10 Opinions on how you should get started with server side Swift

Who Am I?

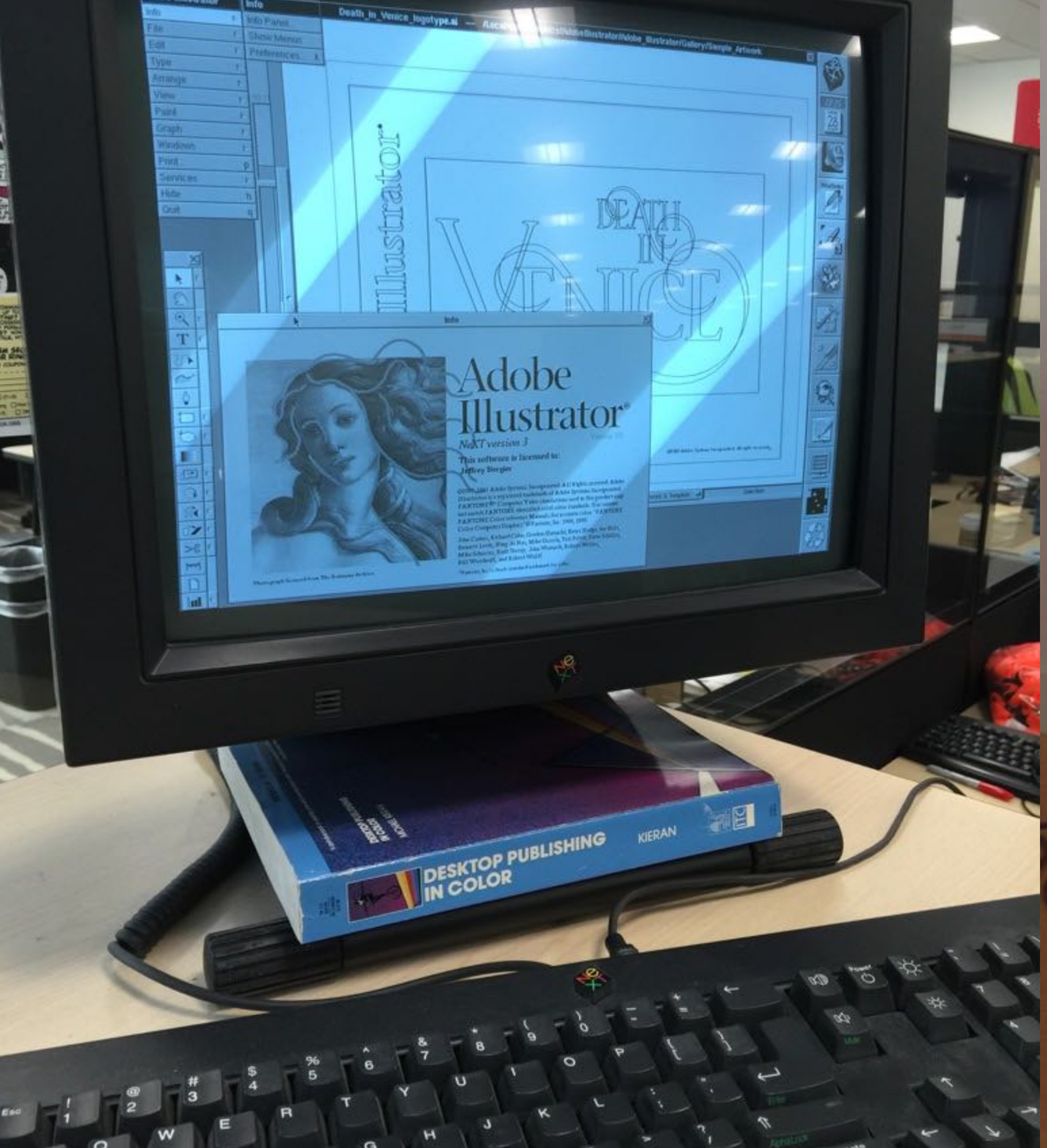
- Jeffrey Bergier
  - iOS Developer @ Topology Eyewear
  - UX Designer in a previous life
- twitter: @jeffburg web: jeffburg.com
- Slides
  - http://bit.ly/2tg40yY
- Previous Talk @SLUG
  - http://bit.ly/2tKodjO

A quick tangent

I’m leaving

(And I have some amazing classic Macs that need a loving home)


