full-stack

Richard Bouska

Table of Contents

1. WIP Software Engineering [0/5] [DOMAIN]

1.1. WIP Number Encoding

1.1.1. OR, AND, XOR, Shift (logical vs arithmetic)

1.1.2. One's Complement

1.1.3. Two's Complement

1.1.4. BCD encoding

1.1.5. Float: (ieee754)

1.1.5.1. Explain mantisa, exponent *[QUESTION]*

1.1.5.2. How do you compare two float values? *[QUESTION]*

1.1.5.3. Explain risk of using floating point in business programming *[QUESTION]*

1.1.6. Endiannes

Primary source: https://en.wikipedia.org/wiki/Endianness

- In computing, endianness is the order or sequence of bytes of a word of digital data in computer memory. Endianness is primarily expressed as big-endian (BE) or little-endian (LE). A big-endian system stores the most significant byte of a word at the smallest memory address and the least significant byte at the largest. A little-endian system, in contrast, stores the least-significant byte at the smallest address. Endianness may also be used to describe the order in which the bits are transmitted over a communication channel. Bit-endianess is seldom used in other contexts.

1.1.7. binary / octal / hex

1.2. WIP Text Encodings

1.2.1. What is unicode

Primary source: http://www.unicode.org/standard/WhatIsUnicode.html

- The Unicode Standard provides a unique number for every character, no matter what platform, device, application or language. It has been adopted by all modern software providers and now allows data to be transported through many different platforms, devices and applications without corruption. Support of Unicode forms the foundation for the representation of languages and symbols in all major operating systems, search engines, browsers, laptops, and smart phonesplus the Internet and World Wide Web (URLs, HTML, XML, CSS, JSON, etc.). Supporting Unicode is the best way to implement ISO/IEC 10646.
- The emergence of the Unicode Standard and the availability of tools supporting it are among the most significant recent global software technology trends.

1.2.2. utf-8

1.2.3. ASCII

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9 _A	_B	_C	_D	_E	_F
0_	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT LF	VT	FF	CR	SO	SI
0	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009 000A	000B	000C	000D	000E	000F
1_	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM SUB	ESC	FS	GS	RS	US
16	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019 001A	001B	001C	001D	001E	001F
2_	SP	!	"	#	$	%	&	'	(	) *	+	,	-	.	/
32	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029 002A	002B	002C	002D	002E	002F
3_	0	1	2	3	4	5	6	7	8	9 :	;	<	=	>	?
48	0030	0031	0032	0033	0034	0035	0036	0037	0038	0039 003A	003B	003C	003D	003E	003F
4_	@	A	B	C	D	E	F	G	H	I J	K	L	M	N	O
64	0040	0041	0042	0043	0044	0045	0046	0047	0048	0049 004A	004B	004C	004D	004E	004F
5_	P	Q	R	S	T	U	V	W	X	Y Z	[	\	]	^	_
80	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059 005A	005B	005C	005D	005E	005F
6_	`	a	b	c	d	e	f	g	h	i j	k	l	m	n	o
96	0060	0061	0062	0063	0064	0065	0066	0067	0068	0069 006A	006B	006C	006D	006E	006F
7_	p	q	r	s	t	u	v	w	x	y z	{	|	}	~	DEL
112	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079 007A	007B	007C	007D	007E	007F

1.2.4. EBCDIC

1.3. TODO I18N

1.3.1. iso-3166

Primary source: https://en.wikipedia.org/wiki/ISO_3166

- ISO 3166 is a standard published by the International Organization for Standardization (ISO) that defines codes for the names of countries, dependent territories, special areas of geographical interest, and their principal subdivisions (e.g., provinces or states). The official name of the standard is Codes for the representation of names of countries and their subdivisions.

1.3.2. iso-639

Primary source: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

- ISO 639 is a standardized nomenclature used to classify languages. Each language is assigned a two-letter (639-1) and three-letter (639-2 and 639-3) lowercase abbreviation, amended in later versions of the nomenclature.
- ISO 639-2/T: three-letter codes, for the same languages as 639-1
- ISO 639-2/B: three-letter codes, mostly the same as 639-2/T, but with some codes derived from English names rather than native names of languages (in the following table, these differing codes are highlighted in boldface)
- ISO 639-3: three-letter codes, the same as 639-2/T for languages, but with distinct codes for each variety of an ISO 639 macrolanguage

1.3.3. bcp-47 / rfc-5646

1.4. WIP Data Serialization Formats [0/5]

Primary source: https://en.wikipedia.org/wiki/Comparison_of_data-serialization_formats

1.4.1. WIP yaml

Primary source: https://rollout.io/blog/yaml-tutorial-everything-you-need-get-started/

Learn more: https://yaml.org/spec/1.2/spec.html

```yaml
---
# Comments in YAML look like this.

################
# SCALAR TYPES #
################

# Our root object (which continues for the entire document) will be a map,
# which is equivalent to a dictionary, hash or object in other languages.
key: value
another_key: Another value goes here.
a_number_value: 100
scientific_notation: 1e+12
# The number 1 will be interpreted as a number, not a boolean. if you want
# it to be interpreted as a boolean, use true
boolean: true
null_value: null
key with spaces: value
# Notice that strings don't need to be quoted. However, they can be.
however: 'A string, enclosed in quotes.'
'Keys can be quoted too.': "Useful if you want to put a ':' in your key."
single quotes: 'have ''one'' escape pattern'
double quotes: "have many: \", \0, \t, \u263A, \x0d\x0a == \r\n, and more."
# UTF-8/16/32 characters need to be encoded
Superscript two: \u00B2

# Multiple-line strings can be written either as a 'literal block' (using |),
# or a 'folded block' (using '>').
literal_block: |
    This entire block of text will be the value of the 'literal_block' key,
    with line breaks being preserved.

    The literal continues until de-dented, and the leading indentation is
    stripped.

        Any lines that are 'more-indented' keep the rest of their indentation -
        these lines will be indented by 4 spaces.
folded_style: >
    This entire block of text will be the value of 'folded_style', but this
    time, all newlines will be replaced with a single space.

    Blank lines, like above, are converted to a newline character.

        'More-indented' lines keep their newlines, too -
        this text will appear over two lines.

####################
# COLLECTION TYPES #
####################

# Nesting uses indentation. 2 space indent is preferred (but not required).
a_nested_map:
  key: value
  another_key: Another Value
  another_nested_map:
    hello: hello

# Maps don't have to have string keys.
0.25: a float key

# Keys can also be complex, like multi-line objects
# We use ? followed by a space to indicate the start of a complex key.
```

```yaml
? |
  This is a key
  that has multiple lines
: and this is its value

# YAML also allows mapping between sequences with the complex key syntax
# Some language parsers might complain
# An example
? - Manchester United
  - Real Madrid
: [2001-01-01, 2002-02-02]

# Sequences (equivalent to lists or arrays) look like this
# (note that the '-' counts as indentation):
a_sequence:
  - Item 1
  - Item 2
  - 0.5  # sequences can contain disparate types.
  - Item 4
  - key: value
    another_key: another_value
  -
    - This is a sequence
    - inside another sequence
  - - - Nested sequence indicators
      - can be collapsed

# Since YAML is a superset of JSON, you can also write JSON-style maps and
# sequences:
json_map: {"key": "value"}
json_seq: [3, 2, 1, "takeoff"]
and quotes are optional: {key: [3, 2, 1, takeoff]}

########################
# EXTRA YAML FEATURES #
########################

# YAML also has a handy feature called 'anchors', which let you easily duplicate
# content across your document. Both of these keys will have the same value:
anchored_content: &anchor_name This string will appear as the value of two keys.
other_anchor: *anchor_name

# Anchors can be used to duplicate/inherit properties
base: &base
  name: Everyone has same name

# The regexp << is called Merge Key Language-Independent Type. It is used to
# indicate that all the keys of one or more specified maps should be inserted
# into the current map.

foo: &foo
  <<: *base
  age: 10

bar: &bar
  <<: *base
  age: 20

# foo and bar would also have name: Everyone has same name

# YAML also has tags, which you can use to explicitly declare types.
explicit_string: !!str 0.5
# Some parsers implement language specific tags, like this one for Python's
# complex number type.
python_complex_number: !!python/complex 1+2j

# We can also use yaml complex keys with language specific tags
? !!python/tuple [5, 7]
: Fifty Seven
# Would be {(5, 7): 'Fifty Seven'} in Python

######################
# EXTRA YAML TYPES #
######################

# Strings and numbers aren't the only scalars that YAML can understand.
# ISO-formatted date and datetime literals are also parsed.
datetime: 2001-12-15T02:59:43.1Z
datetime_with_spaces: 2001-12-14 21:59:43.10 -5
date: 2002-12-14

# The !!binary tag indicates that a string is actually a base64-encoded
# representation of a binary blob.
```

```
gif_file: !!binary |
  R0lGODlhDAAMAIQAAP//9/X17unp5WZmZgAAAOfn515eXvPz7Y6OjuDg4J+fn5
  OTk6enp56enmlpaWNjY6Ojo4SEhP/++f/++f/++f/++f/++f/++f/++f/+
  +f/++f/++f/++f/++f/++SH+Dk1hZGUgd2l0aCBHSU1QACwAAAAADAAMAAAFLC
  AgjoEwnuNAFOhpEMTRiggcz4BNJHrv/zCFcLiwMWYNG84BwwEeECcgggoBADs=

# YAML also has a set type, which looks like this:
set:
  ? item1
  ? item2
  ? item3
or: {item1, item2, item3}

# Sets are just maps with null values; the above is equivalent to:
set2:
  item1: null
  item2: null
  item3: null

# document end
...
```

1.4.1.1. What are the `---` in yaml?

1.4.1.2. What are the `...` in yaml?

1.4.2. TODO json

1.4.3. TODO xml

1.4.4. TODO protobuf

1.4.5. TODO ASN.1

1.5. WIP Complexity [0/2]

1.5.1. TODO Code complexity (big O)

1.5.2. WIP Amortized Analysis

Primary source: https://en.wikipedia.org/wiki/Amortized_analysis

- method for analyzing a given algorithm's complexity, or how much of a resource, especially time or memory, it takes to execute. The motivation for amortized analysis is that looking at the worst-case run time per operation, rather than per algorithm, can be too pessimistic.[1]
- While certain operations for a given algorithm may have a significant cost in resources, other operations may not be as costly. The amortized analysis considers both the costly and less costly operations together over the whole series of operations of the algorithm. This may include accounting for different types of input, length of the input, and other factors that affect its performance.[2]

1.6. Basic Data Structures

1.6.1. List

1.6.2. Set

1.6.3. Map

1.6.4. Queue

1.6.5. Tree

1.6.6. Finger Tree

Primary source: https://en.wikipedia.org/wiki/Finger_tree

- is a purely functional data structure that can be used to efficiently implement other functional data structures. A finger tree gives amortized constant time access to the "fingers" (leaves) of the tree, which is where data is stored, and concatenation and splitting logarithmic time in the size of the smaller piece. It also stores in each internal node the result of applying some associative operation to its descendants. This "summary" data stored in the internal nodes can be used to provide the functionality of data structures other than trees.

1.6.7. Trie

Primary source: https://en.wikipedia.org/wiki/Trie

- In computer science, a trie, also called digital tree or prefix tree, is a kind of search tree—an ordered tree data structure used to store a dynamic set or associative array where the keys are usually strings. Unlike a binary search tree, no node in the tree stores the key associated with that node; instead, its position in the tree defines the key with which it is associated; i.e., the value of the key is distributed across the structure. All the descendants of a node have a common prefix of the string associated with that node, and the root is associated with the empty string. Keys tend to be associated with leaves, though some inner nodes may correspond to keys of interest. Hence, keys are not necessarily associated with every node. For the space-optimized presentation of prefix tree, see compact prefix tree.

1.7. Basic Algorithms

1.7.1. Stable vs unstable sorting

1.8. Advanced Algorithm

1.8.1. cache-oblivious algorithm

Primary source:	https://en.wikipedia.org/wiki/Cache-oblivious_algorithm
Learn more:	http://erikdemaine.org/papers/BRICS2002/paper.pdf
Learn more:	https://www.cs.ox.ac.uk/ralf.hinze/WG2.8/33/slides/Edward.pdf

- In computing, a cache-oblivious algorithm (or cache-transcendent algorithm) is an algorithm designed to take advantage of a CPU cache without having the size of the cache (or the length of the cache lines, etc.) as an explicit parameter.
- An optimal cache-oblivious algorithm is a cache-oblivious algorithm that uses the cache optimally (in an asymptotic sense, ignoring constant factors).
- Thus, a cache-oblivious algorithm is designed to perform well, without modification, on multiple machines with different cache sizes, or for a memory hierarchy with different levels of cache having different sizes. Cache-oblivious algorithms are contrasted with explicit blocking, as in loop nest optimization, which explicitly breaks a problem into blocks that are optimally sized for a given cache.
- Optimal cache-oblivious algorithms are known for matrix multiplication, matrix transposition, sorting, and several other problems. Some more general algorithms, such as Cooley–Tukey FFT, are optimally cache-oblivious under certain choices of parameters. Because these algorithms are only optimal in an asymptotic sense (ignoring constant factors), further machine-specific tuning may be required to obtain nearly optimal performance in an absolute sense. The goal of cache-oblivious algorithms is to reduce the amount of such tuning that is required.
- Typically, a cache-oblivious algorithm works by a recursive divide and conquer algorithm, where the problem is divided into smaller and smaller subproblems. Eventually, one reaches a subproblem size that fits into cache, regardless of the cache size. For example, an optimal cache-oblivious matrix multiplication is obtained by recursively dividing each matrix into four sub-matrices to be multiplied, multiplying the submatrices in a depth-first fashion. In tuning for a specific machine, one may use a hybrid algorithm which uses blocking tuned for the specific cache sizes at the bottom level, but otherwise uses the cache-oblivious algorithm.

*

2. WIP Architecture `[0/18]` [DOMAIN]

architecture and **design**

2.1. WIP DDD *[TOPIC]*

2.1.1. Bounded Context [TOPIC]

2.1.2. ubiquotous language [TOPIC]

2.1.3. Entity and Value Object [TOPIC]

2.2. WIP Messaging *[TOPIC]*

2.2.1. WIP QoS

- Each connection to the broker can specify a quality of service measure. These are classified in increasing order of overhead:
  - **At most once** - the message is sent only once and the client and broker take no additional steps to acknowledge delivery (fire and forget).
  - **At least once** - the message is re-tried by the sender multiple times until acknowledgement is received (acknowledged delivery).
  - **Exactly once** - the sender and receiver engage in a two-level handshake to ensure only one copy of the message is received (assured delivery).

2.2.2. WIP Tools

2.2.2.1. TODO Solace Pub/Sub Plus

2.2.2.2. TODO NATS

2.2.2.3. TODO Google Pub/Sub

2.2.2.4. TODO AWS SNS

2.2.2.5. TODO AWS SES

2.2.2.6. TODO AWS SQS

2.2.2.7. TODO AWS Kinesis

2.2.2.8. TODO RabbbitMQ

2.2.2.9. TODO Kafka

2.2.2.10. WIP Redis

Primary source:	https://redis.io/topics/introduction

- Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.
- You can run atomic operations on these types, like appending to a string; incrementing the value in a hash; pushing an element to a list; computing set intersection, union and difference; or getting the member with highest ranking in a sorted set.
- In order to achieve its outstanding performance, Redis works with an in-memory dataset. Depending on your use case, you can persist it either by dumping the dataset to disk every once in a while, or by appending each command to a log. Persistence can be optionally disabled, if you just need a feature-rich, networked, in-memory cache.
- Redis also supports trivial-to-setup master-slave asynchronous replication, with very fast non-blocking first synchronization, auto-reconnection with partial resynchronization on net split.

- Other features include:
  - Transactions
  - Pub/Sub
  - Lua scripting
  - Keys with a limited time-to-live
  - LRU eviction of keys
  - Automatic failover
  - You can use Redis from most programming languages out there.
- Redis is written in ANSI C and works in most POSIX systems like Linux, *BSD, OS X without external dependencies. Linux and OS X are the two operating systems where Redis is developed and tested the most, and we recommend using Linux for deploying. Redis may work in Solaris-derived systems like SmartOS, but the support is best effort. There is no official support for Windows builds.

2.2.3. TODO JMS style mapping

2.2.3.1. TODO Queue

2.2.3.2. TODO Topics

2.2.4. Protocols

2.2.4.1. WIP AMQP (Advanced Message Queuing Protocol)

Primary source: https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol

- AMQP 1.0 broker implementations
  - **Apache Qpid**, an open-source project at the Apache Foundation
  - **Apache ActiveMQ**, an open-source project at the Apache Foundation
  - Azure Event Hubs
  - Azure Service Bus
  - **Solace PubSub+**, a multi-protocol broker in hardware, software, and cloud
- beware there is more incompatible protcol versions 0-8, 0-9, 0-9-1

2.2.4.2. WIP MQTT (Message Queuing Telemetry Transport)

Primary source: https://en.wikipedia.org/wiki/MQTT

- The MQTT protocol defines two types of network entities: a message broker and a number of clients. An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients.[13] An MQTT client is any device (from a micro controller up to a full-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.[14]
- Information is organized in a hierarchy of topics. When a publisher has a new item of data to distribute, it sends a control message with the data to the connected broker. The broker then distributes the information to any clients that have subscribed to that topic. The publisher does not need to have any data on the number or locations of subscribers, and subscribers, in turn, do not have to be configured with any data about the publishers.
- If a broker receives a message on a topic for which there are no current subscribers, the broker discards the message unless the publisher of the message designated the message as a retained message. A retained message is a normal MQTT message with the retained flag set to true. The broker stores the last retained message and the corresponding QoS for the selected topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic.[15] This allows new subscribers to a topic to receive the most current value rather than waiting for the next update from a publisher.
- When a publishing client first connects to the broker, it can set up a default message to be sent to subscribers if the broker detects that the publishing client has unexpectedly disconnected from the broker.
- Clients only interact with a broker, but a system may contain several broker servers that exchange data based on their current subscribers' topics.
- A minimal MQTT control message can be as little as two bytes of data. A control message can carry nearly 256 megabytes of data if needed. There are fourteen defined message types used to connect and disconnect a client from a broker, to publish data, to acknowledge receipt of data, and to supervise the connection between client and server.
- MQTT relies on the TCP protocol for data transmission. A variant, MQTT-SN, is used over other transports such as UDP or Bluetooth.
- MQTT sends connection credentials in plain text format and does not include any measures for security or authentication. This can be provided by using TLS to encrypt and protect the transferred information against interception, modification or forgery.
- The default unencrypted MQTT port is 1883. The encrypted port is 8883.[16]

2.3. WIP HTTP Family *[TOPIC]*

2.3.1. HTTP

2.3.1.1. What is the use of Accept and Content-Type Headers in HTTP Request? *[QUESTION]*

- Accept headers:
  - tells web service what kind of response client is accepting, so if a web service is capable of sending response in XML and JSON format and client sends Accept header as application/xml then XML response will be sent. For Accept header application/json, server will send the JSON response.
- Content-Type header:
  - is used to tell server what is the format of data being sent in the request. If Content-Type header is application/xml then server will try to parse it as XML data. This header is useful in HTTP Post and Put requests.

2.3.1.2. WIP What is Server-Sent Events (SSE) *[QUESTION]*

- is a standard describing how servers can initiate data transmission towards clients once an initial client connection has been established.
- They are commonly used to send message updates or continuous data streams to a browser client
  - designed to enhance native, cross-browser streaming through a JavaScript API called EventSource,
    - through which a client requests a particular URL in order to receive an event stream.

2.3.1.3. WIP What are WebSockets *[QUESTION]*

2.3.1.4. WIP EventSource

Primary source: https://developer.mozilla.org/en-US/docs/Web/API/EventSource

- The EventSource interface is web content's interface to server-sent events.

- An EventSource instance opens a persistent connection to an HTTP server, which sends events in text/event-stream format.
- The connection remains open until closed by calling EventSource.close().
- Once the connection is opened, incoming messages from the server are delivered to your code in the form of events.
- If there is an event field in the incoming message, the triggered event is the same as the event field value. If no event field is present, then a generic message event is fired.
- Unlike WebSockets, server-sent events are unidirectional; that is, data messages are delivered in one direction, from the server to the client (such as a user's web browser).
- That makes them an excellent choice when there's no need to send data from the client to the server in message form.
- For example, EventSource is a useful approach for handling things like social media status updates, news feeds, or delivering data into a client-side storage mechanism like IndexedDB or web storage.

2.3.1.5. TODO XMLHttpRequest (XHR)

- objects are used to interact with servers.
- You can retrieve data from a URL without having to do a full page refresh.
- This enables a Web page to update just part of a page without disrupting what the user is doing.
- XMLHttpRequest is used heavily in AJAX programming.
- Despite its name, XMLHttpRequest can be used to retrieve any type of data, not just XML.

