	A	B	C	D	E	F	G	H	I
1	ID	API type	Phase	Specific activity	¿Artifact, tool, technique, guideline?	¿Affects usability?	Name	Description	Notes
2	1	Web services	Not mentioned explicitly	Interface description, documentation	Artifact	Not mentioned explicitly	Web Services Description Language (WSDL) documents	Documents that are supposed to describe the API. Service descriptions are mostly brought down to Earth using the Web Services Descrption Language (WSDL), an XLM dialect sponsored by the W3C. These are crucial in enabling third parties to make sense of services and access them.	Las descripciones juegan un rol importane en el descubrimiento de servicios.
3	2	RPC and SOA web services	Not mentioned explicitly	Interface description, documentation	Artifact	Not mentioned explicitly	Web Services Description Language (WSDL) documents	WSDL is the basic unit of RPC web services.	WSDL es la unidad básica de los servicios RPC
4	3	APIs in general, but includes API RESTful example	Requirements, design	Goals identification Usage scenarios identification Resources identification Resources' actions identification	Artifact, technique	Not mentioned explicitly, but it's from a consumer's perspective	API's goal canvas (a table)	To collect users need. To identify an API's goal. To know who are the users, what they can do and how they do it.	Table matching the process we have discovered: *Whos* where you list the API's users (or profiles) *Whats* where you list what these users can do *Hows* where you decompose each *what* in steps what *Inputs (source)* where you list what is needed for each step and where does it come from (to spot missing *whos, whats* or *hows*) *Output (usage)* where you list what is returned by each step and how it is used (to spot missing *whos, whats or hows*) *Goals* where your reformulate explicitly each *how + inputs +outputs* in a concise way how inputs outp
5	4	APIs in general, but includes API RESTful example	Requirements	APIs goals identification	Technique	Not mentioned explicitly, but it's from a consumer's perspective	Questionary	To identify an API's goals, list what users can roughly do and decompose these actions in steps by examining how they do it.	Answer the following questions: What users can do? How they do it? What do they need to do it? What do they get in return? Where does the inputs come from? How does the outputs be used?
6	5	APIs in general, but includes API RESTful example	Design	Identification of resources and it's actions	Guideline (cheat sheet)	Not mentioned explicitly	REST API and HTTP cheat sheet	A resume of how to transpose API goals into REST resources and actions and represent them using the HTTP protocol.	
7	6	APIs in general, but includes API RESTful example	Design	Resources design	Artifact	Not mentioned explicitly	Table of resources	Describing concepts and their properties, gathering the characteristics: name, type, if it's required, adding some description.	.
8	7	APIs in general, but includes API RESTful example	Design	Parameters and responses design	Artifact, technique	Not mentioned explicitly	Ad hoc diagrams of responses and parameters	A same concept may have different representations in the responses of an API depending on the context. An action's response may or may not return the exact concept (or resource), its properties may or may not be adapted (renamed, removed, reorganized). Like for responses, a same concept may have different representations in the parameters of an API depending on the context. A parameter must only provide the needed data but not more. It must not include data that are exclusively handled by the backend.	.
9	8	APIs in general, but includes API RESTful example	Design	Verifying parameter's data	Artifact, technique	Not mentioned explicitly	Flowchart for checking parameters data sources	More detailed view of the input parameters. We must verify again that all needed data can be provided by the cosumer. Consumers must be able to provide all of a parameter's data either because they know the information themselves or because the retrieve it from the API. If a data cannot be provided, it can be the sign of a missing goal or provider's perspective.	.
10	9	APIs in general, but includes API RESTful example	Design	API documentation as a description	Artifact, guideline	Not mentioned explicitly	API description, following a premade format like the OpenAPI Specification (OAS) for REST APIs	When it comes to describe precisely a programming interface and especially it's data, it is more simple and efficient to use a design tool such as an API description format. It is basically a text file containing data describing an API. Provide some basic documentation in the form of descriptions. Such structured and standardized description can be used in many ways and be of great help when designing an API. The OAS is a standard and programming-language agnostic REST API description format. Formerly known as the Swagger Specification. An OAS document can be written in YAML or JSON.	Son útiles desde el diseño hasta la implementación. Es simple y conveniente. Es una manera amigable de compartir la descripción. Útil cuando se quiere obtener retroalimentación.