Mac OS X
Mac OS X


Adobe
Illustrator
NeXT version 3
DESKTOP PUBLISHING
IN COLOR
KIERAN

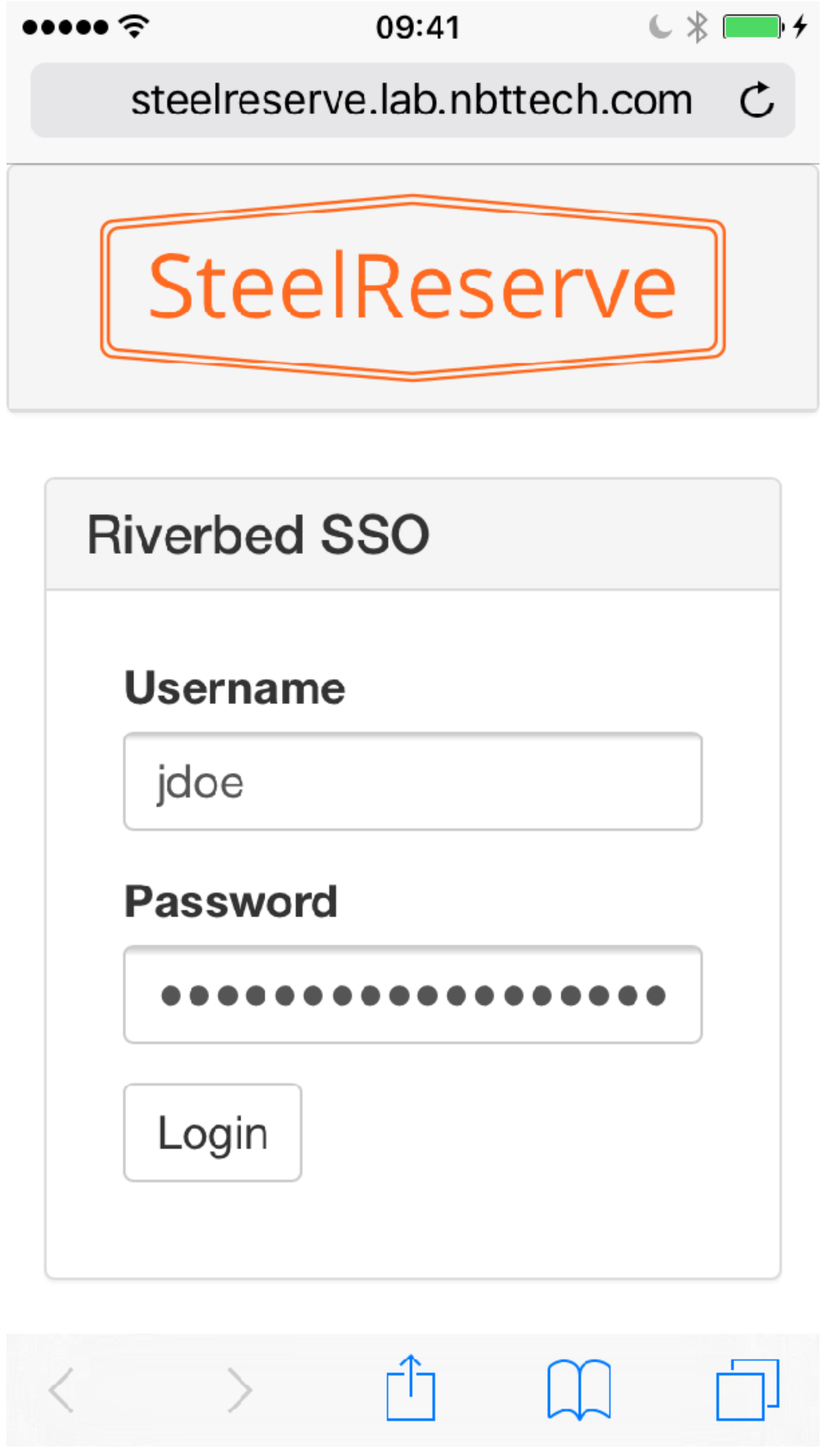
End Tangent

How did I get these opinions?

- My iOS development background shaping my opinions on web development
- I built an internal production app at work that made more sense as a web app
  - I built in Python first
  - I disliked Python so much I learned Swift on Linux 😊
    - And Javascript 😔

