Going Universal
iPad Lessons Learned at ING

Age Mooij
Andrew Snare

Welkom
bij de ING

Xebia
### Betaalrekeningen

<table>
<thead>
<tr>
<th>Naam</th>
<th>Rekeningnummer</th>
<th>Betalingskenmerk</th>
<th>Mededeling</th>
<th>Saldo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mw A Wirdum</td>
<td>764757</td>
<td>Acceptgiro</td>
<td>So long, and thanks for all the fish</td>
<td>€138,30</td>
</tr>
</tbody>
</table>

**Ontvanger**

<table>
<thead>
<tr>
<th>Naam</th>
<th>Rekeningnummer</th>
<th>Betalingskenmerk</th>
<th>Mededeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Douglas Adams</td>
<td>42</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Betaalrekeningen**

<table>
<thead>
<tr>
<th>Datum</th>
<th>Betaling Type</th>
<th>Betalingdetails</th>
<th>Betalingbedrag</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 24, 2012</td>
<td>Transferred to Douglas Adams</td>
<td></td>
<td>-€42,00</td>
</tr>
<tr>
<td>May 23, 2012</td>
<td>Random transfer of -228000</td>
<td>(cents) to account number</td>
<td>-€228,00</td>
</tr>
<tr>
<td>May 22, 2012</td>
<td>Random transfer of 90000</td>
<td>(cents) from account number</td>
<td>+€900,00</td>
</tr>
<tr>
<td>May 21, 2012</td>
<td>Random transfer of -10800</td>
<td>(cents) to account number</td>
<td>-€108,00</td>
</tr>
<tr>
<td>May 20, 2012</td>
<td>Random transfer of -70000</td>
<td>(cents) to account number</td>
<td>-€700,00</td>
</tr>
</tbody>
</table>
Background

Things were going well...
The Challenge

• Strong focus on UI look and feel
• Heavily customizedstyled UI
• Complex security requirements
• Coming from behind the market
• There was an existing app based on the MSN chat-bot: universally embarrassing.
The Process

- A pilot scrum project at ING
- 2 week sprints
- Started with one team, grew to 3 teams (iOS, Android, backend)
- Dedicated UX people
- Lots of internal stakeholders!
That's great, but will it also work on my iPad?
– Really Important Manager
“That’s great, but will it also work on my iPad”?  
– Really Important Manager
Plan A

BigPhone™ — Just Scale Up
Plan A: Results

• Took 2 weeks
• Developer hated being isolated
• Did not meet UX requirements
• It let us estimate the real effort.
For a high quality app, you really need to create a separate iPad UI.*

*Duh!
Plan B

A Tale of Two Teams
Plan B: Results

• Lasted half a sprint

• Not sustainable due to merging overhead

• New team couldn’t help but slow the original team: they needed help.
Lesson Learned

This was a really bad idea!
Plan C

newTeam

oldTeam

One Big Happy Family
Plan C

bigTeam = zip(newTeam, oldTeam)

One Big Happy Family
Plan C Result: 1.0

- This worked!
- Some pairs worked on new functionality, some on existing screens.
- Took 6 weeks, instead of 3
- Business very very very happy.
Lesson Learned

Pair-programming is a fantastic tool for scaling a team up quickly.
So what did we really need to do?
Splitting the Screen
Splitting the Screen

- Oh, the pain: UISplitViewController
  - It must be the root, but our starting grid is.
- View controllers could no longer be in charge of the entire screen.
- Handling teardown on errors was very very hard.

- Keeping the two sides synchronized:
  - Current balance
  - Refresh detail triggers refresh of master
  - Refresh master triggers selection, which forces refresh of the detail.
Lessons Learned

Custom UI and navigation flow is expensive.

View controllers should use a shared model.
Modal Popovers

Reuse existing view controllers inside model dialogs.

This saved us a lot of time.

Here be dragons:

- `(UIViewController *)parentViewController;
- `(UIViewController *)presentingViewController;`
if (iPad) ...

- Sometimes we could get away with a more generic solution:

```objectivec
view.autoresizingMask = UIViewAutoresizingFlexibleWidth | ...
```

- We used a lot of if-iPad blocks:

```objectivec
if ([UIDevice currentDevice].userInterfaceIdiom == UIUserInterfaceIdiomPad) {
    ... }
```

- We used subclassing:

```objectivec
@interface FooViewController_iPad : FooViewController
@end
```
Rotation

- Rotation was originally omitted to save time.

*Not an acceptable compromise on the iPad: people expect all orientations to work.*

- Lots of small changes required, mainly with autoresizing behaviour.
Technical Debt

Cleaning up the mess…
Conscious Compromise

- Decision to trade off technical debt for meeting the deadlines.
- Paying off the debt is still an ongoing effort.
Technical debt can be acceptable…

BUT

…be aware it may never be paid off.
Don’t make the compromise if this is unacceptable.
Next time?

Just start with universal from the start…?

Stay agile, but anticipate going universal:

• For example, UI/UX design that is easier to scale.
Where are we now?

And what’s next?
A Success Story

> 1,000,000 downloads

Lots of feedback (about missing features)

Consistent 4.5 star rating

4 major releases
2 minor releases
About every 6-8 weeks
Sneak Preview
<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 23, 2012</td>
<td>Transferred to Douglas Adams</td>
<td>€42,00</td>
</tr>
<tr>
<td>May 23, 2012</td>
<td><em>Reservation</em></td>
<td></td>
</tr>
<tr>
<td>May 22, 2012</td>
<td>Random transfer of -63600 (cents) to account number 2361888</td>
<td>€636,00</td>
</tr>
<tr>
<td>May 21, 2012</td>
<td>Random transfer of -42400 (cents) to account number 1746914</td>
<td>€424,00</td>
</tr>
<tr>
<td>May 20, 2012</td>
<td>Random transfer of -18600 (cents) to account number 2270811</td>
<td>€186,00</td>
</tr>
<tr>
<td>May 19, 2012</td>
<td>Random transfer of -47100 (cents) to account number 4078985</td>
<td>€471,00</td>
</tr>
<tr>
<td>May 18, 2012</td>
<td>Random transfer of 28800 (cents) from account number 8169800</td>
<td>+€286,00</td>
</tr>
<tr>
<td>May 18, 2012</td>
<td>Random transfer of -86000 (cents) to account number 2366683</td>
<td>-€868,00</td>
</tr>
<tr>
<td>May 17, 2012</td>
<td>Random transfer of -54300 (cents) to account number 7664151</td>
<td>-€543,00</td>
</tr>
</tbody>
</table>
Sneak Preview
One more time...

• For a high quality app, you really need to create a separate iPad UI.
• Bad idea: 2 teams, 1 product.
• Pair-programming is a fantastic tool for scaling a team up quickly.
• Custom UI and navigation flow is expensive to build and maintain.
• View controllers should use a shared model.
• Technical debt may never be paid off: don’t make the compromise if this is unacceptable.
Thank You!
Questions?

Age Mooij (@agemooij)
Andrew Snare (@asnare)