11	10	APIs in general, but includes API RESTful example	Design	API documentation as a description	Tool	Not mentioned explicitly	Editor for writing OAS documents	Editor wich handle this format, like the Swagger Editor (an open source project). Yon can use your favorite text editor.	

	A	B	C	D	E	F	G	H	I
12	11	WS-* web services	Not mentioned	Documentation	Artifact, guideline	Not mentioned explicitly	WSDL document	We can communicate protocol information to the developers of a consumer application using written documentation, or static contracts such as WSDL. In the WS-* stack, the contract elements are typically implemented using XML Schema, WSDL, etc. The source of most complexity: the Web Services Description Language, or WSDL. While WSDL pays lip service to SOAP's message-oriented processing model, in fact it is mostly used as nothing more than a verbose object interface definition language (IDL), which forces an unsuitable RPC-like model of parameters, return values, and exceptions onto Web Services.	
13	12	Web services	Not mentioned	Service documentation	Technique, guideline	Not mentioned explicitly	URI Templates	Various metadata technologies are used to describe service contracts, including URI templates, which describe syntatic patterns for the set of URIs that a service support. Is human and machine-readable documentation. For humans, a good URI template lays out a map of the service with which we want to interact; for machines, URI templates allow easy and rapid validation of URIs that should resolve to valid addresses for a given service and so can help automate the way clients bind services. In practice, we prefer URI templates as means as internal documentation for services, rather than as a contract metadata.	.
14	13	Web-based services	Not mentioned	Interface description, documentation	Artifact, guideline	Not mentioned explicitly	WADL document (Web Application Description Language)	A WADL contract is an XML document that describes a set of resources with URI templates, permitted operations, and request-response representations. As you'd expect, WADL also supports the HTTP fault model and supports the description of multiple formats for resource representations.	CRUD services are great candidates for describing with WADL.
15	14	Web services	Not mentioned	URIs design	Technique	Not mentioned explicitly	URI Tunneling	HTTP integration option. URI tunneling uses URI as a means of transferring information across system boundaries by encoding the information within the URI itself.	.
16	15	Web services	Not mentioned	Service documentation	Artifact	Not mentioned explicitly	XML Template	For the documentation of the service, similar to WADL or WSDL.	
17	16	APIs RESTful	Not mentioned	Interface description, documentation	Tool	Not mentioned explicitly	Swagger, RAML	Interface definition language for RESTful APIs.	Some IDLs, such as Swagger, allow you to define the format of request and response messages. Others, such as RAML, require you to use a separate specification such as JSON Schema As well as describing APIs, IDLs typically have tools that generate client stubs and server skeletons from an interface definition.
18	17	RESTful APIs	Several phases since it's a pattern: design, implementation , integration.	Architectural design	Guideline (pattern)	Not mentioned explicitly	API Facade Pattern	We recommend you implement an API façade pattern. This pattern gives you a buffer or virtual layer between the interface on top and the API implementation on the bottom. You essentially create a façade – a comprehensive view of what the API should be and importantly from the perspective of the app developer and end user of the apps they create.	Permite descomponer un problema grande en problemas pequeños y más simples. El patrón ofrece una capa virtual entre la interfaz (arriba) y la implementación de la API (abajo) Se crea una fachada de lo que la API debería ser desde la perspectiva del desarrollador cliente y los usuarios finales.
19	18	RESTful APIs	Design	Design point of view	Guideline, collection of design practices	Yes, it places the success of the developer over and above any other design principle	Pragmatic REST	We call our point of view in API design "pragmatic REST", because it places the success of the developer over and above any other design principle. The developer is the customer for the Web API. The success of an API design is measured by how quickly developers can get up to speed and start enjoying success using your API. This approach API design from the 'outside-in' perspective. This means we start by asking - what are we trying to achieve with an API?	
20	19	Web APIs	Design process	Includes steps: Define business objectives, outline key user stories, select technology architecture, write an API architecture, validate your decisions.	Guideline (design process)	Yes	User-centric design process (no specific name)	Design process focused on the user experience to anchor the design decisions. It is designed to solicit feedback in a way that will result in decisions that ultimately benefit the API user.	Includes a worksheet as an appendix.