2.3.1.6. WIP Fetch

- Fetch is a fairly recent addition to the JavaScript world (introduced in 2015) utilizing a modern syntax for making requests from the browser.
- Like most modern JS written today, Fetch is based on Promises rather than callbacks.
- In our project, the Fetch example can be seen here and the code is shown below:

```
<!DOCTYPE html>
<head>
  <title>FETCH Example</title>
</head>

<body>
  <div>
    <h1>Select Reddit Sub</h1>
    <div id="list"></div>

  </div>

  <script>
    (function(){
      fetch('/api/endpoint')
        .then(payload => payload.json())
        .then(resp => {
          var data = resp.data // array of Reddit subs returned from "endpoint.js"
          console.log(JSON.stringify(data))
          var html = '<ol>'
          data.forEach(function(sub){
            html += '<a href="/axios/'+sub.path+'">'+sub.name+'</a><br />'
          })
          html += '</ol>'
          document.getElementById('list').innerHTML = html
        })
        .catch(err => {
          console.log('ERROR. Something went wrong.')
        })
    })()
  </script>
</body>
</html>
```

- The actual Fetch request takes place in the <script> tag between lines 16 and 33. On line 16, we make a request to the url "/api/endpoint" which returns a list of Reddit subs available to view. The initial response is on line 17 ("payload") which we then convert into JSON and parse starting on line 19. The "data" object is an array of Reddit subs with "name" and "path" keys. In lines 22 through 28, we populate a tag with id "list" using the objects in the data array and connect each element to a link to view the sub content.

2.3.1.7. WIP axios

Axios: Axios is a Javascript library used to make HTTP requests from node.js or XMLHttpRequests from the browser and it supports the Promise API that is native to JS ES6. It can be used intercept HTTP requests and responses and enables client-side protection against XSRF. It also has the ability to cancel requests.

```
axios.get('url')
  .then((response) => {

    // Code for handling the response
  })
  .catch((error) => {

    // Code for handling the error
  })
```

2.4. WIP REST

2.4.1. What are the HTTP methods supported by REST? [QUESTION]

HTTP methods supported by REST are:

GET	It requests a resource at the request URL. It should not contain a request body as it will be discarded. Maybe it can be cached locally or on the server.

POST	It submits information to the service for processing; it should typically return the modified or new resource
PUT	At the request URL it update the resource
DELETE	At the request URL it removes the resource
OPTIONS	It indicates which techniques are supported
HEAD	About the request URL it returns meta information

2.4.2. What is the purpose of HTTP Status Code? [QUESTION]

HTTP Status code are standard codes and refers to predefined status of task done at server. For example, HTTP Status 404 states that requested resource is not present on server.
Consider following status codes:

200 OK	shows success.
201 CREATED	when a resource is successful created using POST or PUT request. Return link to newly created resource using location header.
304 NOT MODIFIED	used to reduce network bandwidth usage in case of conditional GET requests. Response body should be empty. Headers should have date, location etc.
400 BAD REQUEST	states that invalid input is provided e.g. validation error, missing data.
401 FORBIDDEN	states that user is not having access to method being used for example, delete access without admin rights.
404 NOT FOUND	states that method is not available.
409 CONFLICT	states conflict situation while executing the method for example, adding duplicate entry.
500 INTERNAL SERVER ERROR	states that server has thrown some exception while executing the method.

2.4.3. What is statelessness in RESTful WebServices? [QUESTION]

- As per REST architecture, a RESTful web service should not keep a client state on server.
- This restriction is called statelessness.
- It is responsibility of the client to pass its context to server and then server can store this context to process client's further request.
- For example, session maintained by server is identified by session identifier passed by the client.

2.4.4. What is the difference between PUT and POST? [QUESTION]

- PUT puts a file or resource at a particular URI and exactly at that URI. If there is already a file or resource at that URI, PUT changes that file or resource. If there is no resource or file there, PUT makes one
- POST sends data to a particular URI and expects the resource at that URI to deal with the request. The web server at this point can decide what to do with the data in the context of specified resource
- PUT is idempotent meaning, invoking it any number of times will not have an impact on resources.
- However, POST is not idempotent, meaning if you invoke POST multiple times it keeps creating more resources

2.4.5. TODO HATEOAS

2.5. TODO GraphQL

2.6. WIP 12 Factor Apps

:=PROPERTIES: :learning_1: https://12factor.net/

1. **Codebase**
   - One codebase tracked in revision control, many deploys
2. **Dependencies**
   - Explicitly declare and isolate dependencies
3. **Config**
   - Store config in the environment
4. **Backing services**
   - Treat backing services as attached resources
5. **Build, release, run**
   - Strictly separate build and run stages
6. **Processes**
   - Execute the app as one or more stateless processes
7. **Port binding**
   - Export services via port binding
8. **Concurrency**
   - Scale out via the process model
9. **Disposability**
   - Maximize robustness with fast startup and graceful shutdown
10. **Dev/prod parity**
    - Keep development, staging, and production as similar as possible
11. **Logs**
    - Treat logs as event streams
12. **Admin processes**
    - Run admin/management tasks as one-off processes

2.7. WIP Hasura 3Factor Apps

Primary source:	https://3factor.app/

- epizody 115 116 a 117.
- Konkrétně epizoda 116 na kterou posílám odkaz je hodná poslechu.
- 3f Apps je architektura ke které se blížíme.
- V té 116 epizodě je právě celá diskuse o messagingu, otázky msg fall-back..
- Format diskuse je takový rozbíhavý, ale berou to že široka.
- Zajímavý je to hodně tak po prvních osmi minutách a ke konci zase kecaji jenom tak zbůhdarma.
- Pěkný je i to že to je diskuse mezi 3mi lidmi a ne všichni tomu rozumí, tak tam občas narazíš na blbosti.
- Jo a je tam slyšet o čem že je ta technická architektura.
- https://www.codingblocks.net/episode115
- https://www.codingblocks.net/episode116
- https://www.codingblocks.net/episode117

2.7.1. Factor #1: Realtime GraphQL

- Apart from providing an amazing frontend developer experience, GraphQL is a crucial component in 3factor architecture as it allows flexible API access and realtime capabilities. The GraphQL API should have the following properties:
- Low-latency: An end-user should see instant feedback of an action (i.e. state manipulation) and not have to wait long on an API (<100ms ideal, upto 1 second at worst).
- Support subscriptions: Consume information “realtime” from the backend via GraphQL Subscriptions. Avoid the use of continuous polling (for scalability).

2.7.2. Factor #2: Reliable Eventing

- In 3factor, business logic is initiated via events. This removes complex state management in your API layer and defers it to bespoke business logic functions. Events can also be persisted so that entire history of state is available for observability. Further, the event system must have the following 2 properties:
- Atomic: Mutations to the application state should atomically create event(s).
- Reliable: Events should be delivered (to any consumer) with atleast-once guarantee.

2.7.3. Factor #3: Async Serverless

- Write business logic as event handling functions. Each function only cares about one event and is hence small & cohesive. The easiest way to deploy such functions is in serverless compute. Serverless minimizes backend ops and gives “infinite” scalability while being cost-efficient. The serverless functions should follow few best-practices:
- Idempotent: The code should be prepared for duplicate delivery of events.
- Out-of-order: Events may not be guaranteed to be received in any realtime order. The code should not depend on any expected sequence of events.

2.8. WIP OpenApiSpec / Open API 3.0 / Swagger

Primary source:	https://app.swaggerhub.com/help/tutorials/openapi-3-tutorial
Learn more:	https://editor.swagger.io/

2.8.1. What Is OpenAPI?

- OpenAPI Specification (formerly known as Swagger Specification) is an open-source format for describing and documenting APIs. The Specification was originally developed in 2010 by Reverb Technologies (formerly Wordnik) as a way to keep the API design and documentation in sync. It has since become a de-facto standard for designing and describing RESTful APIs, and is used by millions of developers and organizations for developing their APIs, be it internal or client facing.
- The latest version of OpenAPI is 3.0. OpenAPI definitions can be written in JSON or YAML. We recommend YAML, because it is easier to read and write.
- A simple OpenAPI 3.0 specification looks like this:

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: Sample API
  description: A sample API to illustrate OpenAPI concepts
paths:
  /list:
    get:
      description: Returns a list of stuff
      responses:
        '200':
          description: Successful response
```

2.9. WIP Enterprise Architecture Integration Patterns [0/4]

2.9.1. WIP Integration Styles:

2.9.1.1. TODO File Transfer *[PATTERN]*

2.9.1.2. TODO Shared Database *[PATTERN]*

2.9.1.3. TODO Remote Procedure Invocation *[PATTERN]*

2.9.1.4. TODO Messaging *[PATTERN]*

2.9.2. WIP Messaging Systems

2.9.2.1. Message Channel *[PATTERN]*

2.9.2.2. Message *[PATTERN]*

2.9.2.3. Pipes and Filtres *[PATTERN]*

2.9.2.4. Message Router *[PATTERN]*

2.9.2.5. Message Translator *[PATTERN]*

2.9.2.6. Message Endpoint *[PATTERN]*

2.9.3. TODO Choreography

- distributed decision making
  - simple..
  - for primitive cases.
  - Higher connaissance
  - cyclic dependencies …
  - Overloads domain objectsviin. I minimal

2.9.4. TODO Orchestration

- centralized decision making
  - orchestration objects implements state machine and invokes the participants

2.10. WIP µ-Services design patterns classes [0/6]

Primary source:	https://tsh.io/blog/design-patterns-in-microservices-api-gateway-bff-and-more/

2.10.1. WIP Communication

- It concerns the methods of communication between microservices and client apps (the frontend layer).

2.10.2. WIP Internal communication

- These design patterns constitute various ways microservices can communicate between each other.

2.10.3. WIP Security

- A variety of security-related concerns, such as the organization of the security layer, authorization and level of access to particular microservices for different types of users, etc.

2.10.4. WIP Availability

- Ensuring that all microservices are ready to address the needs of the system (regardless of how intense the traffic is), ensuring the lowest possible downtime. This includes activities such as making sure microservices can restart on another machine, or whether enough machines are available, microservices being able to report their current states on their own, health checks and more.

2.10.5. WIP Service discovery

- It refers to the methods microservices use to find each other and make their locations known.

2.10.6. WIP Configuration

- Setting parameters and monitoring the performance of the entire system to continuously optimize it as you go

2.11. TODO µ-Services Design Patterns [0/8]

Primary source:	https://tsh.io/blog/design-patterns-in-microservices-api-gateway-bff-and-more/

2.11.1. TODO API Gateway [PATTERN]

2.11.2. TODO Circuit Breaker [PATTERN]

2.11.3. WIP CQRS [PATTERN]

2.11.4. TODO Event Sourcing [PATTERN]

2.11.5. TODO Sidecar [PATTERN]

2.11.6. TODO Backend-for-Frontend [PATTERN]

2.11.7. TODO Strangler Pattern [PATTERN]

2.11.8. TODO Saga [PATTERN]

2.12. TODO Event Sourcing [PATTERN]

2.13. TODO CQRS [PATTERN]

2.14. TODO CRUD [PATTERN]

2.15. WIP (Pragmatic) Design principes `[0/11]`

2.15.1. TODO DRY [TOPIC]

2.15.2. TODO KISS [TOPIC]

2.15.3. WIP GRASP

Primary source: https://en.wikipedia.org/wiki/GRASP_%28object-oriented_design%29

- General Responsibility Assignment Software Patterns (or Principles), abbreviated GRASP, consist of guidelines for assigning responsibility to classes and objects in object-oriented design.[1] It is not related to the SOLID design principle.
- The different patterns and principles used in GRASP are controller, creator, indirection, information expert, low coupling, high cohesion, polymorphism, protected variations, and pure fabrication. All these patterns answer some software problems, and these problems are common to almost every software development project. These techniques have not been invented to create new ways of working, but to better document and standardize old, tried-and-tested programming principles in object-oriented design.
- Computer scientist Craig Larman states that "the critical design tool for software development is a mind well educated in design principles. It is not UML or any other technology."[2] Thus, GRASP are really a mental toolset, a learning aid to help in the design of object-oriented software.

2.15.4. WIP SOLID

- **S** single responsibility
- **O** open–closed
- **L** liskov substitution
- **I** interface segregation
- **D** dependency inversion

2.15.5. TODO YAGNI [TOPIC]

2.15.6. TODO Open/Close principal [TOPIC]

2.15.7. TODO High Cohesion [TOPIC]

2.15.8. TODO Low Coupling [TOPIC]

2.15.8.1. What is temporal coupling? [QUESTION]

2.15.9. WIP Information Hiding [TOPIC]

- In computer science, information hiding is the principle of segregation of the design decisions in a computer program that are most likely to change, thus protecting other parts of the program from extensive modification if the design decision is changed. The protection involves providing a stable interface which protects the remainder of the program from the implementation (the details that are most likely to change).
- Written another way, information hiding is the ability to prevent certain aspects of a class or software component from being accessible to its clients, using either programming language features (like private variables) or an explicit exporting policy

2.15.10. WIP Encapsulation [TOPIC]

- The mechanism of Encapsulation in ObjectOriented Programming (OOP) is a direct application of Information Hiding

2.15.11. WIP The Law of Demeter (LoD) or principle of least knowledge

Primary source: https://en.wikipedia.org/wiki/Law_of_Demeter

- is a design guideline for developing software, particularly object-oriented programs. In its general form, the LoD is a specific case of loose coupling. The guideline was proposed by Ian Holland at Northeastern University towards the end of 1987, and can be succinctly summarized in each of the following ways:[1]
  - Each unit should have only limited knowledge about other units: only units "closely" related to the current unit.
  - Each unit should only talk to its friends; don't talk to strangers.
  - Only talk to your immediate friends.
- The fundamental notion is that a given object should assume as little as possible about the structure or properties of anything else (including its subcomponents), in accordance with the principle of "information hiding". It may be viewed as a corollary to the principle of least privilege, which dictates that a module possess only the information and resources necessary for its legitimate purpose.
- It is so named for its origin in the Demeter Project, an adaptive programming and aspect-oriented programming effort. The project was named in honor of Demeter, “distribution-mother” and the Greek goddess of agriculture, to signify a bottom-up philosophy of programming which is also

embodied in the law itself.

2.16. TODO Antifragile architecture

2.17. WIP Evolutionary database design

Primary source: https://martinfowler.com/articles/evodb.html

3. WIP Data `[0/10]` [DOMAIN]

3.1. WIP background *[TOPIC]*

3.1.1. What does ACID mean? [QUESTION]

Primary source: https://en.wikipedia.org/wiki/ACID

A
: atomicity

C
: consistency

I
: isolation

D
: durability

3.1.2. WIP BASE

BA
: Basic Availability

S
: Soft-state

E
: Eventual consistency

3.1.3. ORM

3.1.3.1. What ORM (object-relational-mapping) means *[QUESTION]*

3.1.3.2. What is object relation impedance mismatch *[QUESTION]*

Primary source: https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch

- The object-relational impedance mismatch is a set of conceptual and technical difficulties that are often encountered when a relational database management system (RDBMS) is being served by an application program (or multiple application programs) written in an object-oriented programming language or style.
- particularly because objects or class definitions must be mapped to database tables defined by a relational schema.

3.1.3.3. Bottom Up *[QUESTION : CONCEPT]*

3.1.3.4. Top Down *[QUESTION : CONCEPT]*

3.1.3.5. Meet in the Middle *[QUESTION : CONCEPT]*

3.1.4. CAP theorem

Primary source: https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e

- CAP Theorem is a concept that a distributed database system can only have 2 of the 3: Consistency, Availability and Partition Tolerance.
- CAP Theorem is very important in the Big Data world, especially when we need to make trade off's between the three, based on our unique use case.

3.1.4.1. Beyond CAP database systems

Primary source: http://www.grokkingsystemdesigns.com/beyond-cap-theorem/

3.1.5. Wall: Write Ahead Logs

3.1.5.1. what it is good for

3.1.6. Eventual Consistency

3.1.7. Multi master replication

3.1.8. OLAP

- OnLine Transactional Processing

3.1.9. OLTP

- OnLine Analytical Processing

3.1.10. Slowly Moving Dimensions

3.1.10.1. What is it? *[QUESTION]*

- Type 0 - The passive method
- Type 1 - Overwriting the old value
- Type 2 - Creating a new additional record
- Type 3 - Adding a new column

- Type 4 - Using historical table
- Type 6 - Combine approaches of types 1,2,3 (1+2+3=6)

3.1.10.2. Type 0 - The passive method. *[QUESTION]*

- In this method no special action is performed upon dimensional changes.
- Some dimension data can remain the same as it was first time inserted, others may be overwritten.

3.1.10.3. Type 1 - Overwriting the old value. *[QUESTION]*

- In this method no history of dimension changes is kept in the database.
- The old dimension value is simply overwritten be the new one.
- This type is easy to maintain and is often use for data which changes are caused by processing corrections (e.g. removal special characters, correcting spelling errors).

3.1.10.4. Type 2 - Creating a new additional record. *[QUESTION]*

- In this methodology all history of dimension changes is kept in the database.
- You capture attribute change by adding a new row with a new surrogate key to the dimension table.
- Both the prior and new rows contain as attributes the natural key(or other durable identifier).
- Also 'effective date' and 'current indicator' columns are used in this method. There could be only one record with current indicator set to 'Y'. For 'effective date' columns, i.e. start_date and end_date, the end_date for current record usually is set to value 9999-12-31. Introducing changes to the dimensional model in type 2 could be very expensive database operation so it is not recommended to use it in dimensions where a new attribute could be added in the future.

3.1.10.5. Type 3 - Adding a new column. *[QUESTION]*

- In this type usually only the current and previous value of dimension is kept in the database.
- The new value is loaded into 'current/new' column and the old one into 'old/previous' column.
- Generally speaking the history is limited to the number of column created for storing historical data. This is the least commonly needed technique.

3.1.10.6. Type 4 - Using historical table. *[QUESTION]*

- In this method a separate historical table is used to track all dimension's attribute historical changes for each of the dimension.
- The 'main' dimension table keeps only the current data e.g. customer and customer_history tables.

3.1.10.7. Type 6 - Combine approaches of types 1,2,3 (1+2+3=6). *[QUESTION]*

- In this type we have in dimension table such additional columns as:
- current_type - for keeping current value of the attribute. All history records for given item of attribute have the same current value.
- historical_type - for keeping historical value of the attribute. All history records for given item of attribute could have different values.
- start_date - for keeping start date of 'effective date' of attribute's history.
- end_date - for keeping end date of 'effective date' of attribute's history.
- current_flag - for keeping information about the most recent record.
- In this method to capture attribute change we add a new record as in type 2. The current_type information is overwritten with the new one as in type 1. We store the history in a historical_column as in type 3.

3.1.11. Backup strategy

3.1.11.1. full vs incremental backup

3.1.11.2. snapshots

3.1.11.3. on-line replica

3.2. WIP SQL

3.2.1. partitionning

- co to je? rozdělení tabulek na více částí, může být vertikální či horizontální
- proč se to používá? např. zvýšení výkonu (nevypisuji z celé tabulky ale jen z jedné partition) / zátěž rozložena na více disků rychlejší čtení, velké tabulky které se nevejdou na disk -> vice partiton

3.2.2. inner vs outher join

- typy spojení DB tabulek,
  - inner: výsledek musí obsahovat průnik obou tabulek,
  - outer: výsledek obsahuje data z jedné či druhé, případně z obou tabulek pokud klíč existuje v obou tabulkách …
- záleží na typu outer joinu - Full/left/right

3.2.3. DB Index

- Druhy primary/unique/(secondary) index …. Speciální: partial (na části dat definovaný where condition) / functional (nepoužívá atribut, ale funkci nad atributem/atributy) / spatial (prostorový index nad geografickými daty) k čemu slouží? K rychlejšímu vyhledávání v tabulce, v případě unique/primary zamezuje vložení duplicity

3.2.4. Constraint

- Omezení hodnoty vkládané do atribut, foreign key constraint hodnota musí existovat v referenční tabulce, primary key constaint unikátní hodnota v tabulce

3.2.5. Agregace (group by)

3.2.6. Co jsou window funkce? [QUESTION]

- kalkulace/agregace nad množinou dat, přičemž každý prvek množiny může mít výsledek agregace jiný ….
- Např funkce lead/lag/rank/row_number/first/last/

3.2.7. what is a defference between CTE and temp table? [QUESTION]

- A CTE is just that – Common Table Expression, that is, only a syntax construct. The result set described by a CTE may never be materialized in the specified form.
- A temporary table, on the other hand, is a real database object that is initialized with the structure described by its DDL statement and possibly populated by actual rows.

3.2.8. CTE (with)

- syntaxe pro vytvoření dočasné tabulky, kterou mohu použít v následujícím sql dotazu,
- lze pomocí ní dělat rekurzivní dotazy, výsledek je jen v paměti a neukládá se na disk (pokud mám dostatek working memory)

3.2.9. DML/DDL

- manipuklace s daty v tabulce (insert/delete/update) / definuje datové struktury (db,schema, tabulka) (create / drop / alter / rename)

3.3. WIP Distributed consensus: *[CONCEPT]*

3.3.1. Paxos [TOPIC]

Primary source: https://en.wikipedia.org/wiki/Paxos_(computer_science)

- Paxos is a family of protocols for solving consensus in a network of unreliable or fallible processors.
  - Consensus is the process of agreeing on one result among a group of participants.
  - This problem becomes difficult when the participants or their communications may experience failures.[1]
- Consensus protocols are the basis for the state machine replication approach to distributed computing, as suggested by Leslie Lamport[2] and surveyed by Fred Schneider.[3] State machine replication is a technique for converting an algorithm into a fault-tolerant, distributed implementation. Ad-hoc techniques may leave important cases of failures unresolved. The principled approach proposed by Lamport et al. ensures all cases are handled safely.
- The Paxos protocol was first submitted in 1989 and named after a fictional legislative consensus system used on the Paxos island in Greece, where Lamport wrote that the parliament had to function "even though legislators continually wandered in and out of the parliamentary Chamber".[4] It was later published as a journal article in 1998.[5]
- The Paxos family of protocols includes a spectrum of trade-offs between the number of processors, number of message delays before learning the agreed value, the activity level of individual participants, number of messages sent, and types of failures. Although no deterministic fault-tolerant consensus protocol can guarantee progress in an asynchronous network (a result proved in a paper by Fischer, Lynch and Paterson[6]), Paxos guarantees safety (consistency), and the conditions that could prevent it from making progress are difficult to provoke.
- Paxos is usually used where durability is required (for example, to replicate a file or a database), in which the amount of durable state could be large.
- The protocol attempts to make progress even during periods when some bounded number of replicas are unresponsive.
- There is also a mechanism to drop a permanently failed replica or to add a new replica.

3.3.2. Raft [TOPIC]

3.3.3. Zab [TOPIC]

3.4. WIP Transactions *[CONCEPT]*

3.4.1. transaction problems that are prevented by isolation:

3.4.1.1. What are they and what they mean?

- Dirty reads
- non-repeatable reads
- phantom reads

3.4.2. JDBC transaction isolation levels

3.4.2.1. what are they and what they prevent from?

- TRANSACTION_READ_UNCOMMITTED
  - Dirty reads, non-repeatable reads and phantom reads can occur.
- TRANSACTION_READ_COMMITTED
  - Dirty reads are prevented; non-repeatable reads and phantom reads can occur.
- TRANSACTION_REPEATABLE_READ
  - Dirty reads and non-repeatable reads are prevented; phantom reads can occur.
- TRANSACTION_SERIALIZABLE
  - Dirty reads, non-repeatable reads and phantom reads are prevented.

3.4.3. ANSI/ISO levels

3.4.3.1. what they are

- serializable
- repeatable read
- read committed
- read uncommitted

3.4.3.2. define some

- The Serializable isolation level guarantees that concurrent transactions run as they would if you would run sequentially one by one in order.
- One step weaker is the Read Repeatable isolation level that allows Phantom Reads to happen in the transaction. Contrary to transactions running in the Serializable mode, the set of rows that is returned by two consecutive select queries in a transaction can differ. This can happen if another transaction adds or removes rows from the table we are querying.
- Even weaker is the Read Commited isolation level. Two consecutive select statements in a transaction can return different data. Contrary to the Read Repeatable level, this level allows not only the set of rows to change, but also the data that those rows contain. This can happen if another transaction modifies the rows.
- The weakest isolation level is Read Uncommitted where dirty reads can occur. That means that non-committed changes from other transactions can affect a transaction.

3.5. WIP Datawarehouse?

- úložiště dat,
- organizovaných do určité struktury,
- pro účely analýzy dat … podpora business procesů organizace

3.6. WIP Star schema structure

- tvořené faktovou tabulkou (jednotlivá měření) a dimenzemi (entity popisující fakta); dimenze mohou být denormalizovánay

3.7. WIP Snowflake schema structure

- faktová tabulka (jednotlivá měření) + dimenze jsou normalizovány (jedna dimenze může být složena z více tabulek) 3NF

3.8. WIP WAL / transactional log?

- log kde je historie všech provedených DB operací nad danou instancí/databází.
- Z logu se zapisují záznamy do datových souborů.
- Z logu se dá obnovit databáze do určitého časového okamžiku (PITR-Point In Time Recovery);
- u postgre se dá použít například i k replikaci databáze

3.9. WIP MVCC Multiversion concurrency control

Primary source: https://en.wikipedia.org/wiki/Multiversion_concurrency_control

- Multiversion concurrency control (MCC or MVCC), is a concurrency control method commonly used by database management systems to provide concurrent access to the database and in programming languages to implement transactional memory.

3.10. WIP Tools `[0/22]`

3.10.1. WIP PostgreSQL

Primary source: https://www.highgo.ca/blog/

3.10.1.1. TODO What is the current version of PostgreSQL *[QUESTION]*

3.10.1.2. TODO Explain about Multi version concurrency control? *[QUESTION]*

- Multi version concurrency control or MVCC is used to avoid unnecessary locking of the database.
- This removes the time lag for the user to log into his database.
- This feature or time lag occurs when some one else is on the content. All the transactions are kept as a record.

3.10.1.3. TODO psql

3.10.1.4. WIP B-Tree Index

- is the default index type in PostgreSQL that gets created when you do a 'CREATE INDEX' statement without mentioning the index name.
- is index is much suitable for the data that can be sorted and can handle equality and range queries. The following command is used to create a btree index:

```
CREATE INDEX name ON table (column); or
CREATE INDEX name ON table USING BTREE (column);
```

3.10.1.5. WIP Hash Index

- The hash index prior to PostgreSQL version 10 were almost discouraged for various reasons such as:
- Not WAL logged hence not crash safe
- Not replicated to stand-by server
- Performance is not so good and may take a long time to build the index (depending on the table size)
- However since version 10, these problems have been resolved. They are now crash safe and are able to be replicated to standby server. They sometimes perform better than b-tree indexes and are also space efficient.
- The Hash index only works with equality operators that means that you can only look up for data that matches exactly. However it is now much optimized and does a much faster lookup which makes this index a bit of specialized index that can be used where equal comparison is more important. The following command is used to create a hash index:

```
CREATE INDEX name ON table USING HASH (column);
```

3.10.1.6. WIP Gist Index

- Gist or Generalized Search Tree are useful when the data to be indexed is more complex than to do a simple equate or ranged comparison like finding nearest-neighbor and pattern matching. The example of such data includes geometric data, network address comparisons and full-text searches.
- The Gist index itself provides an infrastructure to implement different strategies for indexing data such as B-trees and R-trees. The following command is used to create a hash index:

```
CREATE INDEX name ON table USING gist (column);
```

3.10.1.7. WIP SP-Gist Index

- SP-Gist or Space partitioned Gist indexes are useful when the data can be grouped into non-overlapping groupings. Like Gist index, it also provides an infrastructure for implementing different indexing strategies. However SP-Gist is non-balanced in nature and divides data in partitions, so it allows the implementation of quad-trees, k-d trees, and radix trees (tries).

3.10.1.8. WIP Gin Index

- Generalized Inverted indexes are useful in indexing data that consist of multiple elements in a single column such as arrays, json documents (jsonb) or text search documents (tsvector).

```
CREATE INDEX name ON table USING gin (column);
```

3.10.1.9. WIP BRIN Index

- Block Range Indexes are useful for large size tables that have columns with some natural sort order. The BRIN index divides the table into block ranges and keeps a summary of those blocks. This summary includes min, max values of the range. The following command is used to create a hash index:

```
CREATE INDEX name ON table USING brin (column);
```

3.10.2. TODO ORACLE:

- READ COMMITTED (the default)
- SERIALIZABLE.
  - that what Oracle call SERIALIZABLE is actually snapshot isolation.

3.10.3. TODO Big Table [GCP]

3.10.4. WIP Big Query [GCP]

Primary source:	https://cloud.google.com/files/BigQueryTechnicalWP.pdf
Learn more:	https://cloud.google.com/bigquery/streaming-data-into-bigquery

3.10.5. TODO Elastic Search

3.10.5.1. Bloom Filter

3.10.6. TODO Atlas MongoDB

Primary source:	https://www.mongodb.com/cloud/atlas

3.10.7. TODO MongoDB

3.10.8. TODO Apache Drill

3.10.9. TODO Apache Cassandra

- **Open Source:** Modern software development organizations have overwhelmingly moved to adopt open source technologies, starting with the Linux operating system and progressing into infrastructure for managing data. Open source technologies are attractive because of their affordability and extensibility, as well as the flexibility to avoid vendor lock-in. Organizations adopting open source report higher speed of innovation and faster adoption.
- **Flexible, Familiar Interface:** The Cassandra Query Language (CQL) is similar to SQL, meaning most developers should have a fairly easy time becoming familiar with it. (Here's an introduction to CQL if you need some help).
- **High Performance:** The majority of traditional databases feature a primary / secondary architecture. In these configurations, a single primary replica performs read and write operations, while secondary replicas are only able to perform read operations. Downsides to this architecture include increased latency, as well as higher costs and lower availability at scale. In Cassandra, no single node is in charge of replicating data across a cluster. Instead, every node is capable of performing all read and write operations. This improves performance and adds resiliency to the database.
- **Active Everywhere / Zero-Downtime:** Since every Cassandra node is capable of performing read and write operations, data is quickly replicated across hybrid cloud environments and geographies. In the event a node fails, users are automatically routed to the nearest healthy node. They won't even notice that a node has been knocked offline because applications behave as designed even in the event of failure. As a result, applications are always available and data is always accessible and never lost. What's more, Cassandra's built-in repair services fix problems immediately after they occur—without any manual intervention. Productivity doesn't even need to take a hit should nodes fail.
- **Scalability:** In traditional environments, scaling applications is a time-consuming and costly process typically accomplished by scaling vertically with more expensive machines. Cassandra enables you to scale horizontally by simply adding more nodes to the cluster. If, for example, four nodes can handle 200,000 transactions/second, eight nodes will be able to handle 400,000 transactions/second. (source)
- **Seamless Replication:** Today's leading enterprises are increasingly moving to multi-data center, hybrid cloud and even multi-cloud deployments to take advantage of the strengths of various deployments without getting locked into any single provider's ecosystem. Getting the most out of multi-cloud environments, however, starts with having an underlying cloud database that offers: scalability, security, performance, and availability. For these reasons, it should come as no surprise that the cloud database market is expected to grow nearly 65% each year and reach $68.9 billion by 2022.

3.10.10. WIP Apache Pig

- is a high-level platform for creating programs that run on Apache Hadoop.
- The language for this platform is called Pig Latin.[1]
- Pig can execute its Hadoop jobs in MapReduce, Apache Tez, or Apache Spark.[2]
- Pig Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for relational database management systems.
- Pig Latin can be extended using user-defined functions (UDFs) which the user can write in Java, Python, JavaScript, Ruby or Groovy[3] and then call directly from the language.

3.10.11. WIP Sparc

Primary source:	https://spark.apache.org/

- Apache Spark™ is a unified analytics engine for large-scale data processing.
- Runs Everywhere:
  - Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud. It can access diverse data sources.
  - You can run Spark using its standalone cluster mode, on EC2, on Hadoop YARN, on Mesos, or on Kubernetes. Access data in HDFS, Alluxio, Apache Cassandra, Apache HBase, Apache Hive, and hundreds of other data sources.

3.10.12. WIP Apache Oozie Workflow Scheduler for Hadoop

Primary source:	https://oozie.apache.org

- Oozie is a workflow scheduler system to manage Apache Hadoop jobs.
- Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions.
- Oozie Coordinator jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availability.
- Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, Sqoop and Distcp) as well as system specific jobs (such as Java programs and shell scripts).
- Oozie is a scalable, reliable and extensible system.

3.10.13. TODO Hadoop

Primary source:	https://en.wikipedia.org/wiki/Apache_Hadoop

3.10.14. TODO hue

Primary source:	https://gethue.com/

3.10.15. TODO Hive

Primary source: https://hive.apache.org/

3.10.16. TODO HBase

Primary source: https://hbase.apache.org/

3.10.17. TODO Nifi

3.10.18. WIP Apache Parquet

Primary source: https://en.wikipedia.org/wiki/Apache_Parquet

- free and open-source column-oriented data storage format of the Apache Hadoop ecosystem.
- It is similar to the other columnar-storage file formats available in Hadoop namely RCFile and ORC.
- It is compatible with most of the data processing frameworks in the Hadoop environment.
- It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk.

3.10.19. WIP Apache Avro

Primary source: https://avro.apache.org

- Avro is a row-oriented remote procedure call and data serialization framework developed within Apache's Hadoop project.
- It uses JSON for defining data types and protocols, and serializes data in a compact binary format.
- Its primary use is in Apache Hadoop, where it can provide both a serialization format for persistent data, and a wire format for communication between Hadoop nodes, and from client programs to the Hadoop services.
- Avro uses a schema to structure the data that is being encoded.
- It has two different types of schema languages; one for human editing (Avro IDL) and another which is more machine-readable based on JSON.[3]
- It is similar to Thrift and Protocol Buffers, but does not require running a code-generation program when a schema changes (unless desired for statically-typed languages).

3.10.20. WIP Apache ORC (Optimized Row Columnar)

- is a free and open-source column-oriented data storage format of the Apache Hadoop ecosystem.
- It is similar to the other columnar-storage file formats available in the Hadoop ecosystem such as RCFile and Parquet.
- It is compatible with most of the data processing frameworks in the Hadoop environment.
- In February 2013, the Optimized Row Columnar (ORC) file format was announced by Hortonworks in collaboration with Facebook.[3]
- A month later, the Apache Parquet format was announced, developed by Cloudera and Twitter.[4]

3.10.21. WIP Protocol Buffers (Protobuf)

- is a method of serializing structured data.
- It is useful in developing programs to communicate with each other over a wire or for storing data.
- The method involves an interface description language that describes the structure of some data and a program that generates source code from that description for generating or parsing a stream of bytes that represents the structured data.

3.10.22. TODO Liquibase

- open source
- tracks DB chanegs
- allows work with data as well as with the schema
- **changeset**:
  - XML, JSON, YAML, SQL
- can use context and label for filteriing chich chnages slould be applied where

4. WIP Methodology [0/10] [DOMAIN]

4.1. TODO Scrum [0/3] *[TOPIC]*

4.1.1. TODO DoD

4.1.2. TODO DoR

4.1.3. TODO Sprint

4.2. TODO Kanban *[TOPIC]*

4.3. TODO XP *[TOPIC]*

4.4. TODO Lean *[TOPIC]*

4.5. TODO Tools

4.5.1. TODO Jira [TOPIC]

4.5.2. TODO Confluence [TOPIC]

4.6. TODO MOSCOW *[CONCEPT]*

4.7. TODO Uml *[TOPIC]*

4.8. TODO C4 *[TOPIC]*

4.9. WIP Arc42 *[TOPIC]*

Primary source:	https://arc42.org/
Learn more:	https://www.se-radio.net/2015/12/se-radio-episode-244-gernot-starke-on-architecture-documentation-using-arc42/

4.10. WIP Kaizen (Continuous Improvement) *[TOPIC]*

- is a strategy where employees at all levels of a company work together proactively to achieve regular, incremental improvements to the manufacturing process.
- In a sense, it combines the collective talents within a company to create a powerful engine for improvement.

5. WIP Cloud [0/12] [DOMAIN]

5.1. TODO Cloud native architecture

5.2. WIP Open Policy Agent

Primary source:	https://www.openpolicyagent.org/docs/latest/
Learn more:	https://www.se-radio.net/2020/04/episode-406-torin-sandall-on-distributed-policy-enforcement/

5.3. TODO IaaS

5.4. TODO PaaS

5.5. TODO SaaS

5.6. WIP AWS Concepts *[TOPIC]*

5.6.1. Regions

- Each Amazon EC2 Region is designed to be isolated from the other Amazon EC2 Regions. This achieves the greatest possible fault tolerance and stability.
- When you view your resources, you see only the resources that are tied to the Region that you specified. This is because Regions are isolated from each other, and we don't automatically replicate resources across Regions.
- When you launch an instance, you must select an AMI that's in the same Region. If the AMI is in another Region, you can copy the AMI to the Region you're using. For more information, see Copying an AMI.

Code	Name	Opt-in Status	Local Zone
us-east-2	US East (Ohio)	Not required	Not available
us-east-1	US East (N. Virginia)	Not required	Not available
us-west-1	US West (N. California)	Not required	Not available
us-west-2	US West (Oregon)	Not required	us-west-2-lax-1a
us-west-2	US West (Oregon)	Not required	aus-west-2-lax-1b
af-south-1	Africa (Cape Town)	Required	Not available
ap-east-1	Asia Pacific (Hong Kong)	Required	Not available
ap-south-1	Asia Pacific (Mumbai)	Not required	Not available
ap-northeast-3	Asia Pacific (Osaka-Local)	Not required	Not available
ap-northeast-2	Asia Pacific (Seoul)	Not required	Not available
ap-southeast-1	Asia Pacific (Singapore)	Not required	Not available
ap-southeast-2	Asia Pacific (Sydney)	Not required	Not available

ap-northeast-1	Asia Pacific (Tokyo)	Not required	Not available
ca-central-1	Canada (Central)	Not required	Not available
eu-central-1	Europe (Frankfurt)	Not required	Not available
eu-west-1	Europe (Ireland)	Not required	Not available
eu-west-2	Europe (London)	Not required	Not available
eu-south-1	Europe (Milan)	Required	Not available
eu-west-3	Europe (Paris)	Not required	Not available
eu-north-1	Europe (Stockholm)	Not required	Not available
me-south-1	Middle East (Bahrain)	Required	Not available
sa-east-1	South America (São Paulo)	Not required	Not available

5.6.2. Availability Zones

- Each Region has multiple, isolated locations known as Availability Zones.
- When you launch an instance, you can select an Availability Zone or let us choose one for you.
- If you distribute your instances across multiple Availability Zones and one instance fails, you can design your application so that an instance in another Availability Zone can handle requests.

5.7. WIP AWS Services *[TOPIC]*

5.7.1. EC2 (Amazon Elastic Compute Cloud) [AWS]

5.7.1.1. AMI *[AWS]*

5.7.2. ELB CLB ALB [AWS]

5.7.3. Route 53 [AWS]

5.7.4. Beanstalk [AWS]

5.7.5. RDS [AWS]

5.7.6. API Gateway [AWS]

Primary source:	https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

- is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud. As an API Gateway API developer, you can create APIs for use in your own client applications. Or you can make your APIs available to third-party app developers. For more information, see Who uses API Gateway?.
- API Gateway creates RESTful APIs that:
  - Are HTTP-based.
  - Enable stateless client-server communication.
  - Implement standard HTTP methods such as GET, POST, PUT, PATCH, and DELETE.
- For more information about API Gateway REST APIs and HTTP APIs, see Choosing between HTTP APIs and REST APIs, Working with HTTP APIs, Use API Gateway to create REST APIs, and Creating a REST API in Amazon API Gateway.
- API Gateway creates WebSocket APIs that:
  - Adhere to the WebSocket protocol, which enables stateful, full-duplex communication between client and server.
  - Route incoming messages based on message content.
- For more information about API Gateway WebSocket APIs, see Use API Gateway to create WebSocket APIs and About WebSocket APIs in API Gateway.

5.7.7. AWS Lambda [AWS]

5.7.8. DynamoDB [AWS]

5.7.9. Kinesis [AWS]

5.7.10. EFS [AWS]

5.7.11. SNS [AWS]

5.7.12. S3 [AWS]

5.7.13. EBS [AWS]

5.7.14. CloudWatch [AWS]

5.7.15. CloudFormation [AWS]

5.7.16. CloudFront [AWS]

5.7.17. Secrets Manager [AWS]

5.7.18. VPC [AWS]

5.7.19. Wavelength [AWS]

- AWS Wavelength allows developers to build applications that deliver ultra-low latencies to mobile devices and end users. Wavelength deploys standard AWS compute and storage services to the edge of telecommunication carriers' 5G networks. Developers can extend an Amazon Virtual Private Cloud (VPC) to one or more Wavelength Zones, and then use AWS resources like Amazon Elastic Compute Cloud (EC2)

instances to run applications that require ultra-low latency and a connection to AWS services in the Region.

5.7.20. Elasticsearch

5.8. TODO AWS Tools and Frameworks

5.8.1. TODO boto3 [AWS]

5.8.2. TODO aws-shell [AWS]

5.8.3. TODO SAM [AWS]

5.9. TODO GCP Services [TOPIC]

5.9.1. TODO BigQuery

5.9.2. TODO PubSub

5.9.3. TODO composer

5.9.4. TODO dataflow

5.9.5. TODO automl

5.9.6. TODO dataproc

5.9.7. TODO secretmanager

5.9.8. TODO iam

5.9.9. WIP aim:service accounts

Primary source:	https://cloud.google.com/iam/docs/understanding-service-accounts

- A service account is a special type of Google account intended to represent a non-human user that needs to authenticate and be authorized to access data in Google APIs.
- Typically, service accounts are used in scenarios such as:
  - Running workloads on virtual machines (VMs).
  - Running workloads on on-premises workstations or data centers that call Google APIs.
  - Running workloads which are not tied to the lifecycle of a human user.
- Your application assumes the identity of the service account to call Google APIs, so that the users aren't directly involved.

5.9.10. WIP Firebase

- is a platform developed by Google for creating mobile and web applications.
- It was originally an independent company founded in 2011[1].
- In 2014, Google acquired the platform[2] and it is now their flagship offering for app development.
- Firebase is a toolset to “build, improve, and grow your app”, and the tools it gives you cover a large portion of the services that developers would normally have to build themselves, but don’t really want to build, because they’d rather be focusing on the app experience itself.
- This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on.
- The services are hosted in the cloud, and scale with little to no effort on the part of the developer.

5.9.10.1. What sort of apps is Firebase good for?

- There’s really no limit to the types of apps that can be helped by Firebase products. There are only limits to the platforms it can be used on. iOS and Android are the primary targets for the Firebase SDKs, and there’s increasing support for web, Flutter, Unity, and C++. You should also know there’s an Admin SDK available for a variety of languages, to be used with any backend components you might require.
- On top of those SDKs, there’s a library called FirebaseUI (Android, iOS, web) that provides a bunch of helpful utilities to make development with Firebase even easier. And there are also projects such as AngularFire that wrap the web SDKs for use with Angular. These are open source. Firebase likes open source.

5.10. TODO GCP CLI [TOPIC]

- gcloud

5.11. TODO FINOPS *[TOPIC]*

5.11.1. GCP Cost Model

5.11.2. AWS Cost Model

5.11.3. data at rest cost

5.11.4. transfer cost

5.11.5. CPU cost

5.12. WIP Storage:

5.12.1. AWS S3 [AWS]

5.12.2. GCS bucket [GCP]

5.12.2.1. api

5.12.2.2. what it is good for

5.12.3. WIP MinIO Object Storage

Primary source:	https://min.io/product/overview

6. WIP DEVOPS / SRE [0/9] [DOMAIN]

6.1. WIP Culture

Primary source:	https://blog.sonatype.com/principle-based-devops-frameworks-three-ways

6.1.1. TODO The Three Ways

Primary source:	https://blog.sonatype.com/principle-based-devops-frameworks-three-ways

The Phoenix Project/DevOps Handbook's Three Ways If you've read either The Phoenix Project or The DevOps Handbook, you've been introduced to The Three Ways framework for DevOps:

- The First Way: Principles of Flow
- The Second Way: Principles of Feedback
- The Third Way: Principles of Continuous Learning

6.1.1.1. The First Way: Principles of Flow

The First Way is mostly concerned with accelerating the "flow" of work throughout a process. Gene Kim also refers to the First Way as Systems Thinking in his article The Three Ways: Principles Underpinning DevOps. Whether you're calling it Flow or Systems Thinking, the principles underpinning the First Way are working toward the same end: viewing the flow of work as one continuous system (unsiloed) that can be continually refined and optimized.
Some of the key principles of the First Way are:

- **Making work "visible".** Unlike manufacturing processes, which are easily observable on a plant floor, the flow of software through its development lifecycle is not easily seen. Using methods such as Kanban boards can surface the activities going on behind the scenes, by showing the left-to-right movement of a user story through the development phases.
- **Limiting work-in-progress (WIP).** Keeping work-in-progress to a minimum has also been shown to accelerate work flow, because it minimizes multi-tasking and context-switching.
- **Reducing batch sizes**. "Chunking" work into smaller pieces like a two-week sprint can also help deliver features (albeit smaller ones) and bug fixes to the customer faster. Issues are often caught earlier when those updates and additions are released sooner.
- **Reducing hand-offs between teams.** The risk of "dropping the baton" increases as the hand-offs do. Although hand-offs can't be completely minimized, the key is to keep the teams in tight communication with one another so that the hand-off itself is almost a non-event rather than a large ordeal with the potential for communication missteps along the way.
- **Identifying and removing constraints and waste.** Constraints might be bottlenecks in the process, such as environments, test setup, and overly tight architecture, while waste includes things like manual work, heroics, and context-switching.

6.1.1.2. The Second Way: Principles of Feedback

The Second Way works to enable fast and constant feedback cycles throughout all stages of a development cycle.
Some of the key principles of the Second Way are:

- **Swarming and solving problems to build new knowledge.** This principle fits into the "fail fast" mentality, so that teams can find issues with an implementation as soon as possible and address them early and often as iterations continue.
- **Pushing quality closer to source.** This principle is at the core of the DevSecOps movement, which is concerned with addressing security concerns during the development cycle, instead of at the end, when rework to remediate is more difficult and costly.
- **Optimizing for downstream work centers.** This principle works against the "throw it over the wall" mentality, by underscoring that development should be just as invested in their application being deployable, working with operations to bridge that gap (and vice versa).

6.1.1.3. The Third Way: Principles of Continuous Learning

The Third Way seeks to create a culture of continual learning and experimentation within the development organization.
Some of the key principles of the Third Way are:

- **Enabling organizational learning and a safety culture.** Leaders must help "set the tone" for the organization, making it okay to learn, make mistakes, and try again.
- **Institutionalizing the improvement of daily work.** Improving what you do and how you accomplish it should be part of everyone's daily thinking and call to action.
- **Transforming local discoveries to global improvements.** Surfacing and sharing improvements at all levels will help enable a "bubble up" culture of continuous improvement.
- **Injecting resilience patterns into daily work.** Some examples might include rehearsing failures, and working toward improving key metrics for deployment.

- **Leaders enforcing a learning culture.** Organization-wide learning is unlikely to take hold and become pervasive unless it is sanctioned and exemplified by its leaders. So being intentional about communicating the value of learning and problem-solving is crucial to building that culture.

6.1.2. ACCELERATE

6.1.3. CALMS (Culture, Automation, Lean, Measurement, Sharing)

Primary source: https://blog.sonatype.com/principle-based-devops-frameworks-calms

- Created by Jez Humble, co-author of The DevOps Handbook and Accelerate, the CALMS framework is used as a means of assessing whether an organization is ready to adopt DevOps processes, or how an organization is progressing in their DevOps transformation. It is based on the following five pillars:
- **Culture.** Before silos can be torn down, there needs to be a culture of shared responsibility, or at least a group of people devoted to establishing that culture in a grassroots type of way, with management approval and support.
- **Automation.** Similar to the technical practices centered around continuous delivery mentioned above, teams undertaking a DevOps transformation should be devoted to automating as many manual tasks as possible, especially with respect to continuous integration and test automation.
- **Lean.** Development teams are making use of lean principles to eliminate waste and optimize the value stream, such as minimizing WIP, making work visible, and reducing hand-off complexity and wait times.
- **Measurement.** The organization is devoted to collecting data on their processes, deployments, etc., in order to understand their current capabilities and where improvements could be achieved.
- **Sharing.** A culture of openness and sharing within and between teams (and enabled with the proper tools) keeps everyone working toward the same goals and eases friction with hand-offs when issues arise.

6.2. WIP TCP/IP

6.2.1. IPv4 address / netmask / CIDR

6.2.1.1. Special IPv4 Ranges

Primary source: https://en.wikipedia.org/wiki/IPv4#Special-use_addresses

Address block	Address range	Number of addresses	Scope	Description
0.0.0.0/8	0.0.0.0–0.255.255.255	16 777 216	Software	Current network[3] (only valid as source address).
10.0.0.0/8	10.0.0.0–10.255.255.255	16 777 216	Private network	Used for local communications within a private network.[4]
100.64.0.0/10	100.64.0.0–100.127.255.255	4 194 304	Private network	Shared address space[5] for communications between a service provider and its subscribers when using a carrier-grade NAT.
127.0.0.0/8	127.0.0.0–127.255.255.255	16 777 216	Host	Used for loopback addresses to the local host.[3]
169.254.0.0/16	169.254.0.0–169.254.255.255	65 536	Subnet	Used for link-local addresses[6] between two hosts on a single link when no IP address is otherwise specified, such as would have normally been retrieved from a DHCP server.
172.16.0.0/12	172.16.0.0–172.31.255.255	1 048 576	Private network	Used for local communications within a private network.[4]
192.0.0.0/24	192.0.0.0–192.0.0.255	256	Private network	IETF Protocol Assignments.[3]
192.0.2.0/24	192.0.2.0–192.0.2.255	256	Documentation	Assigned as TEST-NET-1, documentation and examples.[7]
192.88.99.0/24	192.88.99.0–192.88.99.255	256	Internet	Reserved.[8] Formerly used for IPv6 to IPv4 relay[9] (included IPv6 address block 2002::/16).
192.168.0.0/16	192.168.0.0–192.168.255.255	65 536	Private network	Used for local communications within a private network.[4]
198.18.0.0/15	198.18.0.0–198.19.255.255	131 072	Private network	Used for benchmark testing of inter-network communications between two separate subnets.[10]
198.51.100.0/24	198.51.100.0–198.51.100.255	256	Documentation	Assigned as TEST-NET-2, documentation and examples.[7]
203.0.113.0/24	203.0.113.0–203.0.113.255	256	Documentation	Assigned as TEST-NET-3, documentation and examples.[7]
224.0.0.0/4	224.0.0.0–239.255.255.255	268 435 456	Internet	In use for IP multicast.[11] (Former Class D network).

Address block	Address range	Number of addresses	Scope	Description
240.0.0.0/4	240.0.0.0–255.255.255.254	268 435 455	Internet	Reserved for future use.[12] (Former Class E network).
255.255.255.255/32	255.255.255.255	1	Subnet	Reserved for the "limited broadcast" destination address.[3][13]

6.3. TODO Linux [0/36]

6.3.1. TODO cron

6.3.2. TODO runlevel / init / inittab

6.3.3. TODO systemd

6.3.4. TODO snap

6.3.5. TODO apt

6.3.6. TODO ssh

6.3.6.1. TODO how to create trust between computers

6.3.7. TODO tarpit

6.3.8. TODO awk

Primary source:	https://www.digitalocean.com/community/tutorials/how-to-use-the-awk-language-to-manipulate-text-in-linux

6.3.9. TODO sed

6.3.10. TODO grep

6.3.11. TODO tail

6.3.12. TODO head

6.3.13. TODO rsync

6.3.14. TODO mount

6.3.15. TODO ls / ps / htop

6.3.16. TODO bash [TOPIC]

6.3.16.1. here doc? *[QUESTION]*

6.3.16.2. bracket expansion? *[QUESTION]*

6.3.16.3. input/output redirection *[QUESTION]*

6.3.17. TODO regexp

6.3.18. TODO stdin stdout stderr

6.3.19. TODO list open ports

6.3.20. TODO iptables

6.3.21. TODO BPF

6.3.22. TODO list open files

6.3.23. TODO NFS

6.3.24. TODO file types:

6.3.24.1. block device

6.3.24.2. character device

6.3.24.3. named pipe

6.3.24.4. symlink

6.3.25. TODO etc/nsswitch./conf

6.3.26. TODO etc/resolv.conf

6.3.27. TODO ip