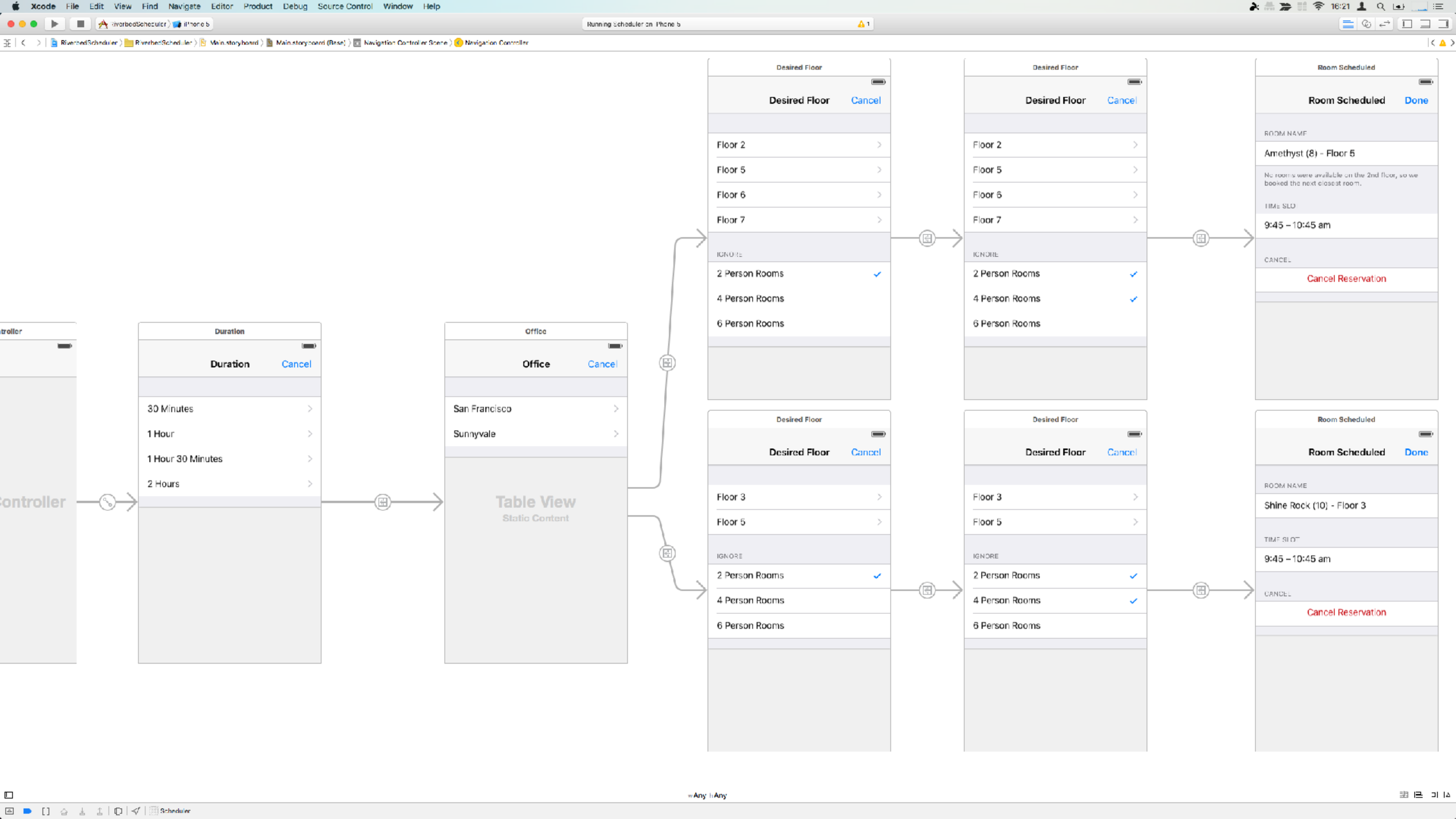
What is the App?

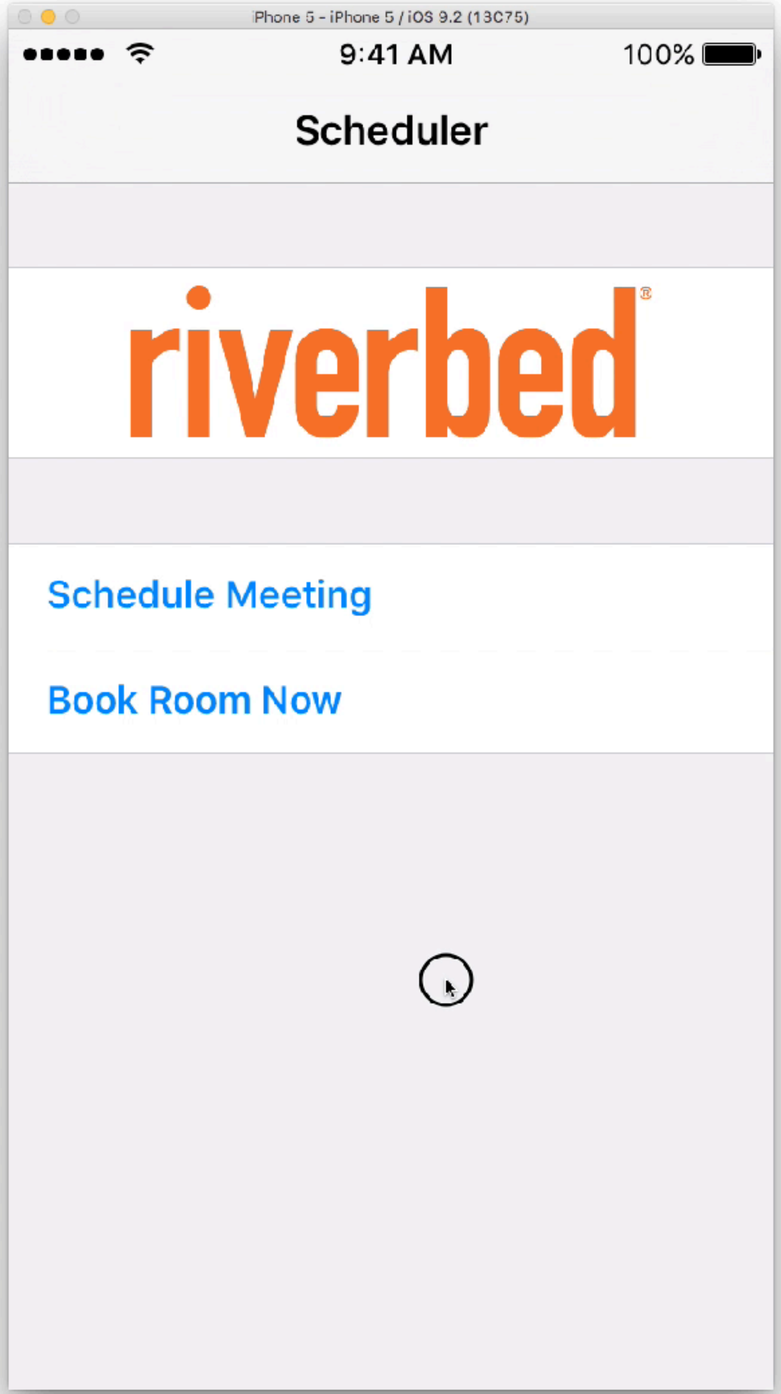
- A super quick way to reserve a conference room w/o Outlook


09:41
steelreserve.lab.nbttech.com
SteelReserve
Riverbed SSO
Username
jdoe
Password
Login

What is the App?