21	20	Web APIs	Requirements (part of the "define business objectives" phase)	Elicitation	Technique	Not mentioned expicitly, but it is part of a user-centric design process	User stories	Write down some of the use cases that you expect your API to fulfill. Template: As a [user type], I want [action] so that [outcome].	
22	21	Web APIs	Design	Interface description, documentation	Guideline, artifact	Not mentioned expicitly, but it is part of a user-centric design process	API specification	To think through your design thoroughly. It also serves as an artifact that you can use to communicate with other people, especially when you are soliciting feedback from stakeholders. Finally, after you have reached agreement on the spec, it serves as a contract, enabling you to build the various parts of the API implementation in parallel.	We recommend using collaborative document-editing software with version control and commenting support. This is a great way to boost participation, track feedback, and keep everyone up to date with the latest chang
23	22	Web APIs	Design	Design validation	Technique, tool	Yes	Mocking data for interactive user testing	We recommend using whatever tools are available to you in order to test your design and gather the feedback you need.	
24	23	Web APIs	Not mentioned	Documentation	Artifact	Yes	Documentation: Getting started, API reference documentation, Tutorials, FAQ section, Landing page, Changelog, Terms of Service, Samples and Snippets, Code Samples, etc.	Developer resources are a set of assets that you should provide your developers so that they can improve how they use your API.	

	A	B	C	D	E	F	G	H	I
25	24	Web APIs	Planning	Definition of business objectives	Task	Not mentioned explicitly	Problem and impact definition	Before you begin coding or writing your API specification, take a moment to ask yourself two questions: what problem are you trying to solve, and what is the impact you want to have by building this API? The answers to both of these questions must focus on the needs of the user as well as the business you have created.	
26	25	Web APIs	Design	Architectural design	Task	Not mentioned explicitly	Selection of technology architecture	Picking the right paradigm and authentication system is so important. We pick our transport and our authentication mechanism.	
27	26	Web APIs	Planning	Establecimiento de objetivo de negocio y el plan para que la API se alinee con el enfoque general de la empresa	Artifact	Not mentioned explicitly	Plan para el programa de API	Incluye respuesta a varias preguntas, como ¿qué sistemas se van a exponer y dónde (y con quién) se encuentran? Entre otras. Esto se hace para "clasificar" el tipo de API que se va a construir, entre "pública" y "privada". También se alineará el diseño de la API con los objetivos de negocio para estar de acuerdo en los aspectos siguientes: • El estado final objetivo e ideal del programa • Las tareas iniciales que permitirán a la organización trabajar para lograr estos objetivos • Las métricas clave que se emplearán para calcular el éxito • Las tareas diarias continuas que permitirán al programa seguir alcanzando objetivos	.
28	27	Web APIs	Design		Technique	Si, se centra en la experiencia del desarrollador cliente	Definición de perfiles de desarrolladores cliente	Diseñar una persona (o grupo de personas) para definir el tipo(s) de desarrolladores a los que va a dirigir las API.	
29	28	APIs web	Design	Diseño de prototipo	Artifact, technique	Si, se centra en la experiencia del desarrollador cliente	Prototipos, prototipado	Prototipo ligero basado en funciones o datos “desechables”. Se busca retroalimentación del cliente.	
30	29	Web APIs	Design	Diseño de prototipo	Tool	Si, se centra en la experiencia del desarrollador cliente	Herramientas para la creación de prototipos	Menciona Apiary, RAML y Swagger	
31	30	Web APIs	Design	Architectural design	Task	Not mentioned explicitly	Elección del tipo de API	La elección de un tipo de API es una de las decisiones más importantes que puede tomar un diseñador de interfaces. Las decisiones de este tipo se verán afectadas de forma inevitable por aspectos técnicos, como la naturaleza específica de los recursos back-end mostrados o las limitaciones de la organización de TI. Sin embargo, también hay que tener en cuenta otros aspectos, como los objetivos de negocio del programa de API o las necesidades y preferencias de los desarrolladores objetivo.	Menciona que la elección del tipo de API se hace después de la identificación de los desarrolladores clientes.
