variety of packages and libraries available for download

Contributed by users

Installing new packages is as simple as:

1. cd /.../PROJECT && npm install package-name
2. Import package within JavaScript file

SMS Integrations with Twilio:

Twilio

- Cloud Communications Platform
- Convenient & Easy to Implement RESTful API
- Send & Receive SMS messages with code
- Utilized by merchants when an order is place

Twilio Sample Call

```
client.messages.create(body=sms_str, to=user_phone.replace('-', ''), from_="+19083325066")
client.messages.create(body=sms_str, to=phone.replace('-', ''), from_="+19083325066")
return self.make_response_from_sqlalchemy(new_order), 201  # HTTP 201 CREATED
```

Backend Implementation with Python:

Flask

- Python Microframework for Web Development
- Easy way to implement RESTful API
- Simple way to Interact with Database with SQLAlchemy

SQL Alchemy

- SQL toolkit allows for full use of SQL power in simple Python adaptation
- Full suite of well known enterprise-level persistence patterns
- Object-Relational Mapper(ORM) allows building of Database Schema and object models in an entirely decoupled fashion

SQL Alchemy + Flask Mapping Example Code

```
response_to_sql_map = {'id': Store.store_id,
        'merchant': Store.merchant_id,
        'name': Store.store_name,
        'address': Store.store_address,
        'offline': Store.store_offline,
        'open': Store.store_open_time,
        'close': Store.store_close_time,
        'delivery': Store.store_delivery,
        'phone': Store.store_phone
        }
```

```
def post(self):
    # Creates a Store in the db. Do not pass in a store_id, the DB created this!
    parser = reqparse.RequestParser()
    parser.add_argument('merchant', type=int)
    parser.add_argument('name', type=str)
    parser.add_argument('offline', type=bool)
    parser.add_argument('address', type=str)
    parser.add_argument('phone', type=str)
    parser.add_argument('open', type=str)
    parser.add_argument('close', type=str)
    parser.add_argument('delivery', type=int)
    args = parser.parse_args(strict=True)
    new_store = self.update_sqlachemy_object(Store(), args)
    db.session.add(new_store)
    db.session.commit()
    return self.make_response_from_sqlalchemy(new_store), 201  # HTTP 201 CREATED
```

Flask API Code Sample

/stores/<int:store_id>

```
def get(self, store_id=None): # Get a store from the DB
    if store_id is None:
        store_info = Store.query.all()
    else:
        store_info = Store.query.get(store_id)
    if store_info is None or store_info == []: # Could not find that persons id
        return {'status': 400, 'message': 'store_id provided does not exist'}, 400
    return self.make_response_from_sqlalchemy(store_info), 200
```

http://127.0.0.1:5000/stores/3


```
{
  "address": "123 Ocean Avenue",
  "close": "06:01 PM",
  "delivery": 0,
  "id": 3,
  "merchant": 2,
  "name": "Rob's Bakery",
  "offline": false,
  "open": "06:00 AM",
  "phone": "234-432-2343"
}
```

SQL Alchemy Example

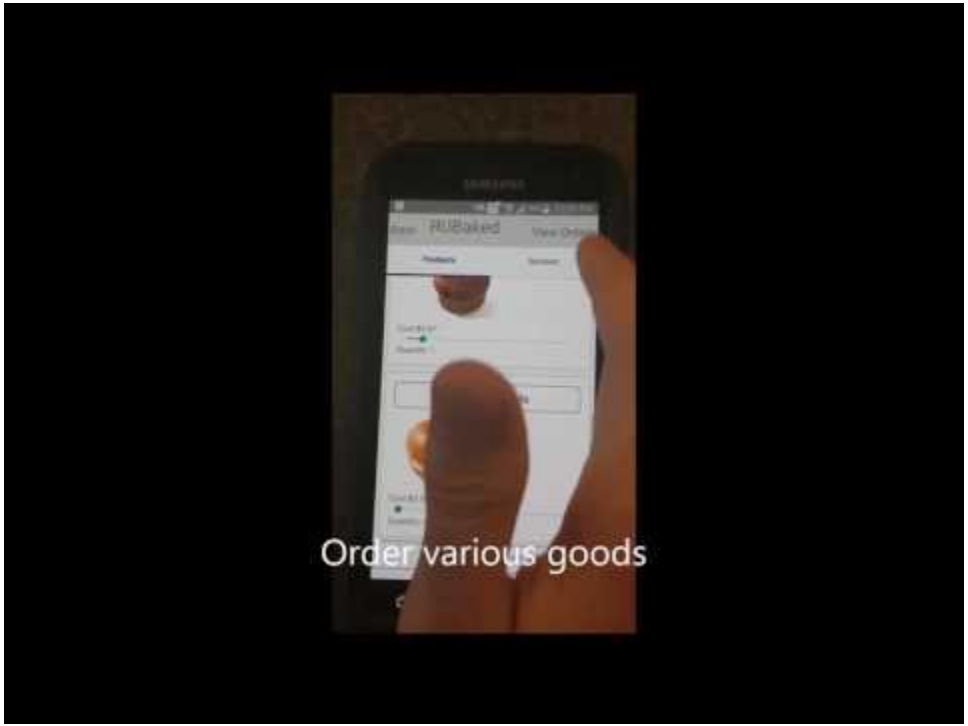
/stores/<int:store_id>/unfulfilled-orders/

```
def get(self, store_id):

    product_ids = Product.query.filter(Product.store_id == store_id).with_entities(Product.product_id)

    order_ids = OrderContent.query.filter(OrderContent.product_id.in_(product_ids)).with_entities(OrderContent.order_id)

    store_order_info = Order.query.filter(Order.order_id.in_(order_ids)).filter(Order.status_id <= 2).all()

    if store_order_info is None or not store_order_info:

        return {'status': 400, 'message': 'No orders are associated with the store id were found.'}, 400

    return self.make_response_from_sqlalchemy(store_order_info), 200
```

http://127.0.0.1:5000/stores/1/unfulfiled-orders


```
[{
    "address": "123 gumball street",
    "date": "2016-11-10",
    "id": 1,
    "status": 1,
    "user": 1
  },
  {
    "address": "123 nowhere street",
    "date": "2016-11-13",
    "id": 5,
    "status": 1,
    "user": 1
  }]
```

App Demo



Challenges

- React-Native is a currently in-development project - Versions are continuously being rolled out with major changes
- Typical development done in OSX or Linux environments making Windows environment tools sparse and difficult to set up.
- Endpoints for API were difficult to determine due to the open-ended aspect of the features - Can always make more!

What We Learned

- Platforms are difficult to work on when they are updated continuously throughout the development process
- Testing is extremely important for applications interfacing directly with a user
- Mobile development requires careful planning of both backend services AND UI features in order to build a coherent application
- RESTful API services are very powerful and useful tools

Questions?