- A super quick way to reserve a conference room w/o Outlook
- Started as a storyboard prototype


Duration
Duration
Cancel
30 Minutes
1 Hour
1 Hour 30 Minutes
2 Hours
Office
Office
Cancel
San Francisco
Sunnyvale
Table View
Static Content
Desired Floor
Desired Floor
Cancel
Floor 2
Floor 5
Floor 6
Floor 7
IGNORE
2 Person Rooms
4 Person Rooms
6 Person Rooms
Desired Floor
Desired Floor
Cancel
Floor 3
Floor 5
IGNORE
2 Person Rooms
4 Person Rooms
6 Person Rooms
Room Scheduled
Room Scheduled
Done
ROOM NAME
Amethyst (8) - Floor 5
TIME SLOT
9:45 – 10:45 am
CANCEL
Cancel Reservation
Room Scheduled
Room Scheduled
Done
ROOM NAME
Shine Rock (10) - Floor 3
TIME SLOT
9:45 – 10:45 am
CANCEL
Cancel Reservation


Phone 5 - iPhone 5 / iOS 9.2 (13C75)
9:41 AM
100%
Scheduler
riverbed
Schedule Meeting
Book Room Now

What is the App?

- A super quick way to reserve a conference room w/o Outlook
- Started as a storyboard prototype
- Turned into a Swift playground to test Exchange ‘EWS’ API

GetUserAvailability request example: Get availability information

The following example of a **GetUserAvailability** operation request shows how to get detailed availability information for two users in the Pacific Time time zone.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
  <soap:Body>
    <GetUserAvailabilityRequest xmlns="http://schemas.microsoft.com/exchange/services/2006/messa
                xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
      <t:TimeZone xmlns="http://schemas.microsoft.com/exchange/services/2006/types">
        <Bias>480</Bias>
        <StandardTime>
          <Bias>0</Bias>
          <Time>02:00:00</Time>
          <DayOrder>5</DayOrder>
          <Month>10</Month>
          <DayOfWeek>Sunday</DayOfWeek>
        </StandardTime>
        <DaylightTime>
          <Bias>-60</Bias>
          <Time>02:00:00</Time>
          <DayOrder>1</DayOrder>
```

GetUserAvailability request example: Get availability information

The following example of a **GetUserAvailability** operation request shows how to get detailed availability information for two users in the Pacific Time time zone.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
  <soap:Body>
    <GetUserAvailabilityRequest xmlns="http://schemas.microsoft.com/exchange/services/2006/messa
                xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
      <t:TimeZone xmlns="http://schemas.microsoft.com/exchange/services/2006/types">
        <Bias>480</Bias>
        <StandardTime>
          <Bias>0</Bias>
          <Time>02:00:00</Time>
          <DayOrder>5</DayOrder>
          <Month>10</Month>
          <DayOfWeek>Sunday</DayOfWeek>
        </StandardTime>
        <DaylightTime>
          <Bias>-60</Bias>
          <Time>02:00:00</Time>
          <DayOrder>1</DayOrder>
          <Month>4</Month>
          <DayOfWeek>Sunday</DayOfWeek>
        </DaylightTime>
      </t:TimeZone>
      <MailboxDataArray>
        <t:MailboxData>
          <t:Email>
            <t:Address>user1@example.com</t:Address>
          </t:Email>
          <t:AttendeeType>Required</t:AttendeeType>
          <t:ExcludeConflicts>false</t:ExcludeConflicts>
        </t:MailboxData>
        <t:MailboxData>
          <t:Email>
            <t:Address>user2@example.com</t:Address>
          </t:Email>
          <t:AttendeeType>Required</t:AttendeeType>
          <t:ExcludeConflicts>false</t:ExcludeConflicts>
        </t:MailboxData>
      </MailboxDataArray>
      <t:FreeBusyViewOptions>
        <t:TimeWindow>
          <t:StartTime>2006-10-16T00:00:00</t:StartTime>
          <t:EndTime>2006-10-16T23:59:59</t:EndTime>
        </t:TimeWindow>
        <t:MergedFreeBusyIntervalInMinutes>60</t:MergedFreeBusyIntervalInMinutes>
        <t:RequestedView>DetailedMerged</t:RequestedView>
      </t:FreeBusyViewOptions>
    </GetUserAvailabilityRequest>
  </soap:Body>
</soap:Envelope>
```

What is the App?

- A super quick way to reserve a conference room w/o Outlook
- Started as a storyboard prototype
- Turned into a Swift playground to test Exchange ‘EWS’ API
- First written as a Python web app during work Hackathon

DOWNLOAD DOCUMENTATION COMMUNITY DEVELOPMENT

CherryPy

A Minimalist Python Web Framework

CHERRYPY IS AS EASY AS...