32	31	Web APIs	Design	Interface description, documentation	Artifact, guideline	Not mentioned explicitly	API description using an API description language, such as RAML or Swagger		
33	32	Web APIs	All phases	Mentioned phases: Domain analysis, architectural design, prototyping, building API software for production, publishing the API.	Guideline (methodology)	Yes	Outside-in + contract-first methodology	This methodology is an outside-in approach and also incorporates ideas of contract first design and simulation. In this methodology, the contract is expressed in the form of an API description. In each step of the methodology, an API description is either created, refined or used -- the API description is the red thread connecting all the steps of the methodology.	
34	33	Web APIs	Domain Analysis (1st phase of the suggested methodology)		Artifact	Not mentioned explicitly	First version of API description, is rather a sketch than an architecture	Defines the API resources, their vocabulary, and wich operations will manipulate those resources.	
35	34	Web APIs	Domain Analysis (1st phase of the suggested methodology)	Verfication of phase "Domain Analysis"	Artifact, technique	Yes	Low-fidelity API prototype	The API prototype should be constructed automatically by generating a simulation based on the API description.	
36	35	Web APIs	Architectural design (2nd phase of the suggested methodology)	Architectural design and detailed design	Artifact	Not mentioned explicitly	API description refined, by including architectural decisions	An appropiate architectural design style should be chosen, such as REST, RPC pr HATEOAS. Architectural decisions should be made. Once the bigger-picture, architectural design decisions are nailed, detailed design decisions can be handled. These design decisions include: Representations, Content type, Parameters, HTTP methods, HTTP status codes, Consistent naming	
37	36	Web APIs	Prototyping (3rd phase of the suggested methodology)		Artifact, technique	Not mentioned explicitly	High-fidelity prototipes	Prototyping is a preparation phase for the productive implementation. Since not every aspect of the API can be implemented, it is important to identify critical aspects of the API, whose feasibility needs to be assessed. The prototype implementation will be tested by pilot consumers.	
38	37	Web APIs	Prototyping (3rd phase of the suggested methodology)	Verification of the "Prototyping" phase	Technique	Yes	Acceptance Tests with Pilot Consumers	An acceptance test is a black-box testing method, where users test if the specifications and requirements of a system are met. Acceptance tests are used to verify the completeness of a system. In our case, API consumers test the API prototype.	
39	38	Web APIs	Domain Analysis (1st phase of the suggested methodology)		Technique, artifact	Not mentioned explicitly	Sketching usage scenarios	The first step of a domain analysis phase is gaining some clarity on the needs of the consumer and possible usage scenarios. Sketching usage scenarios is a creative act.	

	A	B	C	D	E	F	G	H	I
40	39	Web APIs	Domain Analysis (1st phase of the suggested methodology)	Design of resources	Technique, artifact	Not mentioned explicitly	A resource taxonomy for the given usage scenarios.	Shortlist the nouns that would make sense as resource, i.e. nouns for which it would make sense to call operations for create, read, update or delete.	
41	40	Web APIs	Domain Analysis (1st phase of the suggested methodology)	Design of resources	Technique, artifact	Not mentioned explicitly	State diagrams	The resources in the taxonomy have some state and during the execution of the app, the resource may change its state and transition into a new state. You can express the states and transitions in a state diagram	
42	41	Web APIs	Architectural design (2nd phase of the suggested methodology)	Verification of the "Architectural design" phase	Technique	Yes	Simulation	A simulation should be used at this point to quickly verify the effects of the architectural and detailed design decisions. The following questions might help: Is the API still easy to use? Is it still a small, agile and usable API or did we create a monster API? Does this API help us to realize our usage scenarios? Does the API follow the architectural style selected?	
43	42	Web APIs	Requirements (not mentioned by the author)	Requirements specification	Technique, artifact	Yes	Use cases definition with user stories	Use cases are vital throughout the process of creating an API. Once you've defined the business value you want to address with the API and how you're going to measure success, the resulting use cases drive the rest of the process.	
