Testing the API behind a mobile app

Tutorial
Marc van ’t Veer

Content

• What is an API
• Why use an API
• How to use an API

• How-to test an API
• Exercises
  – Group exercises (learning concepts – 7)
  – Individual exercises (apply concepts – 4)
• Lessons learned
What is an API - Definition

• API, stands for application program interface:
  – Set of routines, protocols, tools and blocks for
    building applications (systems of systems) and so
    reducing complexity

• Definition within the context of this training:
  – An internet web service that opens systems and
    giving standard way of access to external users

“There’s an app for that.”
What is an API - API model

- Lego bricks are standard building blocks
- Blocks are a library to build
- Figure is showing which resources there are
- *Sales box* is an example of the possibilities
What is an API – Typical structure

- API is built upon existing internal architecture

Why use an API - example
Telecom companies provide usage details on their websites. App developers used screen scraping to get this data into their apps.

Resulting in:
- Misinterpretation of data
- Malformed data
- Telecom company gets blamed when the app provides wrong information
Why use an API - example

API provides the telecom companies:

• More control over the data

• Control over the content used in an app
  – Key needed to use the API
  – App developer needs to prove that it works correct before getting access

Why use an API

• Scaling development
  – *Enhance a program without editing itself*

• Converting competitors into partners
  – *Allow competitors to build on top of your product*

• Scaling market reach
  – *Open up new markets*

• Empowering users
  – *Allow people to use the product for things it wasn’t designed for*
How to use an API – Mashup

• A mashup is a web page or web application that uses and combines data, presentation or functionality from two or more sources to create new services

  – Easy, fast integration
  – Frequently using API
  – Produce enriched results from the raw source data

How to use an API – Demo of API’s

• Local examples
  – Command line on Windows or Linux
  – Local VB script with Windows API

• Web API examples
  – Overview of API’s: Programmableweb.com
  – Demo of WordPress blog API
How to use an API – Concepts

• Concepts of the API
  – Division of data and presentation
  – Root (Start URL, http or https)
  – Navigation
  – View of a command-line
  – Hierarchy
  – Resources
  – Account (private API)
  – Structure

How to use an API – Who uses it?

• Apps: Android, Iphone, Windows, Blackberry
• Internal organizations
• User via browser version of the API
• Social media: Facebook, twitter, etc.
• (internal/external) developers:
  – Documentation
  – Base line code for new App (with guidelines)
• Operations (monitoring)
Summary – Introduction to an API

• Public available via Internet
• No-frontend
• Interfaces
• API is part of an end-to-end solution
• Architecture / protocol focus
• External users / devices
• Test tools are needed
• **Main goal is: a generic API**

How-to test an API – start from scratch

1. Setup test case (REST in Goapi) → 16
   + Authentication (token, token/certificate) → 16
   + Output (XML/HTML/JSON) → 16
   + Static code testing → 16
2. Setup and pathing → 16
3. Setup test cases → 16
4. Setup test cases → 16
5. Setup test cases → 16
6. Setup test cases → 16
How to test an API – start from scratch

- **Architecture**
  - REST vs. SOAP
  - Stateless vs. Session
  - Media types vs. WSDL
    - text/xml or application/vendor specific
  - Verbs (GET) and resources
  - Headers and caching results

- **Tools**
  - SoapUI, Fiddler, new add-ons for Firefox

How to test an API – RESTful verbs

- Verbs are actions, actions like CRUD on database

- **POST:** Create a new data element
- **GET:** Read a data element
- **PUT:** Update a data element
- **DELETE:** Delete a data element
How to test an API – Headers

- HTTP header fields are components of the message header of requests and responses in the Hypertext Transfer Protocol (HTTP).
- They define the operating parameters of an HTTP transaction

- Request example:
  - Accept: text/plain
  - Cache-Control: no-cache

How to test an API – Cache

- A web browser stores web content for reuse.
- If the back button is pressed, the local cached version of a page is displayed instead of a new request being sent to the web server.