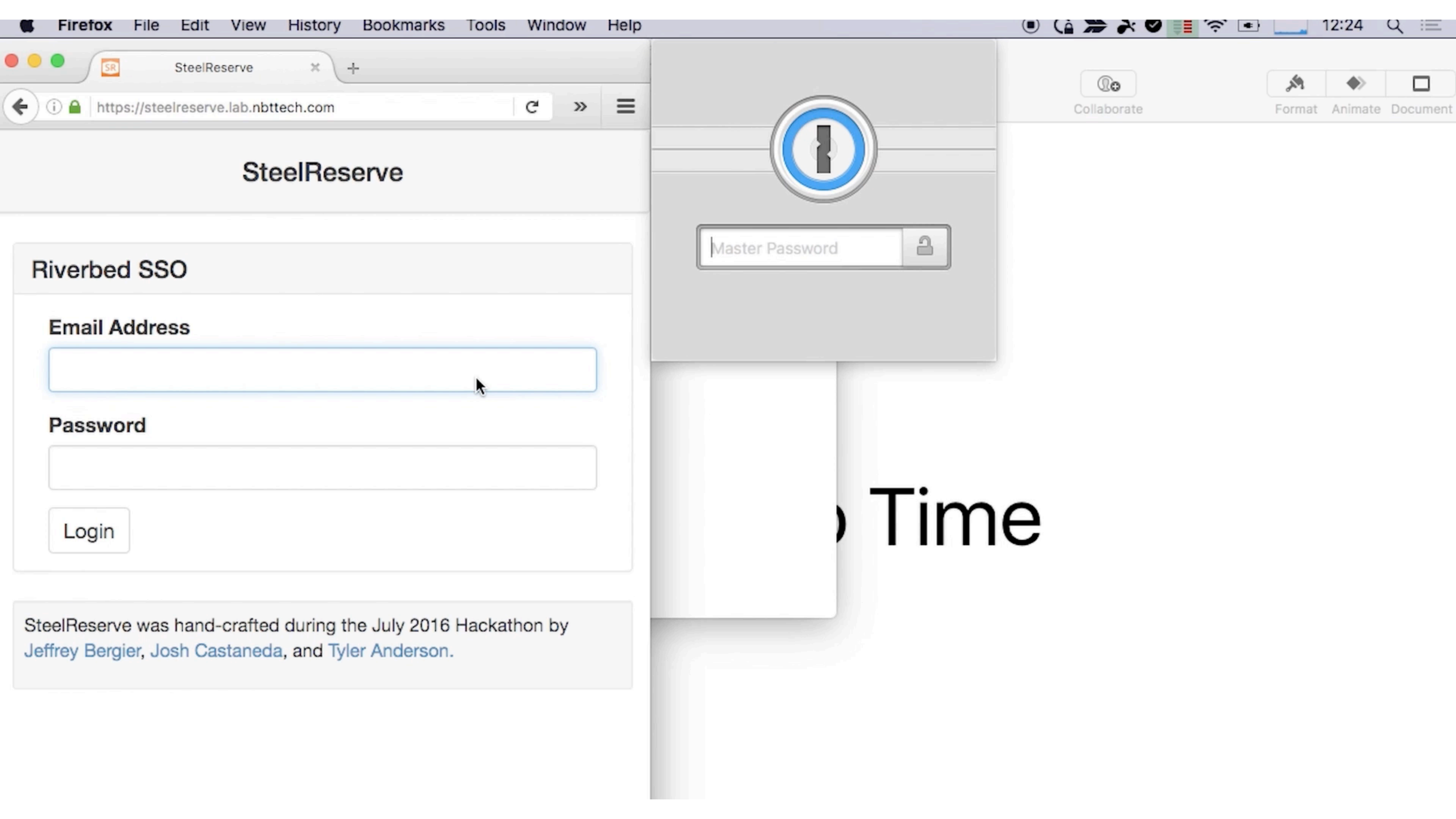
```
import cherrypy

class HelloWorld(object):
    def index(self):
        return "Hello World!"
    index.exposed = True

cherrypy.quickstart(HelloWorld())
```

What is the App?