44	43	Web APIs	Design (not mentioned by the author)		Technique	Not mentioned explicitly	Ad-hoc diagrams for the client and API interaction	To represent the complete set of interactions with the API system.	Example in page 32
45	44	Web APIs	Design (not mentioned by the author)		Technique	Not mentioned explicitly	API calls table	To show each call to the API and a description of what it does.	Example in page 33
46	45	Web APIs	Design (Schema modeling)	Interface description	Guideline, technique	Not mentioned explicitly	Schema modeling	Contract describing what the API is, how it works, and what the endpoints are going to be. Think of it as a map of the API, a user-readable description of each endpoint, which can be used to discuss the API before any code is written. Like a functional specification, this document describes how the API will behave.	
47	46	Web APIs	Several phases, since it's a methodology	Includes: creation of the functional specification, schema model (design document), acceptance criteria and unit tests, development iterations.	Guideline (methodology)	Not mentioned explicitly	Design-driven methodology	You'll create your functional specification document. In parallel or shortly after, the schema model is created with use cases. Before developing, you create acceptance criteria for developers to work against along with the unit tests. Only then do you start with development. Instead of developing the entire system at once, you can parallelize and have different engineers working on different use cases so that they can deploy the API. I	
48	47	Web APIs	Planning, requirements	Specification	Artifact	Not mentioned explicitly	Functional specification	It will help the developers and other stakeholders understand the goals of the project. Answer at least the following questions: what problem is the project solving? What is the business value? What are the metrics and use cases? What resources are needed or available? what does "done" look like? what could go wrong?	
49	48	Web APIs	Design	Verification of design	Technique	Yes	Acceptance tests and use cases	Acceptance criteria are critical to verify that you're making the use cases as easy as designed and not getting off track.	
50	49	Web APIs	Not mentioned explicitly	Defining the value of the API	Task	Not mentioned explicitly	Determining business value	Understanding and communicating the APIs business value to your company, including goals to measure sucess, is critical.	
51	50	Web APIs	Not mentioned explicitly	Defining the value of the API	Task	Not mentioned explicitly	Establishing metrics	Determinate how to measure the success of your platform.	
52	51	Web APIs	Not mentioned explicitly	Interface description	Tool	Not mentioned explicitly	Schema modeling frameworks: RAML and OpenAPI (previously Swagger)	These are two of the main schema modeling frameworks.	
53	52	Web APIs	Not mentioned explicitly	Documentation	Artifact	Yes	Documentation: Reference documentation, workflows and tutorials	The second pillar of developer experience is documentation. Documentation convers a wide range of different methods to help developers understand the platform, work with it, and suceed in integrating the API into their own system.	
54	53	Web APIs	Requirements, design	Recomendations	Guideline	Yes	Best practices	Building an API is easy. Designing a usable, flexible, long-lasting API is hard. The author presents a set of guidelines to consider when designing a web API.	
55	54	Web APIs	Design	Documentation	Tool	Yes	RAML and Swagger/OAI	As we start to plan our API, it's important to understand how our users will interact with the API and how they'll use it in conjunction with other services. Be sure to use tools like RAML or wagger/OAI during this process to involve your users, provide mock APIs for them to interact with, and to ensure your design is consistent and meets their needs	
56	55	APIs in general	Design	Recomendations	Guidelines	Yes	Best practices for API design	Includes a set of best practices for several API characteristics, including consistency, good documentation, understability, and some others.	
57	56	APIs in general	Design	Recomendations	Guidelines	Yes	Guidelines for API design	This manual gathers together the key insights into API design that were discovered through many years of software development on the Qt application development framework at Trolltech (now part of Nokia). When designing and implementing a library, you should also keep other factors in mind, such as efficiency and ease of implementation, in addition to pure API considerations.	
58	57	RESTful APIs	Design	Recomendations	Guidelines	Yes, some of the practices are focused on usability issues	Best practices for RESTful web services during design	Best practices for RESTful web services will be presented in detail so that they can be easily applied during the design phase of such web services.	

	A	B	C	D	E	F	G	H	I
59	58	APIs in general	Design	Recomendations	Guidelines	Yes	Guidelines	Few guidelines to use when designing an API. Being aware of these guidelines and taking them explicitly into account during design makes it much more likely that the result will turn out to be usable.	