```
ip addr sh
ip route sh
```

6.3.28. TODO nmap

6.3.29. TODO nc

6.3.30. TODO socat

6.3.31. TODO chmod

6.3.32. TODO dd

6.3.33. TODO tar

6.3.34. TODO apt-get / apt / dpkg

6.3.35. TODO wget

6.3.36. TODO curl

6.3.37. TODO find

6.4. TODO Observability

6.4.1. 3 pillars

6.4.1.1. logs

6.4.1.2. metrics

6.4.1.3. traces

6.4.2. Tail Latency

6.5. TODO gitops

6.6. TODO Tools

6.6.1. Git [TOPIC]

6.6.1.1. What is the use of Staging area or Indexing in Git? *[QUESTION]*

- From Git's perspective, there are three areas where the file changes can be kept i.e.:
  - working directory,
  - staging area, and
  - repository.
- First, you make changes in your project's working directory stored on your computer file system.
- All the changes remain here until you add them to an intermediate area called staging area.
- You can stage the changes by executing `git add .`
- This staging area gives you a preview of your next commit and basically lets you fine-tune your commits.
- You can add or remove changes in the staging area until you are satisfied with the version you are going to commit.
- Once you verify your changes and sign off the stage changed, then you can finally commit the changes.
- Upon commit, they go the local repository i.e. into `.git/objects` directory.
- If you use Git GUI, then you will see the option to stage your changes.
- In the below screenshot, the file sample.txt is under unstaged changes area which means that it's in your working directory.
- Staging is also referred to as indexing because git maintains an index file to keep track of your file changes across these three areas.
- The files which are staged are currently in your index.
- When you add changes to the staging area, then the information in the index gets updated.
- When you do a commit, its actually what's in the index that gets committed, and not what's in the working directory.
- You can use the git status command to see what's in the index.

6.6.1.2. TODO squash? Waht it is? whan to use it awht it is good for?

6.6.1.3. TODO Pull request

6.6.1.4. TODO Cherry pick

6.6.1.5. TODO Flow

6.6.1.6. TODO Push

6.6.1.7. TODO Rebase vs Merge - semantical and syntactical diff

6.6.1.8. TODO Stash

6.6.2. WIP DNS [TOPIC]

Primary source: https://www.freshersemploy.com/dns-interview-questions-answers/

6.6.2.1. What is DNS?

- Domain Name Systems(DNS) maps domain names with Internet Protocol(IP) address, thus helping computer for translating human-readable(domain name) to machine-readable language(IP address).

6.6.2.2. What is Nameserver?

- It is used for storing the information for the domain name to IP and IP to the domain name. In other words, the name server is used for storing records of the domain names, Name servers help for convert domain name to IP address.

6.6.2.3. What is DNS Spoofing?

- DNS Spoofing occurs when a hacker is redirecting website traffic maliciously to spread the malware or to compromise the user's data through forged DNS record.
- When a user visits a website(like freshersemploy.com) in a browser, it needs to be resolved from human-readable to machine-readable(i.e. IP addresses) using DNS resolver. To decrease the time and resolve faster, DNS resolvers cache the data and store for a specific time(until it expires). An attacker can inject forged DNS entry, thus causing DNS Cache Poisoning and instead of freshersemploy.com, the attacker will redirect it to the wrong domain or fake website.

6.6.2.4. How can we prevent DNS Spoofing?

- DNS Spoofing or DNS Cache Poisoning can be avoided by following guidelines:
  - From a users point of view, it is almost impossible for users to detect whether the DNS records are compromised, but users should be alerted when accessing the sites like banking, payment website where credit card details are used by checking the domain name in the browser and does it have a valid SSL certificate.
  - Domain owner and DNS provider cannot avoid DNS Spoofing but surely can take protective measures to avoid the compromise of DNS records.
    - Using Domain Name System Security Extensions(DNSSEC) which helps in determining the DNS records authenticity by signing each request with certified signature and this is recommended by ICANN. Still this in process of implementation.

6.6.2.5. What is Round Robin DNS? What is the purpose of it?

- Round Robin DNS is a technique for load distribution, load balancing, fault-tolerance service such as mail server, FTP server etc. which enables istribution of load evenly among multiple servers using various DNS A addresses.
- Suppose a domain abc.com is configured with Round robin DNS which has 3 potential IP mapping to the same domain(i.e. freshersemploy.com) as follows:

```
freshersemploy.com -- 104.28.26.01
freshersemploy.com -- 104.28.26.02
freshersemploy.com -- 104.28.26.03
```

- Then the first request will go to `104.28.26.01`, the second request will go to `104.28.26.02` and the third request will go to `104.28.26.03`, thus DNS queries are responded by returning IP addresses in a rotation.

6.6.2.6. What is primary and secondary name server?

- Primary name server reads the data from the domain zone, it has DNS records of domain names and it replicates the data with the secondary name server.
- A secondary name server is the back up of primary name server which is used for high reliability, in case the primary name server is having an issue or not reachable.

6.6.2.7. What is DNS resolver?

- DNS resolvers are being used by ISP (Internet service provider) for the user request to resolve the domain name. If a user request for google.com, DNS resolver needs to contact TLD(Top Level Domain) i.e. .com, for translation of domain name to IP address and it caches the data if the user again queries for the same domain, thus reducing the loads on the server and response time.

6.6.2.8. What is the difference between URL and Domain?

- URL stands for Uniform Resource Locator. URL specifies the full address of a webpage. It consists of three components: protocol(like HTTP, mailto, ftp), domain name(like freshersemploy.com) and file name(eg. homepage.html)
- e.g.:
  - https://www.freshersemploy.com/
  - https://www.freshersemploy.com/html-interview-questions-answers/
  - mailto:admin@freshersemploy.com
- A domain is the name of a website with the top-level domain(like .com, .org etc.). A domain is a part of URL.
- e.g.:
  - https://www.freshersemploy.com/
  - admin@freshersemploy.com

6.6.2.9. What is DNS server? *[QUESTION]*

- DNS servers resolve IP address to respective hostnames and it maintains directory or database to store the information.
- When we try to access the website (like ***freshersemploy.com)***, DNS servers help to translate to machine-readable language i.e. IP address (like ***104.28.27.67***).

6.6.2.10. What are the different types of records in DNS? *[QUESTION]*

- Some of commonly used of DNS records are A, CNAME, NS, MX, PTR, SOA etc.

6.6.2.11. Explain SOA record? *[QUESTION]*

- Start of Authority(SOA) records stores essential information(like refresh rate, expiry, TTL etc) in domain name system(DNS) in a zone file.

; name TTL class rr Nameserver email-address freshersemploy.com. 12000 IN SOA ns.nameserver.com. root.ns.nameserver.com. (2098163206 ; Serial number 12000 ; Refresh rate in seconds 3600 ; Update Retry in seconds 5788864; Expiry in seconds 100 ; minimum in seconds)