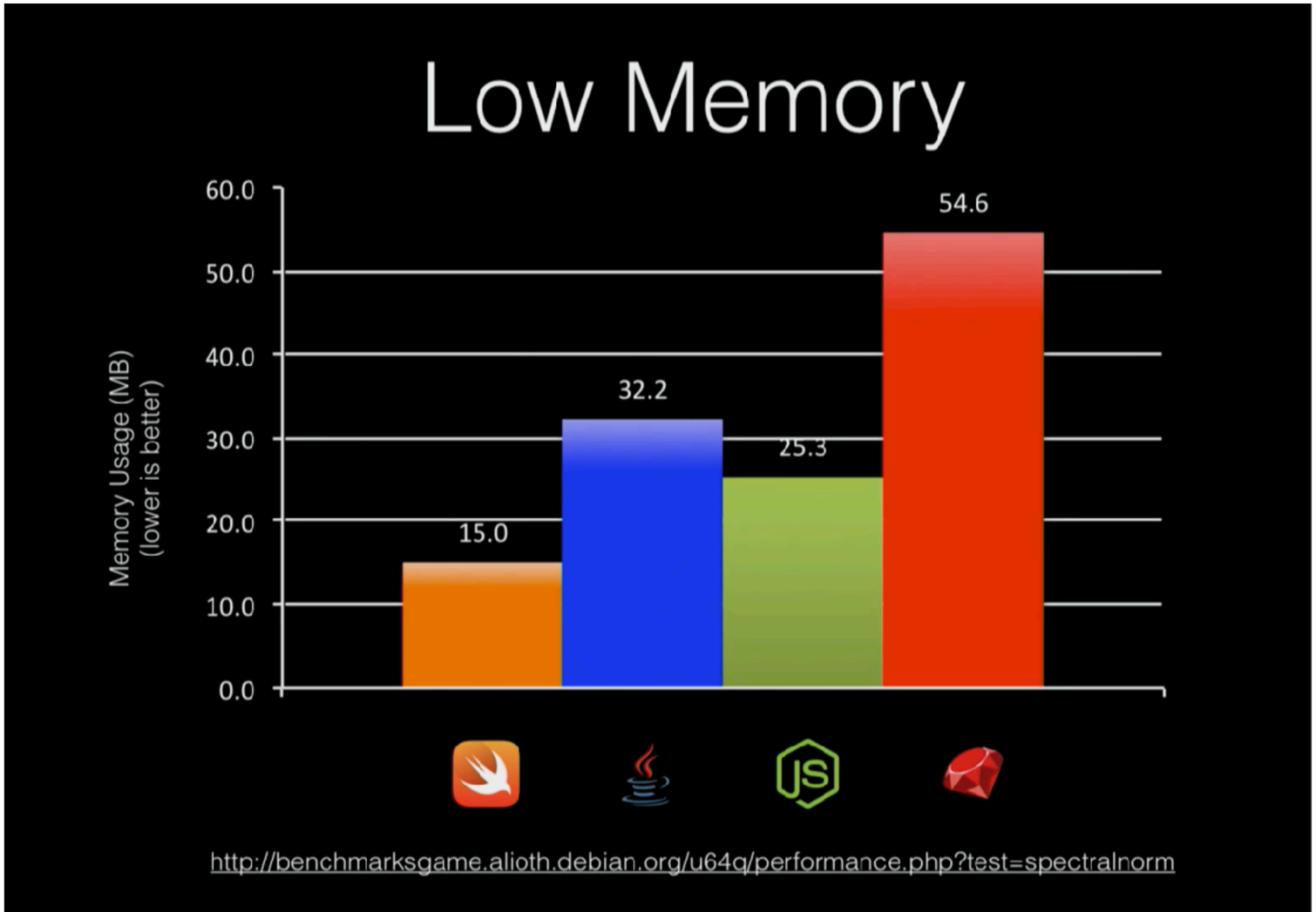
- A super quick way to reserve a conference room w/o Outlook
- Started as a storyboard prototype
- Turned into a Swift playground to test Exchange ‘EWS’ API
- First executed as a Python web app during work Hackathon
- Rewritten in Swift and Javascript in my spare time

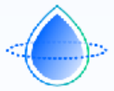

Firefox File Edit View History Bookmarks Tools Window Help
12:24
SteelReserve
https://steelreserve.lab.nbttech.com
Collaborate
Format
Animate
Document
SteelReserve
Master Password
Riverbed SSO
Email Address
Password
Login
SteelReserve was hand-crafted during the July 2016 Hackathon by Jeffrey Bergier, Josh Castaneda, and Tyler Anderson.
Time

1. Has excellent tooling

- Great debugging on Mac
- Swift has strong typing
- Easy testing on Mac and Linux
- Swift strict typing is unique

2. Is likely to save you money





Standard Droplets

Easy-to-deploy and resizable SSD-based virtual machines that are billed hourly. Standard Droplets offer the right amount of RAM, CPU, and local storage space needed to get applications off the ground. Learn More

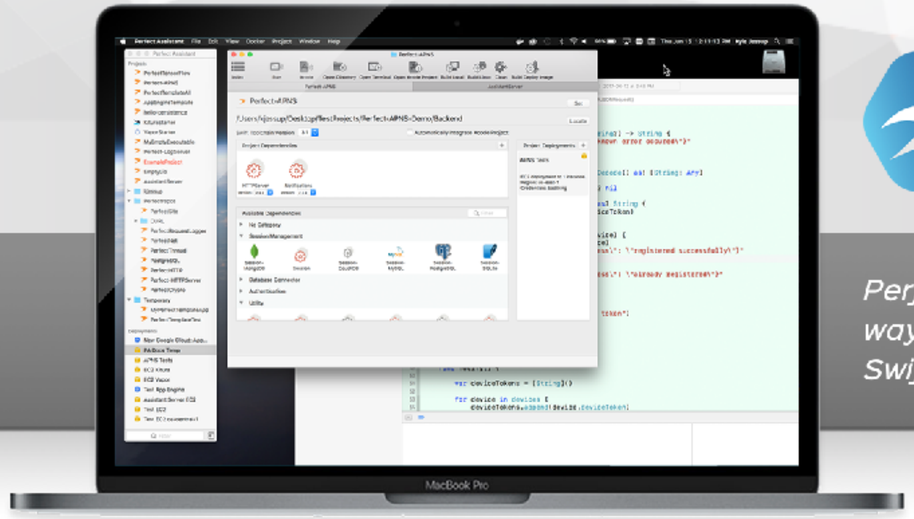
$5 /mo	$10 /mo	$20 /mo	$40 /mo	$80 /mo	$160 /mo
$0.007/hr	$0.015/hr	$0.03/hr	$0.06/hr	$0.119/hr	$0.238/hr
512MB Memory	1GB Memory	2GB Memory	4GB Memory	8GB Memory	16GB Memory
1 vCPU	1 vCPU	2 vCPU	2 vCPU	4 vCPU	8 vCPU
20GB SSD Disk	30GB SSD Disk	40GB SSD Disk	60GB SSD Disk	80GB SSD Disk	160GB SSD Disk
1TB Transfer	2TB Transfer	3TB Transfer	4TB Transfer	5TB Transfer	6TB Transfer
		Get Started			