- Risks
  - Old data
  - Cached errors
  - Layered caching (app – API – server)
How to test an API - Typical risks

• Unknown integration
• Big variation of customer data
• No control of the E2E chain
• Load is unknown
• Wrong use of API
• Dynamic scope

How to test an API - Strategy

• Early integration test with complete infrastructure
  – Integration test and dog fooding during Development and System Testing
  – Multiple integration phases
  – Prototype app (on Acceptance environment)
How to test an API - Dogfooding

• Dogfooding: Don’t try to build the sort of API you think people will want, but use the public API’s your self everywhere

• Examples
  – All Google tools use their own public API’s
  – Windows 7 is build with windows 7
  – Twitter.com uses it’s own public API

How to test an API - Test approach

1. Development (with dogfooding)
2. System testing (API with stubs and SIT)
3. Integration testing (API with backend)
4. Acceptance testing (API with prototype app)
5. Production integration (in phases)
6. Regression testing (automated)
How to test an API - Test approach

- API’s are part of a big E2E chain
- Testing is only possible if a stub framework is available
- Stub, driver, mock
- Example of a stub framework
How to test an API - Mocks and stubs - 2

• Replacement of a reply from backend system

How to test an API - Mocks and stubs - 3

• If there is no connection to a backend system
• Re-routing an interface to a local tool like SoapUI
Group exercises

• Minimum of three implementations is needed for a quality API

• Learning
  – One implementation: it’s a trick
  – Two implementations: it’s a method
  – Three implementations: it’s your own way of working

Group exercises – learning concepts

• Exercise 1 – Lorum ipsum text
• Exercise 2 – Colour lovers and Color picker
• Exercise 3 – Authorization
• Exercise 4 – Status and error codes
• Exercise 5 – Header injections
• Exercise 6 – Business rules
• Exercise 7 – Current and future version
Exercise 1 – Lorum ipsum text

• Description
  – Generation of Lorum ipsum text

• Assignment
  – Get a number of lines of bacon ipsum text
  – Tool: Firefox

Exercise 1 details – Lorum ipsum text

• API
  – http://baconipsum.com/api/

• API documentation
  – Open two tabs
    • One with documentation: http://baconipsum.com/api/
    • One to enter commands like
      – ?type=meat-and-filler
Exercise 2 – Colour lovers and Color picker

• Description
  – API provide content in many different types, like generation of colors

• Assignment
  – Find a favorite color with colour lover API and validate the Hex-code with Color picker
  – Tool: Firefox

Exercise 2 details – Color lovers and Color picker

• API
  – http://www.colourlovers.com/api/colors?
  – http://www.colorpicker.com/

• API documentation
  – http://www.colourlovers.com/api/
Exercise 3 – Authorization

• Description
  – Difference between public and private API
  – Testing of functional security

• Assignment
  – Get access to API by using access tokens
  – Tools: Firefox

Exercise 3 details – Authorization

• API
  – http://openexchangerates.org/api/latest.json
  – Access key:
    • 66e62d8337b545ec9f0508c1215764b5

• API documentation
  • https://openexchangerates.org/documentation
  • https://openexchangerates.org/documentation#app-ids
Exercise 4 – Status and error codes

• Description
  – Explore what is a good and what is a bad request

• Assignment
  – Find status and error codes and rules to trigger them
  – Which error code is shown when (priority)
  – When do you expect an error code
  – Tools: Firefox + Firebug

Exercise 4 details – Status and error codes

• API
  – http://openexchangerates.org/api/latest.json?app_id=66e62d8337b545ec9f0508c1215764b5

• API documentation
  – https://openexchangerates.org/documentation
  – https://openexchangerates.org/documentation#errors
Exercise 5 – Header injections

• Description
  – Explore what header elements are

• Assignment
  – Compare results with and without a header elements
    • If-Modified-Since and If-None-Match
  – Tools: Firefox + Rest Client

Exercise 5 details - Header injections

• API
  – http://openexchangerates.org/api/latest.json?app_id=66e62d8337b545ec9f0508c1215764b5

• Documentation
  – https://openexchangerates.org/documentation#etag
Exercise 6 – Business rules

• Description
  – Compare output between different customer types. Which resource has which output?