```
; name TTL class rr Nameserver email-address
freshersemploy.com. 12000 IN SOA
ns.nameserver.com. root.ns.nameserver.com.
(
    2098163206 ; Serial number
    12000 ; Refresh rate in seconds
    3600 ; Update Retry in seconds
    5788864; Expiry in seconds
    100 ; minimum in seconds )
```

- **Serial Number:** It has the serial number, which gets increments whenever there is a change in the DNS records.
- **Refresh interval:** It gets refresh at the specific interval and if there any changes in the records, data is replicated.
- **Retry:** If the propagation gets failed, it will retry after specific time which is defined in the zone file.
- **Expire:** It is set to have an expiry date, as specified in the zone file. Also used for secondary server how long it should be active in case the primary DNS server is down.
- **TTL:** It has the default time-to-live(TTL),

A zone file should have only one SOA record and it must be at the top of it.

6.6.2.12. What is the use of PTR in DNS? [QUESTION]

PTR(Pointer) records are used for mapping IP addresses which are associated with hostname name. It is also called has reverse DNS lookup as it resolves IP address to domain or hostname. There must be A record for every PTR record. PTR is mainly used for the mail server.

6.6.2.13. Explain CNAME record? [QUESTION]

- CNAME record stands for Canonical Name record. It used as the alias for domain or the Canonical name(another name) for a domain.
- For example, suppose someone incorrectly types(or misspelled the website name) like fresher*s*employ.com as fresheremploy.com(we should also own this domain), then using CNAME record, we can redirect to fresher*s*employ.com
- It is a type of resource record in DNS which is specified in the zone file. CNAME records should always point to another domain and never directly points to IP address.

6.6.2.14. Explain Dynamic DNS [QUESTION]

- Dynamic DNS helps for automatically updating the name servers whenever there is a change in the IP address in the Domain Name System(DNS).

6.6.2.15. What is Resource Record? [QUESTION]

- Resource Record(RR) defines the elements or attributes of a domain name in DNS zone file like Address(A) record, Mail Exchange(MX) record etc. which helps in name resolutions.
- Some of the most common Resource Records are:
- A – IPv4 Address record
- AAAA – IPv6 Address record
- CNAME – Canonical Name
- MX – Mail Exchange
- PTR – Pointer
- SOA – Start of Authority
- NS – Name Server

6.6.2.16. What is DNS Zone? [QUESTION]

- A DNS zone file contains the mapping between a domain name, IP address, recourse records etc. in text representative format. Also, DNS zone refers to the administrative responsibility in the DNS.

6.6.2.17. Define TTL

- Time-to-live caches the DNS records for a specific period of time(when TTL expires, it has to query new record). It helps in queries the records faster, eventually reducing the load on the DNS server.

6.6.2.18. Explain MX record?

- Mail Exchange(MX) record is a type of resource record which is used for email sending and delivery. It must be specified in the DNS zone files mails for the domain.

6.6.2.19. What is Forward Lookup

- Forward Lookup is used to find the IP address through the domain name.

6.6.2.20. What is Reverse Lookup

- Reverse Lookup is used for finding the domain name through it IP.

6.6.2.21. TODO What is DKIM

6.6.3. JFrog: Artifactory

6.6.4. Nexus … sunset

6.6.5. Docker [TOPIC]

6.6.5.1. TODO dockerhub :-)

6.6.5.2. TODO docker file

6.6.5.3. TODO docker image

6.6.5.4. TODO docker compose

6.6.5.5. TODO docker swarm

6.6.5.6. TODO linux namespaces

6.6.5.7. TODO cgroups

6.6.5.8. TODO OCI

6.6.5.9. TODO volumes

6.6.5.10. TODO networks

6.6.5.11. TODO BuildKit

6.6.5.12. TODO DTR - docker trusted registry

6.6.6. Terraform [TOPIC]

6.6.6.1. resource

6.6.6.2. module

6.6.6.3. state

6.6.6.4. backend

6.6.7. Chef [TOPIC]

6.6.8. Puppet [TOPIC]

6.6.9. Ansible [TOPIC]

6.6.10. Cloud Formation [TOPIC]

6.6.11. Hashicorp: Consul [TOPIC]

6.6.12. Hashicorp: Vault [TOPIC]

6.6.13. TODO Zookeeper [TOPIC]

6.6.14. WIP Kibana [TOPIC]

Primary source: https://logz.io/blog/grafana-vs-kibana/

- Kibana is the ‘K’ in the ELK Stack, the world’s most popular open source log analysis platform, and provides users with a tool for exploring, visualizing, and building dashboards on top of the log data stored in Elasticsearch clusters.
- Kibana’s core feature is data querying and analysis. Using various methods, users can search the data indexed in Elasticsearch for specific events or strings within their data for root cause analysis and diagnostics. Based on these queries, users can use Kibana’s visualization features which allow users to visualize data in a variety of different ways, using charts, tables, geographical maps and other types of visualizations.

6.6.15. WIP Graphana [TOPIC]

Primary source: https://logz.io/blog/grafana-vs-kibana/

- Grafana is an open source visualization tool that can be used on top of a variety of different data stores but is most commonly used together with Graphite, InfluxDB, Prometheus, Elasticsearch and Logz.io. As it so happens, Grafana began as a fork of Kibana, trying to supply support for metrics (a.k.a. monitoring) that Kibana (at the time) did not provide much if any such support for.
- Essentially, Grafana is a feature-rich replacement for Graphite-web, which helps users to easily create and edit dashboards. It contains a unique Graphite target parser that enables easy metric and function editing. Users can create comprehensive charts with smart axis formats (such as lines and points) as a result of Grafana’s fast, client-side rendering — even over long ranges of time — that uses Flot as a default option.

6.6.16. TODO ELK [TOPIC]

6.6.17. TODO DynaTrace [TOPIC]

6.6.18. TODO Piwik [TOPIC]

6.6.19. TODO argocd [TOPIC]

6.6.20. TODO fluxcd [TOPIC]

6.6.21. TODO Prometheus [TOPIC]

6.7. TODO CNCF [TOPIC]

6.8. WIP K8S / Kubernetes: [0/27] [TOPIC]

Primary source:	https://classpert.com/kubernetes

6.8.1. TODO What is a service [QUESTION]

6.8.2. WIP What are network policies? [QUESTION]

Primary source:	https://github.com/ahmetb/kubernetes-network-policy-recipes

- firewall on the pod level

6.8.3. WIP what are labels?

- key value pairs that can be assigned to majority of Kubernetes objects

6.8.4. WIP What are network policies rules? [QUESTION]

1. traffic is allowed unless there is a network policy that blocks it
2. rules apply as soon as pod is selected and it is deny by default unless stated differently
3. you then only write rules that allow (whitelist)
4. rules are aditive "ored" i.e. or is applied and single allow rule is enough to open the flow
5. network policies are scoped to a namespace they are deployed to
6. ports in rules are ports on pods not ports of services

6.8.5. WIP what are podSelectors? [QUESTION]

Primary source:	https://medium.com/@zwhitchcox/matchlabels-labels-and-selectors-explained-in-detail-for-beginners-d421bdd05362

matchLabels:

6.8.6. WIP what is namespaceSelector? [QUESTION]

6.8.7. WIP what is empty selector: [QUESTION]

example: matchLabels:{}

6.8.8. WIP what ipBlock and hwhat are they good for?

6.8.9. WIP what is a network policy plugin and how you enable that?

- Calico, for example
- iptables vs overlay network (calico is iptables based)

6.8.10. WIP what is a RBAC

Primary source:	https://www.cncf.io/blog/2018/08/01/demystifying-rbac-in-kubernetes/

6.8.11. WIP network policy API

6.8.12. TODO whats is kube-dns

6.8.13. TODO helm

6.8.14. TODO chart

6.8.15. TODO template

6.8.16. TODO cluster

6.8.17. TODO pod

6.8.18. TODO node

6.8.19. TODO service mesh

6.8.20. TODO ingress controller

6.8.21. TODO ingress and exgress?

6.8.22. TODO etcd

6.8.23. TODO secret

6.8.24. TODO operator

6.8.25. WIP CRD / Custom Resources

- A resource is an endpoint in the Kubernetes API that stores a collection of API objects of a certain kind; for example, the built-in pods resource contains a collection of Pod objects.
- A custom resource is an extension of the Kubernetes API that is not necessarily available in a default Kubernetes installation. It represents a customization of a particular Kubernetes installation. However, many core Kubernetes functions are now built using custom resources, making Kubernetes more modular.
- Custom resources can appear and disappear in a running cluster through dynamic registration, and cluster admins can update custom resources independently of the cluster itself. Once a custom resource is installed, users can create and access its objects using kubectl, just as they do for built-in resources like Pods.

6.8.26. TODO liveness/readines probes

6.8.27. TODO kubectl

6.9. WIP Concepts [0/6]

6.9.1. TODO ChatOps [CONCEPT]

Primary source: https://www.pagerduty.com/blog/what-is-chatops/

- ChatOps, a term widely credited to GitHub, is all about conversation-driven development.While in a chat room, team members type commands that the chatbot is configured to execute through custom scripts and plugins. These can range from code deployments, to security event responses, to team member notifications. By bringing your tools into your conversations and using a chatbot modified to work with key plugins and scripts, teams can automate tasks and collaborate, working better, cheaper and faster—allowing the entire team to collaborate in real time as commands are executed.

6.9.2. TODO Rugged DevOps [CONCEPT]

Primary source: https://ruggedsoftware.org/

- The Rugged Manifesto:
  - I am rugged and, more importantly, my code is rugged.
  - I recognize that software has become a foundation of our modern world.
  - I recognize the awesome responsibility that comes with this foundational role.
  - I recognize that my code will be used in ways I cannot anticipate, in ways it was not designed, and for longer than it was ever intended.
  - I recognize that my code will be attacked by talented and persistent adversaries who threaten our physical, economic, and national security.
  - I recognize these things - and I choose to be rugged.
  - I am rugged because I refuse to be a source of vulnerability or weakness.
  - I am rugged because I assure my code will support its mission.
  - I am rugged because my code can face these challenges and persist in spite of them.
  - I am rugged, not because it is easy, but because it is necessary and I am up for the challenge.

6.9.3. TODO Tail Latency [CONCEPT]

6.9.4. TODO SLI [CONCEPT]

6.9.5. TODO SLO [CONCEPT]

6.9.6. TODO SLA [CONCEPT]

7. WIP Web Front End [0/9] [DOMAIN]

7.1. WIP React

7.1.1. TODO React

7.1.2. TODO React Native

7.1.3. TODO react-hooks

7.1.4. TODO react-styled-components

7.2. TODO Relay

7.3. TODO Apollo

7.4. TODO Next.js

7.5. TODO Angular

7.6. WIP PWA

7.6.1. app-shell model

7.7. WIP Web API

7.7.1. Server-send events

- Traditionally, a web page has to send a request to the server to receive new data; that is, the page requests data from the server.
- With server-sent events, it's possible for a server to send new data to a web page at any time, by pushing messages to the web page.
- These incoming messages can be treated as Events + data inside the web page.

7.7.2. Tools

7.7.2.1. SoapUI

7.7.2.2. curl

7.8. WIP HTML + CSS [/]

7.8.1. Chrome Development Tools

7.8.2. HTML5 Layout tags

```
<header> It is used to define a header for a document or a section.
<nav> It is used to define a container for navigation links
<section> It is used to define a section in a document
<article> It is used to define an independent, self-contained article
<aside> It is used to define content aside from the content (like a sidebar)
<footer> It is used to define a footer for a document or a section
```

7.8.3. What is iframe good for? [QUESTION]

An iframe is used to display a web page within a web page. Syntax:

```
<iframe src="URL"></iframe>
```

Example:

```
<iframe src="demo_iframe.html" width="200px" height="200px"></iframe>
```

Target to a link:

```
<iframe src="http://www.javatpoint.com" name="iframe_a"></iframe>
```

7.8.4. TODO Google Fonts API

Primary source:	https://developers.google.com/fonts

7.8.5. WIP Cookies

Primary source:	https://tools.ietf.org/id/draft-ietf-httpbis-rfc6265bis-03.html

7.8.5.1. COMPETE Cookie Types

- Two Types of Cookies
- **Session cookies**
  - do not contain an expiration date. Instead, they are stored only as long as the browser or tab is open. As soon as the browser is closed,

they are permanently lost.
    - This type of cookie might be used to store a banking user's credentials while they are navigating within their bank's website since their information would be forgotten as soon as the tab is closed.
- **Persistent cookies**
    - do have an expiration date.
    - These cookies are stored on the user's disk until the expiration date and then permanently deleted.
    - They can be used for other activities such as recording a user's habits while on a particular website in order to customize their experience every time they visit.

7.8.6. TODO Local Storage [TOPIC]

7.8.7. WIP Cookies / SameSite [TOPIC]

Primary source: https://web.dev/samesite-cookies-explained/

7.8.8. TODO CSS preprocessors: LESS Sass

Primary source: https://www.lambdatest.com/blog/css-preprocessors-sass-vs-less-vs-stylus-with-examples/

- What Is A CSS Preprocessor?
    - CSS in itself is devoid of complex logic and functionality which is required to write reusable and organized code. As a result, a developer is bound by limitations and would face extreme difficulty in code maintenance and scalability, especially when working on large projects involving extensive code and multiple CSS stylesheets. This is where CSS Preprocessors come to the rescue.
    - A CSS Preprocessor is a tool used to extend the basic functionality of default vanilla CSS through its own scripting language. It helps us to use complex logical syntax like – variables, functions, mixins, code nesting, and inheritance to name a few, supercharging your vanilla CSS. By using CSS Preprocessors, you can seamlessly automate menial tasks, build reusable code snippets, avoid code repetition and bloating and write nested code blocks that are well organized and easy to read.
    - However, browsers can only understand native vanilla CSS code and will be unable to interpret the CSS Preprocessor syntax. Therefore, the complex and advanced Preprocessor syntax needs to be first compiled into native CSS syntax which can then be interpreted by the browsers to avoid cross browser compatibility issues. While different Preprocessors have their own unique syntaxes, eventually all of them are compiled to the same native CSS code.
    - Moving forward in the article, we will take a look at the 3 most popular CSS Preprocessors currently being used by developers around the world i.e Sass, LESS, and Stylus.

7.8.9. TODO Sass

Primary source: https://sass-lang.com/guide

7.8.10. TODO Less

7.8.11. WIP CORS [0/5]

Primary source: https://www.codecademy.com/articles/what-is-cors
Learn more: https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l

7.8.11.1. WIP what is CORS *[QUESTION]*

- Cross-Origin Resource Sharing (CORS) is a mechanism that uses HTTP headers to specify:
    - which outer origin have access to the local assets and
    - how can access it
    - that is mean we can make a white list for the allowed Cross-Origins that has access to our assets.

7.8.11.2. WIP How CORS works? *[QUESTION]*

- when the site makes a get request to get resource from the out server, the browser adds a header that contains the origin like the example Origin: http://www.example.com.
- The server receives the preflight request and searches in its white-list for Access-Control-Allow-Origins about the giving origin and sends to the browser option call, then the browser will determine if the actual request is safe to send or not, example Access-Control-Allow-Origin: http://www.example.com or this header Access-Control-Allow-Origin: * will allow any request to take the resource.
- if the server specifies the Methods it will compare the request method with its example Access-Control-Allow-Methods: PUT, DELETE.

7.8.11.3. TODO Why is CORS necessary? *[QUESTION]*

7.8.11.4. TODO How Does CORS Manage Requests From External Resources? *[QUESTION]*

7.8.11.5. WIP What http headers were added by CORS? *[QUESTION]*

```
Access-Control-Allow-Origin
Access-Control-Allow-Credentials
Access-Control-Allow-Headers
Access-Control-Allow-Methods
Access-Control-Expose-Headers
Access-Control-Max-Age
Access-Control-Request-Headers
Access-Control-Request-Method
Origin
```

7.8.12. WIP Content Security Policy (CSP)

Primary source: https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l

- Content Security Policy is more security layer that helps in detect and mitigate different sort of militias attacks like (Cross-Site Scripting (XSS), data injection attacks, ClickJacking, ETC…).
    - **Cross-Site Scripting (XSS)**: it a vulnerability that allows the hacker to inject a militias code in the base website and it is for making the

client execute it to take sensitive data like cookies, session's info and site-specific information, That happens because web app does not use enough validation or encoding, The user's browser cannot detect the malicious script is untrustworthy.
- **data injection attacks**: is a malicious code injected in the network which fetched all the information from the database to the attacker and the number one type of it is the SQL injection.
- **Click Jacking**: or "*UI redress attack*" is when an attacker tricks a user into clicking on a button or link on another page that uses multiple transparent or opaque layers when he intended to click on the top-level.

7.8.12.1. How CSP works?

- it use Directives concept that's every Directive have to specify where resources can load from, preventing browsers from loading data from any other locations.
- Most used Directives are:
  - default-src: the default policy for loading (JavaScript, images, CSS, AJAX requests, ETC…) example `default-src 'self' cdn.example.com;`
  - img-src: defines sources for images example `img-src 'self' img.example.com;`
  - style-src: defines sources for CSS files example `style-src 'self' css.example.com;`
  - script-src: defines sources for JavaScript files example `script-src 'self' js.example.com;`
  - connect-src: defines valid targets for to XMLHttpRequest (AJAX), WebSockets or EventSource, If it makes any connections to a host, that's not allowed here, the browser will respond with a 400 error example `connect-src 'self';`
  - Multi-label directives defines: `default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src 'self';`

7.8.13. WIP Bundlers

Primary source:	https://bundlers.tooling.report/

7.8.14. WIP Webpack

Primary source:	https://webpack.js.org/concepts/

- At its core, webpack is a static module bundler for modern JavaScript applications.
- When webpack processes your application, it internally builds a dependency graph which maps every module your project needs and generates one or more bundles.

7.9. TODO The Elm Architecture (TEA)

- http://blog.jenkster.com/2016/06/how-elm-slays-a-ui-antipattern.html

8. WIP Security `[0/14]` [DOMAIN]

8.1. WIP Crypto `[0/5]`

8.1.1. WIP Block Ciphers

8.1.1.1. ECB, CBC, OFB, CFB, CTR - what they are

Primary source:	https:/www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm//

8.1.1.2. RSA

8.1.1.3. AES

8.1.1.4. DES

8.1.2. WIP Hash

base64

8.1.3. TODO Symetric Encryption

8.1.4. TODO Asymetric Encryption

8.1.5. WIP HMAC

Primary source:	https://en.wikipedia.org/wiki/HMAC

- In cryptography, an HMAC (sometimes expanded as either keyed-hash message authentication code or hash-based message authentication code) is a specific type of message authentication code (MAC) involving a cryptographic hash function and a secret cryptographic key.
- As with any MAC, it may be used to simultaneously verify both the data integrity and the authenticity of a message.

8.2. WIP SPIFFE [TOPIC]

8.2.1. what is spiffe

Primary source:	https://spiffe.io/docs/latest/spiffe/overview/

- SPIFFE, the Secure Production Identity Framework for Everyone, is a set of open-source standards for securely identifying software systems in dynamic and heterogeneous environments. Systems that adopt SPIFFE can easily and reliably mutually authenticate wherever they are running.
- Distributed design patterns and practices such as micro-services, container orchestrators, and cloud computing have led to production environments that are increasingly dynamic and heterogeneous. Conventional security practices (such as network policies that only allow traffic between particular IP addresses) struggle to scale under this complexity. A first-class identity framework for workloads in an organization becomes necessary.
- Further, modern developers are expected to understand and play a role in how applications are deployed and managed in production environments. Operations teams require deeper visibility into the applications they are managing. As we move to a more evolved security stance, we must offer better tools to both teams so they can play an active role in building secure, distributed applications.

- SPIFFE is a set of open-source specifications for a framework capable of bootstrapping and issuing identity to services across heterogeneous environments and organizational boundaries. The heart of these specifications is the one that defines short lived cryptographic identity documents – called SVIDs via a simple API. Workloads can then use these identity documents when authenticating to other workloads, for example by establishing a TLS connection or by signing and verifying a JWT token.

8.3. WIP Oauth2/OIDC *[TOPIC]*

8.3.1. grant (flow) types [/]

8.3.1.1. TODO Authorization Code Flow

8.3.1.2. TODO Client Credentials Flow

8.3.1.3. TODO Device Code

8.3.1.4. TODO Refresh Token

8.3.1.5. TODO PKCE

8.3.1.6. TODO (Implicit Flow)

8.3.1.7. TODO (Password Grant)

8.3.1.8. WIP Resource Owner Password Credential Grant

- from soapUI template
- requested attributes:
  - Resource Owner Name
  - Resource Owner Password
  - Client identification
  - Client Secret
  - Access Token URI:
  - scope
- example for netIq