3. Pick the HTTP framework you like

- They’re all fairly similar and have similar functionality
- I chose Perfect
- Vapor also seems to be popular
- IBM Kitura is intriguing because it has the backing of IBM

4. Start with Perfect Assistant

- Its a glorified wrapper around terminal commands
  - I’m totally OK with that
- Gets dependencies
- Builds Xcode projects
- Does docker magic



Introducing the
perfect assistant 2.0

Perfect Assistant is the quickest & smartest way to build & deploy your server-side Swift projects.

Free Download ›› Documentation ››

Server-side Swift at your fingertips

The macOS companion application is a set of convenience tools designed to help Server Side Swift developers start, manage, compile, test and prepare for linux deployment more easily. From those expanding into backend Swift development for the first time to seasoned engineers working on business critical applications, the Perfect Assistant will facilitate your workflow and get your Swift projects deployed faster and error-free.

It's in the details

Demo

- *Make new project with Assistant*
- *Show router code / load page in Safari*
- *Make random number*
- *Set breakpoint*
- *Save random number as state*

5. Linux Foundation not always 💯

- Sometimes runtime crashes (when running on Linux)
- Sometimes compile errors (when compiling on Linux)
  - Perfect assistant helps with this
- Its usually easy to find lightweight wrappers around common C libraries used in the Linux world
  - E.g. The XML parsing was done with a PerfectXML library that wraps libxml2

Demo

- *Make new project with Assistant*
- *Make random number*
- *See compile error*
- *Show run script build phase*

6. Force Unwrapping is Bad

- In Cocoa/Touch development, force unwrapping nil crashes one person's app
  - In a Swift server, ALL your users rely on this app not crashing
  - This changes the calculus quite a bit

7. Treat the browser as a client

- This fits more in-line with how we think of our iOS / Mac apps
- You're going to have to learn Javascript no matter what
  - Might as well keep your swift code Swifty by just sending/receiving JSON to/from the browser client
  - Multiline String literals will help clean up this code, but still, we don't want to be sending HTML directly back to the browser

```
struct ChooseBuilding: UIJSONExportable {

    static let vPageTitle = "Choose Building"
    static let vCurrentStep = CurrentStep.chooseBuilding

    var organization: Organization
    var duration: Duration

    func jsonString() -> String {
        let panelRows = self.organization.buildings.map() { building -> [String : Any] in
            return [
                UIPage.kPanelRowPrimaryText : building.name,
                UIPage.kPanelRowSelectionID : building.name
            ]
        }
        let panels: [[String : Any]] =  [
            [
                UIPage.kPanelTitle : organization.name,
                UIPage.kPanelRows : panelRows,
                UIPage.kFormData : [
                    UIPage.kFormDataCurrentStep : type(of: self).vCurrentStep.rawValue,
                    UIPage.kFormDataDuration : "\(self.duration.rawValue)"
                ]
            ]
        ]
        let dictionary: [String : Any] = [
            UIPage.kPageTitle : type(of: self).vPageTitle,
            UIPage.kPagePanels : panels,
            UIPage.kPageWells : [UIPage.wellText]
        ]
        let jsonString = try? dictionary.jsonEncodedString()
        return jsonString ?? InvalidData.ToBrowser.rawJSONString
    }
}
```

8. Don’t be lured by JS craziness

- Javascript can be a scary world
  - React, Redux, ES6, Babel, Angular, Vue, etc
- Javascript can be simple
  - You can build your own ‘Virtual DOM’ without React
  - If you squint hard enough it looks like Swift

```
function Message(messageObject) {

  function isValidMessageKind(input) {
    if (isString(input)) {
      var lowerInput = input.toLowerCase();
      if (lowerInput == "success" || lowerInput == "info"  || lowerInput == "warning" || lowerInput == "danger") {
        return true;
      } else {
        return false;
      }
    } else {
      return false;
    }
  }

  function throwIfNotValidMessageKind(input) {
    if (isValidMessageKind(input)) {
      return true;
    } else {
      throw "Exception: throwIfNotValidMessageKind: message kind was not valid: (" + input + ")";
      return false;
    }
  }

  var kind = messageObject["kind"];
  var text = messageObject["text"];
  throwIfNotStringOrNumber(text); throwIfNotValidMessageKind(kind);

  this.kind = kind;
  this.text = text;

  this.generateDomElement = function() {
    var containerElement = document.createElement("div");
    containerElement.className = "container-fluid";
    var alertElement = document.createElement("div");
    alertElement.className = "alert alert-" + this.kind + " " + "text-center";
    alertElement.setAttribute("role", "alert");
    var h5Element = document.createElement("h5");

    h5Element.innerHTML = this.text;
    alertElement.appendChild(h5Element);
    containerElement.appendChild(alertElement);

    return containerElement;
  };
}
```