• Assignment
  – Find output of a prepaid and a business customer
  – Find a invoice for a prepaid customer
  – Tool: Firefox

Exercise 6 details – Business rules

• API
  – [https://capi.t-mobile.nl/](https://capi.t-mobile.nl/)
  – Access key
    • {Bearer} with:
      – {StubbedResidentialPostPaid}
      – {StubbedBusinessPostPaid}
      – {StubbedResidentialPrePaid}

• Documentation
  – [https://capi.t-mobile.nl/documentation](https://capi.t-mobile.nl/documentation)
Exercise 7 – Versions of resources

• Description
  – Resources have a life cycle which is organized in versions for now and the future

• Assignment
  – Find a resource that is active, have multiple versions and is end of life
  – Tool: Firefox

Exercise 7 details - Versions of resources

• API
  – [https://capi.t-mobile.nl/](https://capi.t-mobile.nl/)
  – Access key
    • {Bearer} with:
      – {StubbedResidentialPostPaid}
      – {StubbedBusinessPostPaid}
      – {StubbedResidentialPrePaid}

• Documentation
  – [https://capi.t-mobile.nl/documentation](https://capi.t-mobile.nl/documentation)
Individual exercises – Apply concepts

• Exercise 8 - Design review
• Exercise 9 – Testing design techniques
• Exercise 10 – Explore test application
• Exercise 11 - Search of the defects

Test levels for testing an API

<table>
<thead>
<tr>
<th>Phase</th>
<th>Structure</th>
<th>Element</th>
<th>Test case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Development</td>
<td>Framework</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Check easy to use framework and how good it functions</td>
</tr>
<tr>
<td>2. System testing</td>
<td>Functionality with and without stubs</td>
<td>Test applications</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIT</td>
<td>No stubs</td>
<td>Integration with Middleware</td>
<td>Variation in customers, care plans, prices, contract types, history, options with no stubs</td>
</tr>
<tr>
<td>3. Integration testing</td>
<td>Internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Acceptance testing</td>
<td>Internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Production testing</td>
<td>Real customers</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Regression</td>
<td>Regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Test application**

- JSON API as plugin for WordPress
- CMS system
- Customizable and Open source
- Based on PHP language and MySQL database
- Also has mobile application for all platforms


---

**Test environment setup**

![Diagram showing WordPress server and test tool with API verification and request-reply process.]
IP address WordPress

Exercise 8 – Design review

• Description
  – An API design should give all information about how it operates

• Assignment
  – Use a checklist to review design of JSON API plugin
Exercise 8 details – Design review

• API
  – http://wordpress.org/extend/plugins/json-api/other_notes/
  – http://<local IP address>/documentation/json-api/

• Documentation
  – https://github.com/WhiteHouse/api-standards
  – http://<local IP address>/documentation/design-review-checklist/

Exercise 8 - Design review result

• Result of design review on JSON API plugin
Exercise 9 – Test design techniques

• Description
  – Normal test techniques are useful

• Assignment
  – Create test cases for the JSON API plugin of Wordpress
  – Use techniques like
    • State transition
    • Data combination / classification tree
    • Equivalence Class partitioning
    • Boundary Value analysis

Exercise 9 – Testing design techniques

• Use scrap paper to write test cases
• See example workout

• Documentation
Exercise 10 – Explore Test application

• Description
  – Before you can test the API you should first explore it

• Assignment
  – Open the JSON API and try it out

Exercise 10 – Explore Test application details

• API
  – http://<local IP address>/?json=

• Documentation
  • http://wordpress.org/extend/plugins/json-api/other_notes/
  • http://<local IP address>/json-api/
  • To comment or post with RestClient use:
    – “name” = “Content-Type” and “value” = “application/x-www-form-urlencoded”
Exercise 11 – Search the defects

• Description
  – Search defects on the JSON API in WordPress

• Assignment
  – Use the design test cases
  – Use the learned tooling and techniques
  – Analyze the results
  – Tool: Firefox + Rest Client

Exercise 11 – API details

• API
  – http://<local IP address>/?json=

• Documentation
  • http://wordpress.org/extend/plugins/json-api/other_notes/
  • http://<local IP address>/json-api/
  • To comment or post with RestClient use:
    – “name” = “Content-Type” and “value” = “application/x-www-form-urlencoded”
Exercise 11 - Search the defects

• Which defects are found
  –
  –
  –
  –
  –
  –
  –
  –
  –
  –

Evaluation of exercises

• What went well?
  –
  –
  –
  –
• What went wrong?
  –
  –
  –
  –
• Conclusion
  –
Lessons learned