9. Don’t fret about RESTfulness

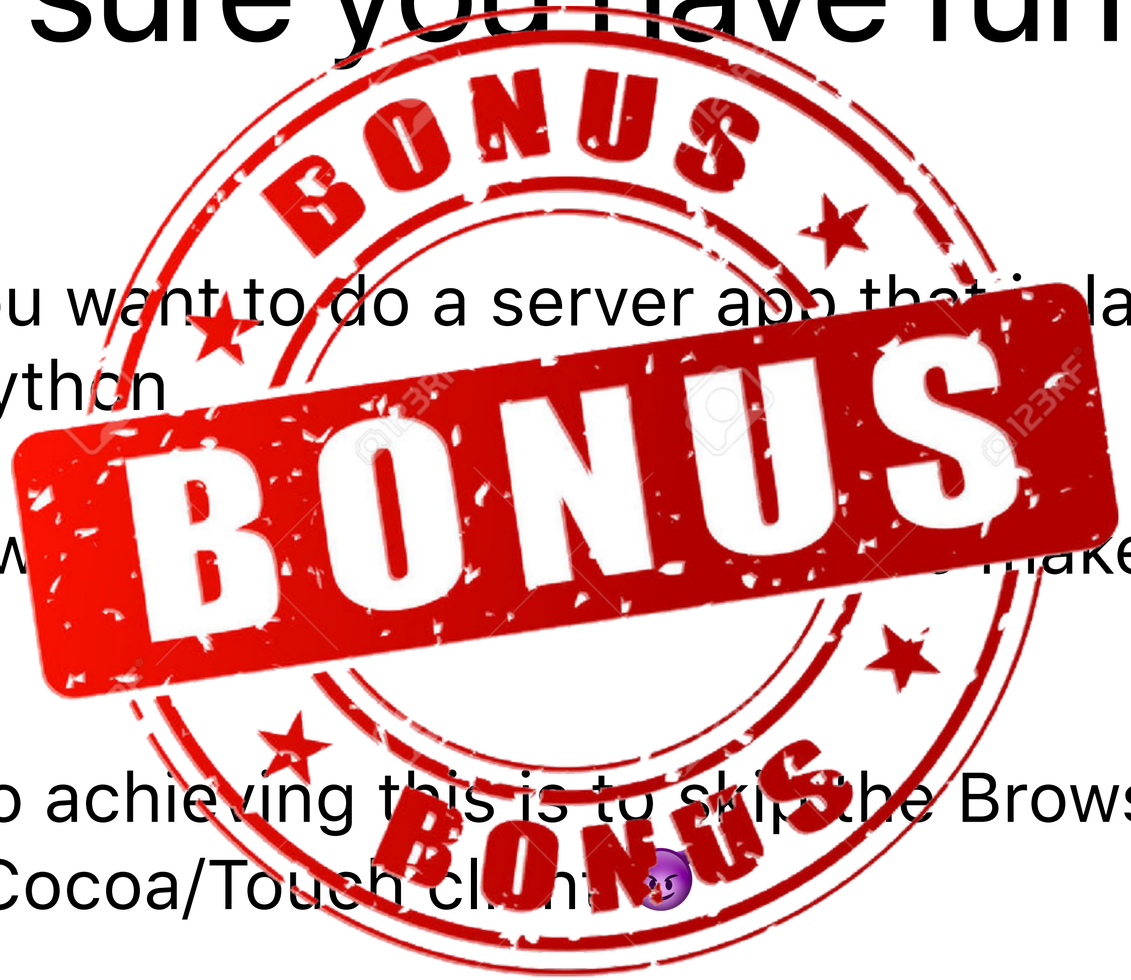
- Well designed REST API’s are great when:
  - You don’t know what the client app looks like or does
  - You want to expose your database in a restricted way that is authenticated and rate limited
- Well designed REST API’s are not great when:
  - Your app has to make more than one request in response to a user action

10. Take security seriously

- This is regardless of the server language. But seriously people
- SteelReserve was taking enterprise credentials
  - I encrypted them and then only stored them in memory
  - I stored the 2 decryption keys in browser cookies
  - If cookies were missing, then it treated the user as logged out
- If you can’t access user data, then malicious people can’t either

11. Make sure you have fun!

- Seriously, if you want to do a server app that is lame and boring, use Ruby or Python
- We're doing Sw[illegible] make sure you're liking it.
  - A shortcut to achieving this is to skip the Browser client and just build a Cocoa/Touch cli[illegible]



Thank You!

Q&A

@jeffburg

http://bit.ly/2tg40yY