• API

• API/app communication

• Testing

API

• New test type: production tests
• Command line
• Scope
• Different skills
  – Security
  – Performance
  – Automation
• No backup tricks
API/app communication

• Provider of the API is seen as responsible for the presentation of the data in the app
• Presentation errors of data will always occur
• Very tempting to fix defects in app instead of API
• More explanation needed with secured API

Testing

• Normal test techniques
• Instrument for defect analysis
• Automated regression tests on production
• New responsibilities
• SOA/Interfaces/HTTP protocol/Tools
• Experience with building interfaces
More information

Marc van 't Veer
Test consultant
Polteq Amersfoort (Netherlands)
+31 (0) 6 46 63 61 48 (mob)
http://www.polteq.com
marc.vantveer@polteq.com

Polteq supports its customers:
- with executing and managing test projects
- with implementation and optimization of test processes
- with practical training and certification courses

References

<table>
<thead>
<tr>
<th>Source</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Apple, 2009]</td>
<td>iPhone 3G Commercial &quot;There's An App For That&quot;</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.youtube.com/watch?v=zsrsfeyLzyg">http://www.youtube.com/watch?v=zsrsfeyLzyg</a></td>
</tr>
<tr>
<td>[SmartBear Software, 2011]</td>
<td>SoapUI with REST:</td>
</tr>
<tr>
<td></td>
<td>API testing:</td>
</tr>
<tr>
<td>Test design technique</td>
<td><a href="http://www.pragmaticmarketing.com/resources/why-api-as-a-strategy">http://www.pragmaticmarketing.com/resources/why-api-as-a-strategy</a></td>
</tr>
</tbody>
</table>
### References, continued

<table>
<thead>
<tr>
<th>Source</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Practical API Design</td>
<td>Jaroslav Tulach, chapter 9: Keep Testability in Mind</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Source</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>REST architecture: <a href="http://en.wikipedia.org/wiki/Representational_state_transfer">http://en.wikipedia.org/wiki/Representational_state_transfer</a></td>
</tr>
<tr>
<td></td>
<td>Internet Media Types: <a href="http://en.wikipedia.org/wiki/Internet_media_type#Type_application">http://en.wikipedia.org/wiki/Internet_media_type#Type_application</a></td>
</tr>
</tbody>
</table>
## References, continued

<table>
<thead>
<tr>
<th>Source</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sequence diagrams:</td>
</tr>
<tr>
<td></td>
<td>Dogfooding:</td>
</tr>
<tr>
<td></td>
<td><a href="http://en.wikipedia.org/wiki/Eating_your_own_dog_food">http://en.wikipedia.org/wiki/Eating_your_own_dog_food</a></td>
</tr>
<tr>
<td>Learning REST API</td>
<td><a href="http://www.restapitutorial.com/">http://www.restapitutorial.com/</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://toc.oreilly.com/2013/02/a-publishers-job-is-to-provide-a">http://toc.oreilly.com/2013/02/a-publishers-job-is-to-provide-a</a></td>
</tr>
<tr>
<td></td>
<td>good-api-for-books.html</td>
</tr>
<tr>
<td></td>
<td><a href="http://net.tutsplus.com/tutorials/other/a-beginners-introduction-to-">http://net.tutsplus.com/tutorials/other/a-beginners-introduction-to-</a></td>
</tr>
<tr>
<td></td>
<td>http-and-rest/</td>
</tr>
<tr>
<td></td>
<td><a href="https://github.com/WhiteHouse/api-standards">https://github.com/WhiteHouse/api-standards</a></td>
</tr>
<tr>
<td>HTTP design</td>
<td><a href="http://www.ietf.org/rfc/rfc2616.txt">http://www.ietf.org/rfc/rfc2616.txt</a></td>
</tr>
<tr>
<td>App testing</td>
<td><a href="http://www.kohl.ca">http://www.kohl.ca</a></td>
</tr>
<tr>
<td>HTTP Header elements</td>
<td><a href="http://themayesfamily.com/blogs/b/2011/05/rest-client-for-">http://themayesfamily.com/blogs/b/2011/05/rest-client-for-</a></td>
</tr>
<tr>
<td></td>
<td>firefox-sample-post-request/</td>
</tr>
</tbody>
</table>