```
curl --request POST \
 --url https://<idphost:port>/nidp/oauth/nam/token \
 --header 'content-type: application/x-www-form-urlencoded' \
 --data 'grant_type=password&client_id=bb775b12-bbd4-423b-83d9-
647aeb98608d&client_secret=bBbE4mNO_kWWAnEeOL1CLTyuPhNLhHkTThArEckyrdLmRLn3GhnxjsKI2mEijCSlPjftxHod05dpuGs6wA&u
sername=user1&password=pass@123&scope=email%20profile'
```

8.3.2. Scope

A mechanism that defines the specific actions applications can be allowed to do or information that they can request on a user's behalf. Often, applications will want to make use of the information that has already been created in an online resource. To do so, the application must ask for authorization to access this information on a user's behalf. When an app requests permission to access a resource through an authorization server, it uses the Scope parameter to specify what access it needs, and the authorization server uses the Scope parameter to respond with the access that was actually granted.

8.3.3. What is acr_values

Primary source: https://ldapwiki.com/wiki/Acr_values

- Acr_values is an OPTIONAL parameter that is a Space-separated string that specifies the Authentication Context Class Values within the Authentication Request that the Authorization Server is being requested to use for processing requests from this Client, with the values appearing in order of preference.
- Acr_values was originally specified within the JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants (RFC 7523)
- The Authentication Context Class Reference values actually utilized by the Authorization Server is returned within the Identity Token in the acr parameter.

8.4. WIP JWT *[TOPIC]*

Primary source: https://tools.ietf.org/html/rfc7519

8.4.1. What are the registered claim names? [QUESTION]

8.4.2. What is registered claim name "iss" (Issuer) Claim? [QUESTION]

- The "iss" (issuer) claim identifies the principal that issued the JWT.
- The processing of this claim is generally application specific.
- The "iss" value is a case-sensitive string containing a StringOrURI value.
- Use of this claim is OPTIONAL.

8.4.3. What is registered claim name "sub" (Subject) Claim [QUESTION]

- The "sub" (subject) claim identifies the principal that is the subject of the JWT.
- The claims in a JWT are normally statements about the subject.
- The subject value MUST either be scoped to be locally unique in the context of the issuer or be globally unique.
- The processing of this claim is generally application specific.
- The "sub" value is a case-sensitive string containing a StringOrURI value.
- Use of this claim is OPTIONAL.

8.4.4. What is registered claim name "aud" (Audience) Claim [QUESTION]

- The "aud" (audience) claim identifies the recipients that the JWT is intended for.
- Each principal intended to process the JWT MUST identify itself with a value in the audience claim.
- If the principal processing the claim does not identify itself with a value in the "aud" claim when this claim is present, then the JWT MUST be

rejected.
- In the general case, the "aud" value is an array of case-sensitive strings, each containing a StringOrURI value.
- In the special case when the JWT has one audience, the "aud" value MAY be a single case-sensitive string containing a StringOrURI value.
- The interpretation of audience values is generally application specific.
- Use of this claim is OPTIONAL.

8.4.5. What is registered claim name "exp" (Expiration Time) Claim [QUESTION]

- The "exp" (expiration time) claim identifies the expiration time on or after which the JWT MUST NOT be accepted for processing.
- The processing of the "exp" claim requires that the current date/time MUST be before the expiration date/time listed in the "exp" claim.
- Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew.
- Its value MUST be a number containing a NumericDate value.
- Use of this claim is OPTIONAL.

8.4.6. What is registered claim name "nbf" (Not Before) Claim [QUESTION]

- The "nbf" (not before) claim identifies the time before which the JWT MUST NOT be accepted for processing.
- The processing of the "nbf" claim requires that the current date/time MUST be after or equal to the not-before date/time listed in the "nbf" claim.
- Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew.
- Its value MUST be a number containing a NumericDate value.
- Use of this claim is OPTIONAL.

8.4.7. What is registered claim name "iat" (Issued At) Claim [QUESTION]

- The "iat" (issued at) claim identifies the time at which the JWT was issued.
- This claim can be used to determine the age of the JWT.
- Its value MUST be a number containing a NumericDate value.
- Use of this claim is OPTIONAL.

8.4.8. What is registered claim name "jti" (JWT ID) Claim [QUESTION]

- The "jti" (JWT ID) claim provides a unique identifier for the JWT.
- The identifier value MUST be assigned in a manner that ensures that there is a negligible probability that the same value will be accidentally assigned to a different data object;
- if the application uses multiple issuers, collisions MUST be prevented among values produced by different issuers as well.
- The "jti" claim can be used to prevent the JWT from being replayed.
- The "jti" value is a case-sensitive string.
- Use of this claim is OPTIONAL.

8.4.9. What are Public Claim Names [QUESTION]

- Claim Names can be defined at will by those using JWTs.
- However, in order to prevent collisions, any new Claim Name should either be registered in the IANA "JSON Web Token Claims" registry established by Section 10.1 or be a Public Name:
  - a value that contains a Collision-Resistant Name.
- In each case, the definer of the name or value needs to take reasonable precautions to make sure they are in control of the part of the namespace they use to define the Claim Name.

8.4.10. RS256 vs HS256: What's the difference?

Primary source: https://stackoverflow.com/questions/39239051/rs256-vs-hs256-whats-the-difference

- Both choices refer to what algorithm the identity provider uses to sign the JWT. Signing is a cryptographic operation that generates a "signature" (part of the JWT) that the recipient of the token can validate to ensure that the token has not been tampered with.
  - **RS256 (RSA Signature with SHA-256)** is an asymmetric algorithm, and it uses a public/private key pair: the identity provider has a private (secret) key used to generate the signature, and the consumer of the JWT gets a public key to validate the signature. Since the public key, as opposed to the private key, doesn't need to be kept secured, most identity providers make it easily available for consumers to obtain and use (usually through a metadata URL).
  - **HS256 (HMAC with SHA-256)**, on the other hand, involves a combination of a hashing function and one (secret) key that is shared between the two parties used to generate the hash that will serve as the signature. Since the same key is used both to generate the signature and to validate it, care must be taken to ensure that the key is not compromised.
- If you will be developing the application consuming the JWTs, you can safely use HS256, because you will have control on who uses the secret keys. If, on the other hand, you don't have control over the client, or you have no way of securing a secret key, RS256 will be a better fit, since the consumer only needs to know the public (shared) key.
- Since the public key is usually made available from metadata endpoints, clients can be programmed to retrieve the public key automatically. If this is the case (as it is with the .Net Core libraries), you will have less work to do on configuration (the libraries will fetch the public key from the server). Symmetric keys, on the other hand, need to be exchanged out of band (ensuring a secure communication channel), and manually updated if there is a signing key rollover.
- Auth0 provides metadata endpoints for the OIDC, SAML and WS-Fed protocols, where the public keys can be retrieved. You can see those endpoints under the "Advanced Settings" of a client.
- The OIDC metadata endpoint, for example, takes the form of https://{account domain}/.well-known/openid-configuration.
  - If you browse to that URL, you will see a JSON object with a reference to https://{account domain}/.well-known/jwks.json, which contains the public key (or keys) of the account.
- If you look at the RS256 samples, you will see that you don't need to configure the public key anywhere: it's retrieved automatically by the framework.

8.5. WIP JWKS

Primary source: https://auth0.com/docs/tokens/json-web-tokens/json-web-key-sets

- JSON Web Key Sets:
  - The JSON Web Key Set (JWKS) is a set of keys containing the public keys used to verify any JSON Web Token (JWT) issued by the authorization server and signed using the RS256 signing algorithm.
  - When creating applications and APIs in Auth0, two algorithms are supported for signing JWTs: RS256 and HS256.
    - RS256 generates an asymmetric signature, which means a private key must be used to sign the JWT and a different public key must be used to verify the signature.
  - Auth0 uses the JSON Web Key (JWK) specification to represent the cryptographic keys used for signing RS256 tokens. This specification defines two high-level data structures: JSON Web Key (JWK) and JSON Web Key Set (JWKS). Here are the definitions from the

specification:

Item	Description
JSON Web Key (JWK)	A JSON object that represents a cryptographic key. The members of the object represent properties of the key, including its value.
JSON Web Key Set (JWKS)	A JSON object that represents a set of JWKs. The JSON object MUST have a keys member, which is an array of JWKs.

Auth0 exposes a JWKS endpoint for each tenant, which is found at https://YOUR_DOMAIN/.well-known/jwks.json. This endpoint will contain the JWK used to sign all Auth0-issued JWTs for this tenant.

8.6. WIP LDAP

8.6.1. object class

8.6.1.1. inetOrgPerson

8.6.2. OID

8.6.3. CN, DN, SN,

8.7. WIP OWASP top 10 2017

Primary source:	https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/

8.7.1. A1:2017-Injection

8.7.2. A2:2017-Broken Authentication

8.7.3. A3:2017-Sensitive Data Exposure

8.7.4. A4:2017-XML External Entities (XXE)

8.7.5. A5:2017-Broken Access Control

8.7.6. A6:2017-Security Misconfiguration

8.7.7. A7:2017-Cross-Site Scripting (XSS)

8.7.8. A8:2017-Insecure Deserialization

8.7.9. A9:2017-Using Components with Known Vulnerabilities

8.7.10. A10:2017-Insufficient Logging & Monitoring

8.8. TODO (OWASP) attacks

8.8.1. WIP CSRF / XSRF

Primary source:	https://owasp.org/www-community/attacks/csrf
Learn more:	https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
Learn more:	http://www.adambarth.com/papers/2008/barth-jackson-mitchell-b.pdf
Learn more:	https://devnet.kentico.com/articles/protection-against-cross-site-request-forgery-%28csrf-xsrf%29

8.9. TODO bastillion.io

8.10. TODO PKI

8.11. WIP X509

8.11.1. Why does GoDaddy have four different certificate chain, G2, G3, G4? [QUESTION]

- "G" stands for Root certificates generation.
- Basically, it identifies the generation (version) of a Root certificate that signs the Chain of trust.
- When CA needs to get a new chain (for example, because of upgrade from SHA-1 to SHA-256), they just increment the generation number.
- This is good because obsolete certificates can be better identified.

8.12. WIP Concepts [0/2]

8.12.1. WIP DevSecOps [CONCEPT]

Primary source:	https://snyk.io/devsecops/
Learn more:	https://snyk.io/devsecops/

- DevSecOps manifesto:
- Through Security as Code, we have and will learn that there is simply a better way for security practitioners, like us, to operate and contribute value with less friction. We know we must adapt our ways quickly and foster innovation to ensure data security and privacy issues are not left behind because we were too slow to change.
- By developing security as code, we will strive to create awesome products and services, provide insights directly to developers, and generally favor iteration over trying to always come up with the best answer before a deployment. We will operate like developers to make security and compliance available to be consumed as services. We will unlock and unblock new paths to help others see their ideas become a reality.
- We won't simply rely on scanners and reports to make code better. We will attack products and services like an outsider to help you defend what

you've created. We will learn the loopholes, look for weaknesses, and we will work with you to provide remediation actions instead of long lists of problems for you to solve on your own.
- We will not wait for our organizations to fall victim to mistakes and attackers. We will not settle for finding what is already known; instead, we will look for anomalies yet to be detected. We will strive to be a better partner by valuing what you value:
- Manifesto:

Leaning in	over	Always Saying “No”
Data & Security Science	over	Fear, Uncertainty and Doubt
Open Contribution & Collaboration	over	Security-Only Requirements
Consumable Security Services with APIs	over	Mandated Security Controls & Paperwork
Business Driven Security Scores	over	Rubber Stamp Security
Red & Blue Team Exploit Testing	over	Relying on Scans & Theoretical Vulnerabilities
24x7 Proactive Security Monitoring	over	Reacting after being Informed of an Incident
Shared Threat Intelligence	over	Keeping Info to Ourselves
Compliance Operations	over	Clipboards & Checklists

- What Is DevSecOps?
  - DevSecOps refers to the integration of security practices into a DevOps software delivery model.
  - Its foundation is a culture where development and operations are enabled through process and tooling to take part in a shared responsibility for delivering secure software.
  - The definition of DevSecOps, at a high-functioning level, is to integrate security objectives as early as possible in the lifecycle of software development.
  - While security is “everyone’s responsibility,” DevOps teams are uniquely positioned at the intersection of development and operations, empowered to apply security in both breadth and depth.

8.12.2. TODO Zero Trust Network [CONCEPT]

8.13. WIP Software Composition Analysis (SCA) [0/2]

Primary source:	https://sudonull.com/post/27911-Forrester-Research-A-Comparison-of-Ten-Top-Software-Composition-Analysis-Vendors

8.13.1. TODO Snyk

8.13.2. TODO Jfrog Xray

8.14. WIP Tools & Solutions [0/9]

8.14.1. TODO Illumio Security

8.14.2. TODO Burp Suite

8.14.3. TODO Metasploit

8.14.4. TODO wireshark

8.14.5. TODO hydra

8.14.6. TODO nmap

8.14.7. TODO nessus

9. WIP Programming Languages [0/13] [DOMAIN]

9.1. WIP Functional Paradigm [0/8] *[TOPIC]*

9.1.1. WIP Lambda Calculus [TOPIC]

9.1.1.1. α-conversion
- changing bound variables

9.1.1.2. β-reduction
- applying functions to their arguments;

9.1.1.3. η-reduction
- which captures a notion of extensionality.

9.1.2. TODO Immutability [TOPIC]

9.1.3. TODO Idempotency

9.1.4. TODO Continuation Passing Style

9.1.5. TODO Persistent data structure (Chris Okasaki)

9.1.6. TODO Algebraic Data Types

9.1.7. WIP Patterns

9.1.7.1. TODO Monoid

9.1.7.2. TODO Functor

9.1.7.3. TODO Pro Functor

9.1.7.4. TODO Applicative

9.1.7.5. TODO Monad

9.1.7.6. TODO Kleisli Arrow

9.1.7.7. TODO Free Monad

9.1.7.8. TODO Fix

Primary source:	https://www.parsonsmatt.org/2016/10/26/grokking_fix.html

9.1.7.9. Lenses

9.1.7.10. Prisms

9.1.8. TODO Recursion Schemes

9.1.8.1. catamorphisms

9.1.8.2. anamorphisms

9.1.8.3. hylomorphisms

9.1.8.4. paramorphisms

9.2. WIP Object Oriented Paradigm [TOPIC]

9.2.1. Polymorphism

9.2.2. Gof Patterns

9.2.2.1. Creational Patterns

Pattern Name	Description
Singleton	The singleton pattern restricts the initialization of a class to ensure that only one instance of the class can be created.
Factory	The factory pattern takes out the responsibility of instantiating a object from the class to a Factory class.
Abstract Factory	Allows us to create a Factory for factory classes.
Builder	Creating an object step by step and a method to finally get the object instance.
Prototype	Creating a new object instance from another similar instance and then modify according to our requirements.

9.2.2.2. Structural Patterns

- There are 7 structural design patterns defined in the Gangs of Four design patterns book.

Pattern Name	Description
Adapter	Provides an interface between two unrelated entities so that they can work together.
Composite	Used when we have to implement a part-whole hierarchy. For example, a diagram made of other pieces such as circle, square, triangle, etc.
Proxy	Provide a surrogate or placeholder for another object to control access to it.
Flyweight	Caching and reusing object instances, used with immutable objects. For example, string pool.
Facade	Creating a wrapper interfaces on top of existing interfaces to help client applications.
Bridge	The bridge design pattern is used to decouple the interfaces from implementation and hiding the implementation details from the client program.
Decorator	The decorator design pattern is used to modify the functionality of an object at runtime.

9.2.2.3. Behavioral Patterns

- There are 11 behavioral design patterns defined in the GoF design patterns.

Pattern Name	Description
Template Method	used to create a template method stub and defer some of the steps of implementation to the subclasses.
Mediator	used to provide a centralized communication medium between different objects in a system.
Chain of Responsibility	used to achieve loose coupling in software design where a request from the client is passed to a chain of objects to process them.
Observer	useful when you are interested in the state of an object and want to get notified whenever there is any change.
Strategy	Strategy pattern is used when we have multiple algorithm for a specific task and client decides the actual implementation to be used at runtime.
Command	Command Pattern is used to implement lose coupling in a request-response model.
State	State design pattern is used when an Object change it's behavior based on it's internal state.
Visitor	Visitor pattern is used when we have to perform an operation on a group of similar kind of Objects.
Interpreter	defines a grammatical representation for a language and provides an interpreter to deal with this grammar.
Iterator	used to provide a standard way to traverse through a group of Objects.
Memento	The memento design pattern is used when we want to save the state of an object so that we can restore later on.

9.3. WIP Java *[TOPIC]*

9.3.1. hashCode()

9.3.1.1. is this code wrong? yes / no / why *[QUESTION]*

Imagine you replace all haskCode implementation by this - what will happen

```
@Override
public final int hashCode() {
    int result = 17;
    return result;
}
```

9.3.2. hashCode vs equals() [QUESTION]

9.3.3. what is a contract? [QUESTION]

9.3.4. Java Collections Framework:

9.3.4.1. what is a difference between List vs Set vs Map *[QUESTION]*

9.3.4.2. what is the difference ArrayList vs LinkedList *[QUESTION]*

9.3.5. StringBuilder vs StringBuffer vs String

9.3.6. Can we have static method in the interface? [QUESTION]

- Yes, we can have static method in the interface from Java 8.

9.3.7. You have a list of Custom objects? How can you sort them? [QUESTION]

You need to use Comparable or Comparator interface to sort list of custom objects.

9.3.8. How to create thread safe Singleton in Java [QUESTION]

Primary source:	https://javarevisited.blogspot.com/2012/12/how-to-create-thread-safe-singleton-in-java-example.html

- use double checked locking
- use class loader
- use enum
- Read more: https://javarevisited.blogspot.com/2012/12/how-to-create-thread-safe-singleton-in-java-example.html#ixzz6Y1Y1v8C1

9.3.9. Difference between List and Set in Java? [QUESTION]

- hint: List is ordered and allows duplicate. Set is unordered and doesn't allow duplicate elements.

9.3.10. How do you prevent a class from being sub-classed in Java? [QUESTION]

- just make its constructor private

9.3.11. Difference between throw and throws keyword in Java? [QUESTION]

- throws declare what exception a method can throw in case of error but throw keyword actually throws an exception.
- See Java Fundamentals: Exception Handling to learn more about Exception handling in Java.

9.3.12. Difference between Iterator and Enumeration in Java? [QUESTION]

- Iterator also gives you the ability to remove an element while iterating while Enumeration doesn't allow that.

9.3.13. What is IdentityHashMap in Java? [QUESTION]

- A Map, which uses the == equality operator to check equality instead of the equals() method.

9.3.14. What is String pool in Java? [QUESTION]

- A pool of String literals. Remember it's moved to heap from perm gen space in JDK 7.

9.3.15. Can a Serializable class contain a non-serializable field in Java? [QUESTION]

- Yes, but you need to make it either static or transient.

9.3.16. Difference between this and super in Java? [QUESTION]

- this refers to the current instance while super refers to an instance of the superclass.

9.3.17. Difference between Comparator and Comparable in Java? [QUESTION]

- Comparator defines custom ordering while Comparable defines the natural order of objects, e.g. the alphabetic order for String. See The Complete Java MasterClass to learn more about sorting in Java.

9.3.18. BigDecimal vs float [QUESTION]

9.3.19. what is the difference betwen checked and unchecked exception? [QUESTION]

9.3.20. Co všechno je špatně na tomto kusu kódu [QUESTION]

```
try {
  // ...
} catch (Exception e) {
  log.error("Oh no!", e.getMessage());
}
```

- Zdaleka největší průšvih je, že jsme přišli o stack trace!
  - Musíme nechat zalogovat celý objekt "e" a nebo aspoň přímo jeho ST.
- Chytáme příliš obecný typ, takže nemůžeme nijak rozlišit mezi jednotlivými typy chyb.
- Kombinace chytání příliš obecné chyby a neudělání re-throw, je velmi nebezpečná.
  - Pokud nechceme dělat re-throw, měli bychom chytat jen ten typ, kterého se to týká.
- A další věci, jako stupidní message "Oh no!", atd.

9.3.21. Explain Java 7 ARM (Automatic Resource Managemet) Feature and multi-catch block? [QUESTION]

- If you are catching a lot of exceptions in a single try block, you will notice that catch block code looks very ugly and mostly consists of redundant code to log the error,
- keeping this in mind Java 7 one of the feature was multi-catch block where we can catch multiple exceptions in a single catch block.
- The catch block with this feature looks like below:

```
catch(IOException | SQLException | Exception ex){
    logger.error(ex);
    throw new MyException(ex.getMessage());
}
```

- Most of the time, we use finally block just to close the resources and sometimes we forget to close them and get runtime exceptions when the resources are exhausted.
- These exceptions are hard to debug and we might need to look into each place where we are using that type of resource to make sure we are closing it.
- So java 7 one of the improvement was try-with-resources where we can create a resource in the try statement itself and use it inside the try-catch block.
- When the execution comes out of try-catch block, runtime environment automatically close these resources.
- Sample of try-catch block with this improvement is:

```
try (MyResource mr = new MyResource()) {
    System.out.println("MyResource created in try-with-resources");
} catch (Exception e) {
    e.printStackTrace();
}
```

9.3.22. TODO Generics

9.3.23. TODO Java 9 - project jigsaw

9.3.24. TODO JPA

9.3.25. TODO QueryDSL

9.3.26. WIP Maven

Jaký je rozdíl mezi <dependencies> a <dependencyManagement>? [QUESTION]

- dependencyManagement má smysl je v multi-module projektu, konkrétně v otcovském pomu.
- Používá se pro deklarování dependencí a jejich scopů a verzí a každý sub-modul si pak jen řekne, které z nich chce používat (a už nemusí deklarovat scope a verzi).
- dependencies v otcovském pomu znamenají, že každý sub-modul všechny takové dependence automaticky dostane.
- Používá se to např. pro testovací balíky nebo nějaké velmi obecně použitelné commons balíky a jiné pomocné tooly jako Lombok.

K čemu se používá scope provided? [QUESTION]

- Jde o dependence, které chceme při kompilaci, ale nechceme na runtimu
- Většinou jde o dependence, které nám v JavaEE poskytne aplikační server
- Nebo o tooly jako Lombok, které potřebujeme při kompilaci a nepotřebujeme na runtimu

9.3.27. TODO Gradle

9.3.28. JavaEE / JakartaEE

9.3.28.1. access intent

optimistic vs pessimistic

9.3.28.2. Jaký je rozdíl mezi Java SE a JavaEE *[QUESTION]*

- Java SE se používá hlavně pro nějaké procesy, které se nastartují, provedou nějakou činost a skončí
- JavaEE se používá hlavně pro aplikace, které jsou nasazené na nějakém trvale běžícím serveru a vystavují buď nějaké služby nebo přímo front end.

9.3.28.3. Co je to aplikační server a nějaký příklad *[QUESTION]*

9.3.29. Implemetations

9.3.29.1. GraalVM

9.3.29.2. Quarcus

9.3.29.3. AdoptOpenJDK

9.3.30. JDBC

9.3.31. JMS

9.4. WIP JavaScript *[TOPIC]*

9.4.1. what is IIFE? [QUESTION]

- An Immediately Invoked Function Expression is a good way at protecting the scope of your function and the variables within it.

9.4.2. inheritance model of JS

9.4.2.1. What is template based inheritance

9.4.3. hoisting

9.4.4. event loop

9.4.5. promisses

9.4.6. observables

9.4.7. RxJS

9.4.8. JS modular system

9.4.9. The Mostly Adequate Guide to Functional Programming

Primary source:	https://github.com/MostlyAdequate/mostly-adequate-guide

9.5. WIP TypeScript *[TOPIC]*

9.6. WIP ReasonML *[TOPIC]*

9.7. TODO Python

9.8. TODO OCaml

9.9. TODO Haskell

9.10. TODO Scala

9.11. TODO JS on backend

9.12. TODO JS environment

9.12.1. npm

9.12.1.1. Scoped Packages

Primary source:	https://docs.npmjs.com/about-scopes

About scopes

- Note: You must be using npm version 2 or greater to use scopes. To upgrade to the latest version of npm, on the command line, run npm install npm@latest -g
- When you sign up for an npm user account or create an Org, you are granted a scope that matches your user or Org name. You can use this scope as a namespace for related packages.
- A scope allows you to create a package with the same name as a package created by another user or Org without conflict.
- When listed as a dependent in a package.json file, scoped packages are preceded by their scope name. The scope name is everything between the @ and the slash:
- “npm” scope:

```
@npm/package-name
```

- “npmcorp” scope:

```
@npmcorp/package-name
```

- To create and publish public scoped packages, see “Creating and publishing scoped public packages”.
- To create and publish private scoped packages, see “Creating and publishing private packages”.
- Scopes and package visibility§
  - Unscoped packages are always public.
  - Private packages are always scoped.
  - Scoped packages are private by default; you must pass a command-line flag when publishing to make them public.
- For more information on package scope and visibility, see “Package scope, access level, and visibility”.

9.12.2. yarn

9.12.3. nvm

9.12.4. WIP Babel

Primary source: https://babeljs.io/

9.13. WIP Language theory *[TOPIC]*

9.13.1. Garbage Collector

9.13.2. Type systems

9.13.2.1. strongly typed

9.13.2.2. weakly typed

9.13.3. Parameter passing

9.13.3.1. Pass by Reference

9.13.3.2. Pass By Name

9.13.3.3. Pass By Value

9.13.4. Recursion

9.13.4.1. Tail call Optimization

9.13.5. Evaluation type

9.13.5.1. Lazy Evaluation

9.13.5.2. Strict Evaluation

10. WIP Libraries and Frameworks [DOMAIN]

10.1. WIP Spring & Spring Boot

Primary source: https://www.baeldung.com/spring-interview-questions

10.1.1. Jaký je rozdíl mezi JavaEE a Springem? [QUESTION]

- JavaEE je standard, který potřebuje aplikační server (JBoss, WebSphere, etc.) jakožto implementaci a runtime prostředí.
- Spring je framework, který roztáčí vlastní Spring Context, což je taková obdoba kontejneru v JavaEE a v něm všechno běží.

10.1.2. Difference between spring and spting boot [QUESTION]

10.1.3. Co přinesl Spring Boot do Spring ekosystému? [QUESTION]

- Strašně zjednodušený start aplikace.
- Vše je velmi opinionated a není potřeba napsat ani jednu řádku konfigurace, protože všechno má nějaký "rozumný default".
- Aplikace může být standalone JAR proces, WAR nebo JAR s embedded Tomcatem.
- V každém případě je to one liner.
- Plus mnoho "starter" dependencí, kterými jde snadno přidat další moduly a tooly tímto opinionated stylem.

10.1.4. Jaký je hlavní rozdíl v transaction handlingu mezi JavaEE (od verze 3) a Springem? [QUESTION]

- JavaEE to má implicitní a Spring explicitní
- V JavaEE deklarujeme EJB a ta má všechny své public metody automaticky v transakci v nějakém defaultním módu
- Ve Springu deklarujeme beanu nějakého typu a automaticky nikdy žádné transakce nemá.
- Zapínáme si je explicitně pomocí anotace.

10.1.5. How Would You Enable Transactions in Spring and What Are Their Benefits? [QUESTION]

There are two distinct ways to configure Transactions:

- with **annotations** or
- by using **Aspect Oriented Programming** (AOP) each with their advantages.

The benefits of using Spring Transactions, according to the official docs, are:

- Provide a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO
- Support declarative transaction management
- Provide a simpler API for programmatic transaction management than some complex transaction APIs such as JTA
- Integrate very well with Spring's various data access abstractions

10.1.6. What is the difference betweek JPA and Hibernate? [QUESTION]

- JPA is a Data Access Abstraction used to reduce the amount of boilerplate code
- Hibernate is an implementation of Java Persistence API and offers benefits of loose coupling

10.1.7. Can use jetty instead of tomcat in spring-boot-starter-web? [QUESTION]

- Yes, we can use jetty instead of tomcat in spring-boot-starter-web, by removing the existing dependency and including the following:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

10.1.8. What Is Reactive Programming? [QUESTION]

- Reactive programming is about:
  - non-blocking,
  - event-driven applications that
  - scale with a small number of threads, with
  - back pressure being a key ingredient that aims to ensure producers don't overwhelm consumers.
- The primary benefits of reactive programming are:
- increased utilization of computing resources on multicore and multi-CPU hardware
- and increased performance by reducing serialization
- Reactive programming is generally event-driven, in contrast to reactive systems, which are message-driven.
- Thus, using reactive programming does not mean we're building a reactive system, which is an architectural style.

However, reactive programming may be used as a means to implement reactive systems if we follow the Reactive Manifesto, which is quite vital to understand.

- Based on this, reactive systems have four important characteristics:
  - Responsive: the system should respond in a timely manner
  - Resilient: in case the system faces any failure, it should stay responsive
  - Elastic: reactive systems can react to changes and stay responsive under varying workload
  - Message-driven: reactive systems need to establish a boundary between components by relying on asynchronous message passing

10.1.9. What Is Spring Webflux? [QUESTION]

- Spring WebFlux is Spring's reactive-stack web framework, and it's an alternative to Spring MVC.
- In order to achieve this reactive model and be highly scalable, the entire stack is non-blocking. Check out our tutorial on Spring 5 WebFlux for additional details.

10.1.10. What Are the Mono and Flux Types? [QUESTION]

- The WebFlux framework in Spring Framework 5 uses Reactor as its async foundation.
- This project provides two core types: Mono to represent a single async value, and Flux to represent a stream of async values.
- They both implement the `Publisher` interface defined in the Reactive Streams specification.
- Mono implements `Publisher` and returns 0 or 1 elements:

```
public abstract class Mono<T> implements Publisher<T> {...}
```

Also, Flux implements `Publisher` and returns N elements:

```
public abstract class Flux<T> implements Publisher<T> {...}
```

- By definition, the two types represent streams,
  - hence they're both lazy, which means nothing is executed until we consume the stream using the subscribe() method.
  - Both types are immutable, therefore calling any method will return a new instance of Flux or Mono.

10.1.11. What Is the Use of Webclient and Webtestclient? [QUESTION]

- **WebClient** is a component in the new Web Reactive framework that can act as a reactive client for performing non-blocking HTTP requests.
  - Being a reactive client, it can handle reactive streams with back pressure, and it can take full advantage of Java 8 lambdas.
  - It can also handle both sync and async scenarios.
- On the other hand, the **WebTestClient** is a similar class that we can use in tests.
  - Basically, it's a thin shell around the WebClient. It can connect to any server over an HTTP connection. It can also bind directly to WebFlux applications using mock request and response objects, without the need for an HTTP server.

10.1.12. What Are the Disadvantages of Using Reactive Streams? [QUESTION]

The major disadvantages of using reactive streams are:

- Troubleshooting a Reactive application is a bit difficult; be sure to check out our tutorial on debugging reactive streams for some handy debugging tips
- There is limited support for reactive data stores, as traditional relational data stores have yet to embrace the reactive paradigm
- There's an extra learning curve when implementing

10.1.13. Is Spring 5 Compatible With Older Versions of Java? [QUESTION]

In order to take advantage of Java 8 features, the Spring codebase has been revamped. This means older versions of Java cannot be used. Hence, the framework requires a minimum of Java 8.

10.1.14. Can We Use Both Web Mvc and Webflux in the Same Application? [QUESTION]

- As of now, Spring Boot will only allow either Spring MVC or Spring WebFlux, as Spring Boot tries to auto-configure the context depending on the

dependencies that exist in its classpath.
- Also, Spring MVC cannot run on Netty.
- Moreover, MVC is a blocking paradigm and WebFlux is a non-blocking style, therefore we shouldn't be mixing both together, as they serve different purposes.

10.1.15. What Is the Role of the @Autowired Annotation? [QUESTION]

- The @Autowired annotation can be used with fields or methods for injecting a bean by type.
- This annotation allows Spring to resolve and inject collaborating beans into your bean.

10.1.16. "Spring 4.3 přinesl constructor-based dependency injection i bez nutnosti anotace Autowired. Jaké jsou výhody a nevýhody takového DI?" [QUESTION]

- "Lépe se to testuje - máme větší kontrolu nad dependencemi a můžeme je mockovat dle libosti a velmi snadno.
- Současně tím v testech vynucujeme nějakou inicializaci těchto dependencí. Jinak by klidně mohly být null.
- Je to takový mnohem přirozenější a intuitivnější způsob inicializace v OOP světě.
- Asi jediná nevýhoda: pokud bude dependencí hodně, začne to být trochu nepřehledné."

10.2. TODO NodeJS [TOPIC]

11. WIP Quality and Testing [0/7] [DOMAIN]

11.1. TODO Blue green deployment

11.2. TODO AB testing

11.3. TODO Code coverage

11.4. TODO Cyclomatic complexity [TOPIC]

11.5. TODO Code quality [TOPIC]

11.6. TODO Unit testing

11.7. WIP Tools [1/2]

11.7.1. WIP Jest

Primary source: https://jestjs.io/

- Jest is a delightful JavaScript Testing Framework with a focus on simplicity.
- It works with projects using: Babel, TypeScript, Node, React, Angular, Vue and more!
- universal
- lambdas can be tested
- all in one:
  - orchestrator
  - assertion library (expect)
  - report generator
  - react-test-libraries
- contains Istambul as a JS coverage tool

11.7.2. Enzyme [OUTDATED]

Primary source: https://enzymejs.github.io/enzyme/

- Enzyme is a JavaScript Testing utility for React that makes it easier to test your React Components' output.
- You can also manipulate, traverse, and in some ways simulate runtime given the output.
- Enzyme's API is meant to be intuitive and flexible by mimicking jQuery's API for DOM manipulation and traversal.

11.7.3. COMPETE Enzyme vs react-testing-library

:learning_1: https://blog.logrocket.com/enzyme-vs-react-testing-library-a-mindset-shift/

- With react-testing-library, you're able to easily write tests that represent well enough how the application is experienced by users.
- Let's say that when you write your tests with react-testing-library, you're testing your application as if you were the user interacting with the application's interface.
- On the other hand, when you're writing your tests with Enzyme, even though you are also able to achieve the same level of confidence that you might get with react-testing-library, it is a bit more cumbersome to build your test structure in a way that resembles a real user.
- In general, what you might see in codebases when looking at tests with Enzyme is that you're actually testing the props and state of your components, meaning you are testing the internal behavior of components to confirm that the correct view is being presented to users.

11.7.4. Jasmine [OUTDATED]

11.7.5. Istambul [OUTDATED]

Primary source: https://istanbul.js.org/

- Istanbul instruments your ES5 and ES2015+ JavaScript code with line counters, so that you can track how well your unit-tests exercise your codebase.
- The nyc command-line-client for Istanbul works well with most JavaScript testing frameworks: tap, mocha, AVA, etc.

11.7.6. Karma (Orchestrator) [OUTDATED]

11.7.7. Mocha [OUTDATED]

11.7.8. FantomJS [OUTDATED]

11.7.9. Protractor

11.7.10. Cucumber

12. WIP Learning Channels [0/6] [DOMAIN]

- where to learn, what to read and follow

12.1. TODO YouTube

12.2. TODO quicklabs

12.3. TODO Udemy

12.4. TODO Podcasts

12.4.1. TODO InfoQ

12.4.2. TODO Coding Blocks

12.4.3. TODO SE-Radio

12.5. WIP Conferences

12.5.1. StrangeLoop

Primary source:	https://www.thestrangeloop.com/

12.5.2. DEVOXX

12.5.3. GOTO

12.5.4. NDC

12.5.5. QCon

12.5.6. Beauty in Code

12.5.7. Bifrost 2018

- Re Imagine The Hiring Process

12.5.8. DevOneConf 2018

- Making Security an integrated part of the SW Dev Lifecycle

12.6. WIP Books

12.6.1. Gene Kim, Patrick Debois, John Willis, Jez Humble: The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations [AUDIBLE : BOOK]

- Increase profitability, elevate work culture, and exceed productivity goals through DevOps practices….

12.6.2. Daniel Kahneman: Thinking, Fast and Slow [AUDIBLE : BOOK]

- The guru to the gurus at last shares his knowledge with the rest of us….

12.6.3. Nassim Nicholas Taleb: The Black Swan, Second Edition: The Impact of the Highly Improbable: [AUDIBLE : BOOK]

- (With a new section: "On Robustness and Fragility": Incerto, Book 2)
  - A black swan is a highly improbable event with three principal characteristics: It is unpredictable;
  - it carries a massive impact; and, after the fact, we concoct an explanation that makes it appear less random, and more predictable, than it was….

12.6.4. Nassim Nicholas Taleb: Fooled by Randomness: The Hidden Role of Chance in Life and in the Markets [AUDIBLE : BOOK]

- Series: Incerto
  - This audiobook is about luck, or more precisely, how we perceive and deal with luck in life and business….

12.6.5. Nassim Nicholas Taleb: Antifragile: Things That Gain from Disorder [AUDIBLE : BOOK]

- Series: Incerto
  - From the best-selling author of The Black Swan and one of the foremost thinkers of our time, a book on how some things actually benefit from disorder. Taleb's message is revolutionary: What is not antifragile will surely perish….

12.6.6. Nassim Nicholas Taleb: Skin in the Game: Hidden Asymmetries in Daily Life [AUDIBLE : BOOK]

- Series: Incerto
  - From the New York Times best-selling author of The Black Swan, a bold new work that challenges many of our long-held beliefs about risk and reward, politics and religion….

12.6.7. Barry O'Reilly: Unlearn: Let Go of Past Success to Achieve Extraordinary Results[AUDIBLE : BOOK]

- By intentionally and routinely applying the system of unlearning, you'll be able to adapt your mindset, adopt new behaviors, acquire new skills, and explore new options that will totally transform your performance and the business you lead….

12.6.8. Stuart Firestein: Failure: Why Science Is so Successful [AUDIBLE : BOOK]

- Stuart Firestein shows us that the scientific enterprise is riddled with mistakes and errors - and that this is a good thing!…

12.6.9. Nate Silver: The Signal and the Noise: Why So Many Predictions Fail - but Some Don't[AUDIBLE : BOOK]

- Drawing on his own groundbreaking work, Silver examines the world of prediction, investigating how we can distinguish a true signal from a universe of noisy data….

12.6.10. Philip Tetlock , Dan Gardner: Superforecasting: The Art and Science of Prediction[AUDIBLE : BOOK]

From one of the world's most highly regarded social scientists, a transformative book on the habits of mind that lead to the best predictions….

12.6.11. Daniel Coyle: The Culture Code: The Secrets of Highly Successful Groups[AUDIBLE : BOOK]

- In The Culture Code, Daniel Coyle goes inside some of the world's most successful organizations - including
- Pixar, and the US Navy's SEAL Team Six - and reveals what makes them tick….

12.6.12. Andy Greenberg: Sandworm: A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers [AUDIBLE : BOOK]

- From Wired senior writer Andy Greenberg comes the true story of the most devastating cyberattack in history and the desperate hunt to identify and track the elite Russian agents behind it….

12.6.13. Philip E. Tetlock: Expert Political Judgment: How Good is it? How can We Know?[AUDIBLE : BOOK]

- The intelligence failures surrounding the invasion of Iraq dramatically illustrate the necessity of developing standards for evaluating expert opinion….

12.6.14. Steven Pinker: The Stuff of Thought: Language as a Window into Human Nature[AUDIBLE : BOOK]

- In The Stuff of Thought, Steven Pinker marries two of the subjects he knows best: language and human nature. The result is a fascinating look at how our words explain our nature….

12.6.15. Michael A Britt: Psych Experiments: From Pavlov's dogs to Rorschach's inkblots, put psychology's most fascinating studies to the test [DEAD_TREE : BOOK]

Forget the labs and lecture halls. You can conduct your very own psych experiments at home!
Famous psychological experiments–from Freud's ego to the Skinner box–have changed the way science views human behavior. But how do these tests really work? In Psych Experiments, you'll learn how to test out these theories and experiments for yourself…no psychology degree required!
Guided by Michael A. Britt, creator of popular podcast The Psych Files, you can conduct your own experiments when browsing your favorite websites (to test the "curiosity effect"), in restaurants (learning how to increase your tips), when presented with advertisements (you'd be surprised how much you're influenced by the color red), and even right on your smartphone (and why you panic when you can't find it). You'll even figure out how contagious yawning works!
With this compulsively readable little book, you won't just read about the history of psychology–you'll live it!

13. WIP Psychology [DOMAIN]

13.1. Cognitive Biases [0/5] *[TOPIC]*

13.1.1. WIP The Backfire Effect

- The Misconception: When your beliefs are challenged with facts, you alter your opinions and incorporate the new information into your thinking.
- The Truth: When your deepest convictions are challenged by contradictory evidence, your beliefs get stronger.

13.1.2. TODO Confirmation bias

13.1.3. TODO Loss aversion

13.1.4. TODO Parkinson's law of triviality

- The tendency to give disproportionate weight to trivial issues. Also known as bikeshedding, this bias explains why an organization may avoid specialized or complex subjects, such as the design of a nuclear reactor, and instead focus on something easy to grasp or rewarding to the average participant, such as the design of an adjacent bike shed.

13.1.5. TODO Pygmalion effect

13.2. Famous experiments [0/5] *[TOPIC]*

13.2.1. TODO Milgram experiment

13.2.2. TODO Robbers Cave Experiment

13.2.3. TODO Little Albert Experiment

13.2.4. TODO Stanford Prison Study

13.2.5. TODO The Marshmallow Test

13.3. Glossary *[TOPIC]*

13.3.1. WIP Knowledge

- definition
  - practical and/or theoretical understanding of a topic
- obtained throughput
  - learning training experience

13.3.2. WIP Skill

- definition
  - things that one learns through training and/or through experience
- obtained throughput
  - practicing, learning

13.3.3. WIP Ability

- definition
  - level of being able to perform or deliver a task
- obtained throughput
  - natural capability

14. WIP Big Names [0/27] [DOMAIN]

- Important people, people that are worth to follow

14.1. TODO Bartosz MILEWSKI [0/1]

14.1.1. TODO Opening keynote - Fun with categories - Bartosz Milewski

Authors	Bartosz MILEWSKI
Source	YouTube
Subject	Opening keynote - Fun with categories - Bartosz Milewski

https://youtu.be/MXbJPe30ONQ

14.1.2. TODO Category Theory II 8.1: F-Algebras, Lambek's lemma

Authors	Bartosz MILEWSKI
Source	YouTube
Subject	Category Theory II 8.1: F-Algebras, Lambek's lemma

https://youtu.be/zkDVCQiveEo

14.2. TODO Venkat SUBRAMANIAM

14.3. TODO Hannah FRY

14.3.1. Hello World

14.4. TODO Trisha GEE

14.5. WIP Kevlin HENNEY

- interresting and usually funny conference talks
- GOTO 2018 • Old Is the New New • Kevlin Henney
- Kevlin Henney - 1968

14.6. WIP Michael PLÖD [0/2]

14.6.1. TODO Webinar: Overview and Core Values of Domain-Driven Design - Part 1/5

Authors	Michael PLÖD
Subject	Webinar: Overview and Core Values of Domain-Driven Design - Part 1/5

https://youtu.be/Z0zpSB85XGs

14.6.2. TODO Webinar: Internal Building Blocks: Aggregates, Entities, Value Objects Part 3/5

Authors	Michael PLÖD
Source	YouTube
Subject	Webinar: Internal Building Blocks: Aggregates, Entities, Value Objects Part 3/5

Url	https://github.com/mploed/ddd-with-spring
Youtube	https://youtu.be/RHg53wMflCc

14.7. TODO Mark REINHOLD

- Modern Java: Change is the Only Constant by Mark Reinhold
- Java Lead architect

14.8. TODO Brian GOETZ

14.9. TODO Troy HUNT

14.10. TODO Scott MEYERS

14.11. TODO Erik MEIJER

14.12. TODO Edward KMETT [0/1]

14.12.1. TODO Stop Treading Water: Learning to Learn by Edward Kmett

Authors	Edward KMETT
Source	YouTube
Subject	Stop Treading Water: Learning to Learn by Edward Kmett

https://youtu.be/j0XmixCsWjs

14.13. TODO Chris ALLEN

14.14. TODO Martin FOWLLER

14.15. TODO Martin KLEPPMANN

- Book: Designing data intensive applications

14.16. TODO Kent BECK

14.17. TODO Zuzana SOCHOVA

14.18. TODO Brian LONSDORF (dr Boolean)

- https://twitter.com/drboolean
- https://github.com/DrBoolean
- https://egghead.io/instructors/brian-lonsdorf

14.19. TODO Chris RICHARDSON [0/3]

- YOW! Conference 2018 - Chris Richardson - Events and Commands
- Developing Microservices with Aggregates by Chris Richardson

14.19.1. TODO There is No Such Thing as a Microservice! - Chris Richardson January 2018

Subject	There is No Such Thing as a Microservice! - Chris Richardson January 2018

14.20. TODO Elizabeth STOKOE

14.21. WIP Linda RISING

- Linda Rising is an independent consultant who lives near Nashville, Tennessee. Linda has a Ph.D. from Arizona State University in object-based design metrics. Her background includes university teaching as well as work in industry in telecommunications, avionics, and tactical weapons systems. She is an internationally known presenter on topics related to agile development, patterns, retrospectives, the change process, and the connection between the latest neuroscience and software development. Linda is the author of numerous articles and five books. The latest, More Fearless Change, co-authored with Mary Lynn Manns. Her web site is: www.lindarising.org
- Meeting resistance and moving forward — Linda Rising
- SE-Radio 238: Linda Rising on the Agile Brain

14.22. WIP Siren Hofvander

- twitter: @securitypony
- Making Security an integrated part of the SW Dev Lifecycle

14.23. TODO Eugenia CHENG

14.24. TODO David SPIVAK

14.25. TODO Steven PINKER [0/0]

14.26. TODO Nassim TALEB

14.27. TODO Marek Orko VÁCHA

15. WIP Core Values [0/3] [DOMAIN]

The moral aspects

15.1. TODO Code of Conduct [TOPIC]

15.2. WIP SW developer responsibility [TOPIC]

- finally the code that was driving the engine in diesel gate was written by someone
  - -> those developer(s) must have known what they were doing

15.3. WIP ACM ethics Code

16. WIP Concepts and Terms [0/15] [DOMAIN]

:CUSTOM_ID: CONCEPTS

16.1. TODO Regression to the Mean [CONCEPT]

16.2. TODO Fat Tail Distribution [CONCEPT]

16.3. TODO Data Driven Decision [CONCEPT]

16.4. TODO Shift Left [CONCEPT]

- like shift left security

16.5. TODO Immutability [CONCEPT]

16.6. TODO I-Shaped vs T-shaped / Π-shaped [CONCEPT]

16.7. TODO Chaos Engineering [CONCEPT]

16.8. TODO Intrinsic vs Extrinsic Motivation [CONCEPT]

16.9. TODO Leading vs Lagging Indicators [CONCEPT]

16.10. TODO Declarative vs Imperative [CONCEPT]

16.11. TODO Strict vs Lazy Evaluation [CONCEPT]

16.12. TODO Back-pressure Strategies [CONCEPT]

Primary source:	https://medium.com/@jayphelps/backpressure-explained-the-flow-of-data-through-software-2350b3e77ce7

16.13. TODO Overqualified Query [CONCEPT]

16.14. TODO Access Intent [CONCEPT]

16.15. TODO Skepticism [CONCEPT]

17. WIP Commented Resources [CHAPTER]

17.1. WIP Videos [/] [CHAPTER]

17.1.1. YouTube [0/14]

17.1.1.1. WIP GOTO 2018 • Old Is the New New • Kevlin Henney

Youtube	https://youtu.be/AbgsfeGvg3E

- https://youtu.be/AbgsfeGvg3E
- 1968: year full of news
- Algol 68 - to big to ambitious expression oriented. If and if. Has closure.
- Not a new ideas.
- Singleton single malt whiskey
- System has either zero or multiple singleton
- Rian Foote: Patterns are aggressive disregard of originality.
- Richard Gabriel: worse is better in 1990.
  - Develop with minimal invention.
  - Better to have underfeatute software that is rock solid
- Simplicity
- Completeness
- C2 wiki was 150 lines of perl code.
- Correctness:
- Consistent
- 1968 NATO science conference. Software development process is iterative
- 1968 afips. Influential conference..
- Improve usability by making the software Faster.
- The fastest io is no io
- Command line tools can be 235x faster than your Hadoop cluster
- 299792458 is in no configuration file.
- can you fit your code in your cognitive limit
- lion's commentary of Unix 6th edition 9500 lines is a complete os.
  - there are Java classes bigger than that
- Stonehenge as an enterprise software.
  - Mono lithos one huge stone difficult to move
- UNIX philosophy.

    - White one thing that does one thing and does it well.
- 11 years of sun cycle reflect the SW industry cycle.
- write programs to work together
- ask Alfred Aho. Tdd on 1970 th
- Ward Cunningham oopsla 92 technical dept
- Jack W. Reeves:
    - programming is a design activity
- Alan Kay:
    - oop means only messaging,local retention and protection and hiding of state-process, and extreme late-binding of all things
- William Cook lambda calculus was the first object-oriented language (1941) 1932
- Simon Peyton-Jones (mr. Haskell):
    - excel is the world's most popular functional language
- coordinatio languages: separate computation and coordination
    - Unix pipe 1964
    - Ken Thompson 1970 implementace pipe in Unix

17.1.1.2. WIP Kevlin Henney - 1968

Authors	Kevlin Henney
Quality	worth returning to it
Source	YouTube
Youtube	https://youtu.be/v-RkKohsF78

https://youtu.be/v-RkKohsF78

- Pražské jaro.
- Jimmy Hendrix
- Led Zeppelin
- Facebook in 1969
- NATO congerence in Garmish (1968)
- Peter McBride: software craftsmanship
- Apollo project Margaret Heafield Hamilton inventés the software engineering term
- AFIPS conference proceedings vol. 33 1968
    - PERT .. createss only a paper
    - most deadly think in SW: you specify what you are going to do and then you do it an
    - design process is an iterative one
    - unit test, modularity
    - Patterns: aggressive disregard to originality
- Christopher Alexander: notes on the synthesis of form
    - book published in 1964
    - referenced in many later works in system / subsystem thinking
- Melvin Conway: how committees invent
    - Conway's law mandatory to mention on each conference
    - returns every 10 years
    - communication structures define system design
- Adam Thornhill:
    - your code as a crime scene (book)
    - talks about Conway's law.. based on much more data than the original Conway definition
- Stonehenge..
    - as a legacy system..
    - how did they do it..
    - we don't know
    - there is no documentation
    - Doug Mcllroy: this is the UNIX philosophy:
        - write programs that do one thing and do it well
        - write programs to work together
        - isn't this microservice definition
- Alan Perlis (first recipient of the Turing award)
    - a software system can be best designed if the testing is interlaced with the design instead of being used after the design
- bi-quinary coded decimal system
    - first system used in early computers
- ALGOL 68
    - int, long, void, bool, char, short, union, struct
    - born shell was inspired by algol
    - { did not exist at that time keyboard
    - actually try to write { on Scandinavian keyboard it is worse than. Emacs
    - start to look like lisp..
    - TDD in algol 68
- TDD instituted in 70 th wit AWK
- Alfred Aho : we instituted rigorous regression tests for all of the features of AWK. Any of the three of us who put in new feature first had to write a test for the new feature
- simula 67
    - definition os class and instance
- pragmatic bullshit
    - we use pragmatic Agile
    - pragmatic functional
    - pragmatic structured programming
- lisp….

17.1.1.3. WIP YOW! Conference 2018 - Chris Richardson - Events and Commands

Subject	YOW! Conference 2018 - Chris Richardson - Events and Commands

https://youtu.be/fhZwzm-d9ys

- REST

- gRPC
- saga
- CQRS
- event-sourcing
- replication-lag
- transactional messaging
- temporaril coupling
- api-composition-pattern
- transactional outbox pattern
- transactional-log-tailing
- wal

Microservices.io Eventuate.io Microservices.io/book (ctwyowconf18)

- Saga - 1987 paper Hector Garcia-Molina and Kenneth Salem
  - use chains of small transactions coordinated by asynchronous messages
- mainly
  - microservices are more than REST
  - microservices are more than events
  - microservices != Event sourcing
  - Apache Kafka != Event store
- service definition (contains of following parts)
  1. API
     - operation (both sync or asynchronous)
       - commands (CRUD)
       - queries
     - Events
       - event publisher
  2. has a dedicated database / table / schema
     - might contain replica of data of another service
  3. is a API client
     - invokes operations
  4. is event subscriber
- service enable
  - continuous delivery / deployment
  - testability and deployability
  - loosely coupled teams
- no acid transactiona position the easy w
  - no distributed transactions!
- sagas are ACD not Acid
- asynchronous messaging manages the Saga completion owing at least one delivery while preserving order
- Saga step = s transaction local to service
- 2pc is not an option
- sequencing Saga transactions
  - after completion of transaction T(I) "something" must decide what step to execute next
    - success: which T(I+1) … Branching
    - failure: C(i-1)
- coordination
  - choreography: **distributed** decision making
    - simple.. for primitive cases. Higher connaissance cyclic dependencies … Overloads domain objectsviin. I minimal
  - orchestration: **centralized** decision making
    - orchestration objects implements state machine and invokes the participants
- queries
  - API compositions pattern
    - to difficult to implement in many cases
  - CQRS
    - mouthful way of saying: maintain a replica of the data in a format that makes it easy to query (use elastic search or mongodb)
      - the view is maintained up to date by subscribing to the events emitted by the services that own the data … Update the replica
      - commands
        - POST
        - PUT
        - DELETE
      - Queries
        - GET
          - you can use even different types of databases per type of view
  - replication lag (handling)
    - has to be handled by the UI side
- implementing transactional messaging
  - publish to message broker first
    - only in very simple cases
  - use even sourcing
    - event sourcing = event centric approach to business logic and persistence
    - persist object as a sequence of events
      - replay events to re-enter create in memory state
      - sidenote: Apache Kafka != Event store
    - Event sourcing guarantees: state change -> event is published
  - traditional persistence + transactional outbox pattern
    - transactional log tailing
    - polling - but it introduces latency

17.1.1.4. TODO Developing Microservices with Aggregates by Chris Richardson

Subject	Developing Microservices with Aggregates by Chris Richardson
Youtube	https://youtu.be/gR_EGN86fvg

17.1.1.5. WIP Re Imagine The Hiring Process

Subject	Re Imagine The Hiring Process
Youtube	https://youtu.be/TbZ57raC1TE

- Elle Meredith
- Lachlan Hardy
- HumanResources
- Diversity
- Inclusion
- Culture
- Unconscious-bias
- Privilege
- Oppression
- Fluency-heuristic
- Collaborative-friction
- Diversity-dept
- Cultural-fit

Mark s. Luckie http://www.getluckie.net/ https://hbr.org/product/an-everyone-culture-becoming-a-deliberately-developmental-organization/14259-HBK-ENG https://www.youtube.com/channel/UCuVsMYBwV0gJTsWRZ2Dj7iw/
There is more CEO called David than womens CEO

- Homogeneous teams feels more fluent - it is just a bias.
- You must start diverse to keep a diverse team
- Hiring is a part of the work (should be)
- Language of the job postings:
  - competitive terms are more appealing to man than to woman (hitting deadline, competitive work)
    - Cooperating or teamwork in job posting: you get more women appliance
- Whiteboard coding consider harmful
- Organizations that call themselves meritocratic are actually more likely to discriminate
- Have a supportive environment for diversity and beginners
- psychological safety
  - emotionally secure work environment where people bring their whole selves to work
  - people trust each others
  - don't blame people for mistakes
- gradual iterative improvement
  - process are improved to deliver long term benefits rather than focus on short term results
  - change is expected as are mistakes
  - learn fast learn often
- good communication is vital
  - discussions are better than assumptions
  - clear expectations are defined then adjusted as necessary
  - feedback is not just welcomed sought out
  - diverse perspectives are actively engaged with
- life is larger than work
  - flexibility in when and where work is done
  - engagement and support for employee and customer communities
- shared values
  - clearly stated goals and values that all staff share
  - common sense requites commonality (not in contrast with diversity)
  - efforts are targeted at meaningful goals
- Hiring process
  - diversify your inputs
  - share what you have learned
  - how is your marketing stacking up
    - look at your company web page
  - employee performance predictor
    - unstructured interview 14%
    - reference checks 7%
    - years of experience 3%
    - structured interview 26%
    - structured interview with sample test 29%

17.1.1.6. TODO GOTO 2019 • Mastering the Linux Command Line • Bert Jan Schrijver

- Hezký video. Za 45 minut jsem se dozvěděl skoro všechno co už jsem o příkazové rádce v Linuxu věděl, ale hezky udělané
- https://youtu.be/qmh7Uppd3x0

17.1.1.7. TODO YOW! Conference 2018 - Cat Swetel - The Metrics You Should Use But Probably Don't

Youtube	https://youtu.be/WEuHleDlpdc

- Nelson rules
- Weibull distribution
- Littles's law
- Leading indicators
- Lagging indicators
- #NOESTIMATES
- Unlabeled axes

What is the quality? Measurement of how much the quality you have
Quality is a characteristic of a relationship with customers

- Delivery date prediction
  - time in process
    - units of time per unit of work
    - cycle time … Maybe not a good name
    - weibull distribution not a standard distribution

        - is a better way to handle
    - automatically inform manager about the delay on other features that will get delayed
        - better to discuss the impact with customer
        - when it comes to non-profit random …. NELSON RULES!
    - multi domain model curve
- 4 aspects
    - quality
    - responsiveness
    - productivity
    - predictability
- Throughput
    - Units of work per unit of time
- Leading indicators Vs lagging indicators
- Littles's law
    - if your arrival rate is higher than departure rate everything in your system will take a more time.
- Don't use red and green as many man are colorblind
- Data … Thing just are … no judgment - productivity is lost by people being scared
- Ron Jeffries:
    - either I invented story points or I was there when it happened. Story points are thinly obfuscated time no more no less.
    - If I am the inventor of story points I am extremely sorry.
- If you see multi modal curve it shows that you have multiple types of workload

17.1.1.8. TODO DevOneConf 2018 - Siren Hofvander - Making Security an integrated part of the SW Dev Lifecycle

Youtube	https://youtu.be/8NzkYsY7TOk

- Each security talk she gives is full of sarcasm but useful an eye opening
- "I would never o that … This could never happened to me … really?"
- From 13th minute on she starts with practical suggestions.
- Názorně s nadhledem a sarkasmem o security.
- "To by se mi nikdy nestalo. To bych nikdy neudělal? "
- Hmm jo ale nejsme sami. Používáme cizí kód a možná*to* někdo udělal.
- GDPR - hoďme na to ručníkna budeme dělá že se to nikdy nestalo.
- Hmm a co to security experti co chodí v kravatě a c level managementu ukazují obrázek hackerů co mají kapucu a Anonymous masku na obličeji.
- Jasně a proces vše vyřeší. Hmm a taky koláčové diagramy to je terpve terno.
- A jasně že vývojáři milují kvanta formulářů.
- A od 13. minuty končí sarkasm a začínají dobré praktické rady.
- Na 38. Minutě je pár tipů a nástrojů

17.1.1.9. WIP Modern Java: Change is the Only Constant by Mark Reinhold

Youtube	https://youtu.be/SyVLioZmbOM

- Mluví o novinkách v Java 9 10 a 11.
- Nic objeveného ale kdo ho ještě neviděl mluvit, no proč né.

17.1.1.10. TODO Paying Technical Debt at Scale - Migrations @Stripe

- Něco i migracích a splácení technického dluhu - poučné …
- Od 15:00 minuty. Every migration is a product
- https://youtu.be/OFjvJmS_uDo
- YouTube | InfoQ

17.1.1.11. TODO Data Modeling, Normalization and Denormalization

- Hezká a klasika.
- Od 18:00 popisuje normální formy.
- Ok trochou starý ale furt na tom něco je.
- https://youtu.be/059NNng5BUQ
- http://www.postgresqltutorial.com/postgresql-identity-column/
- YouTube | FOSDEM

17.1.1.12. TODO CSAR: Prof. Elizabeth Stokoe

https://www.youtube.com/watch?v=-1TUF5N1b7E

17.1.1.13. TODO The Interactional 'Nudge' - Talking About Talk

https://www.youtube.com/watch?v=e-QbxjXDwXU

17.1.1.14. TODO Q&A - The Interactional Nudge - With Elizabeth Stokoe

https://www.youtube.com/watch?v=nrV0OWSJDHU

17.1.2. Vimeo [0/1]

17.1.2.1. WIP Meeting resistance and moving forward — Linda Rising

Authors	Linda Rising
Subject	Watch Meeting resistance and moving forward — Linda Rising
Vimeo	https://vimeo.com/325143661

Cognitive-bias

- Daniel Kahneman: Thinking, Fast and Slow
- Dan Ariely: predictably irrational
- how we have conversations with people we disagree
    - we show how much more we know
    - winning by killing the other people's does not work
    - thank you for showing me how stupid I am… Hardly helps
- we technical people believe that our smartness gives us an edge during discussion

        - our opponents are missing facts
        - we feel we must supply them with missing data
        - we can't understand why they are still resistant
        - the problem must be them
    - behavioral economics
        - we have to find a new way of
            - influencing
            - having discussion
            - rational arguments have its place but as a mean of influencing others is severely fault
        - read
            - thinking fast and slow
            - predictably irrational
                - easier to digest with nice experiments
        - we are not rational decision makers
        - preposition that people are rational is false
        - we are not convinced by
            - rational argument
            - facts
            - Data
                - scientific arguments
    - biases (heuristics that our brains use to protect itself)
        - confirmation bias
            - early 60th
            - Peter wason's research
            - when people have formed hypothesis they focus on evidence supporting pre-existing view
                - I believe you are right
                - believe you are wrong
                - I won't listen to you, you want listen to me… Imagine having a gun.
                - we think that we see the world clearly
                - we think that out explanation is the correct one
                - who disagree with us is just wrong
    - the backfire effect
        - Brendan Nyhon and Jason reifler
        - when you argue using facts
            - people reject/discount the information
            - cling even more to their original view
        - correcting people ,*increases* erroneous beliefs
    - cognitive dissonance
        - Leon Festinger
        - J. Merrill Carlsmith
        - 1959
        - we are extremely uncomfortable holding contradictory believes
            - contardictory "evidence" causes severe cognitive discomfort
    - cynics Vs skeptics
        - skeptics
            - are useful
            - every group should have one
            - every meeting should have one
            - every team should have one
            - Edvard de Bono : six thinking hats - someone who wears the black hat
        - cynics
            - are negative for the sake of being negative not RTO be helpful
                - typically focused on low level interests, not the greater good

17.2. WIP Podcasts [0/1] *[CHAPTER]*

17.2.1. WIP SE-Radio [0/1]

17.2.1.1. WIP SE-Radio 238: Linda Rising on the Agile Brain

Authors	Linda Rising
Seen	<2018-08-04 Sat>
Event Date	<2015-11-09 Mon>
Keywords	Soft%20Skills
Quality	worth returning to it
Source	SE-Radio
Subject	SE-Radio 238: Linda Rising on the Agile Brain
SE Radio	https://www.se-radio.net/2015/09/se-radio-episode-238-linda-rising-on-the-agile-brain/

My Remarks

- you should provide feedback based on a efford not on the result

Official Description

SE-Radio Episode 238: Linda Rising on the Agile Brain
11.09.15 by se-radio team
Web player: http://podplayer.net/#/?id=7160439 Episode: http://feedproxy.google.com/~r/se-radio/~5/5G8ZZrFHC5I/SE-Radio-Episode-238-Linda-Rising-on-the-Agile-Brain.mp3

- Johannes Thönes talks to Linda Rising, author, speaker and independent consultant, about the Agile Brain.
- They start by talking about the fixed, talent-oriented mindset and then contrast with the lg-oriented mindset.
- After establishing the terms, Linda explains how we know which mindset we are in currently and how we can change it for us and others,

17.2.2. Full Stack Radio

Url	http://fullstackradio.com
Rss	https://feeds.transistor.fm/full-stack-radio

- A podcast for developers interested in building great software products. Hosted by Adam Wathan.

17.2.2.1. 139: Alex DeBrie - DynamoDB for Relational Database Diehards

Full Stack Radio	https://fullstackradio.com/139

- Does DynamoDB only make sense for things like your cache, or is it a good choice for a primary data store?
- An overview of the terminology used in DynamoDB and how the terminology compares to a relational database
- How primary keys work in DynamoDB
- What data types are available in DynamoDB
- How DynamoDB is a schemaless database
- Why it's important to understand your access patterns in advance with DynamoDB, unlike in a relational database
- Understanding why and how you usually have multiple record types in a single DynamoDB table
- What "index overloading" is in DynamoDB
- Understanding partition keys and sort keys
- How to structure your data in DynamoDB to make it possible to query related data, and how those queries work
- How secondary indexes work, allowing you to access the same data in different ways
- How to accommodate access patterns you didn't know about before you designed your schema
- When to flatten relationships vs. nest them
- Should you use DynamoDB if you aren't "web-scale"?
- How local development works with DynamoDB

17.3. TODO Udemy [0/4]

17.3.1. TODO Docker Mastery: The Complete Toolset From a Docker Captain

- https://www.udemy.com/share/101WekBEEadVpUQXg=/
- Build, test, deploy containers with the best mega-course on Docker, Kubernetes, Compose, Swarm and Registry using DevOps

17.3.2. TODO Kubernetes Mastery: Hands-On Lessons From A Docker Captain

- https://www.udemy.com/share/102iLUBEEadVpUQXg=/
- Learn the latest Kubernetes features (1.16) and plugins while practicing DevOps workflows, from a container expert

17.3.3. TODO Docker Swarm Mastery: DevOps Style Cluster Orchestration

- https://www.udemy.com/share/101u3oBEEadVpUQXg=/
- Build, automate, and monitor a server cluster for containers using the latest open source on Linux and Windows

17.3.4. TODO Docker for Node.js Projects From a Docker Captain

- https://www.udemy.com/share/1022zOBEEadVpUQXg=/
- Build, test, deploy Node for Docker, Kubernetes, Swarm, and ARM with the latest DevOps practices from a container expert

17.4. TODO Blogs [/]

18. WIP Index [CHAPTER]

18.1. Index of Learning Links

Item	ref
Endiannes	https://en.wikipedia.org/wiki/Endianness
What is unicode	http://www.unicode.org/standard/WhatIsUnicode.html
iso-3166	https://en.wikipedia.org/wiki/ISO_3166
iso-639	https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes
Data Serialization Formats [0/5]	https://en.wikipedia.org/wiki/Comparison_of_data-serialization_formats
yaml	https://rollout.io/blog/yaml-tutorial-everything-you-need-get-started/
Amortized Analysis	https://en.wikipedia.org/wiki/Amortized_analysis
Finger Tree	https://en.wikipedia.org/wiki/Finger_tree
Trie	https://en.wikipedia.org/wiki/Trie
cache-oblivious algorithm	https://en.wikipedia.org/wiki/Cache-oblivious_algorithm
Redis	https://redis.io/topics/introduction
AMQP (Advanced Message Queuing Protocol)	https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol
MQTT (Message Queuing Telemetry Transport)	https://en.wikipedia.org/wiki/MQTT
EventSource	https://developer.mozilla.org/en-US/docs/Web/API/EventSource
Hasura 3Factor Apps	https://3factor.app/
OpenApiSpec / Open API 3.0 / Swagger	https://app.swaggerhub.com/help/tutorials/openapi-3-tutorial
µ-Services design patterns classes [0/6]	https://tsh.io/blog/design-patterns-in-microservices-api-gateway-bff-and-more/
µ-Services Design Patterns [0/8]	https://tsh.io/blog/design-patterns-in-microservices-api-gateway-bff-and-more/
CQRS	http://codebetter.com/gregyoung/2010/02/16/cqrs-task-based-uis-event-sourcing-agh/
GRASP	https://en.wikipedia.org/wiki/GRASP_%28object-oriented_design%29
The Law of Demeter (LoD) or principle of least knowledge	https://en.wikipedia.org/wiki/Law_of_Demeter
Evolutionary database design	https://martinfowler.com/articles/evodb.html
What does ACID mean?	https://en.wikipedia.org/wiki/ACID
What is object relation impedance mismatch	https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch

Item	ref
CAP theorem	https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e
Beyond CAP database systems	http://www.grokkingsystemdesigns.com/beyond-cap-theorem/
Paxos	https://en.wikipedia.org/wiki/Paxos_(computer_science)
MVCC Multiversion concurrency control	https://en.wikipedia.org/wiki/Multiversion_concurrency_control
PostgreSQL	https://www.highgo.ca/blog/
Big Query	https://cloud.google.com/files/BigQueryTechnicalWP.pdf
Atlas MongoDB	https://www.mongodb.com/cloud/atlas
Sparc	https://spark.apache.org/
Apache Oozie Workflow Scheduler for Hadoop	https://oozie.apache.org
Hadoop	https://en.wikipedia.org/wiki/Apache_Hadoop
hue	https://gethue.com/
Hive	https://hive.apache.org/
HBase	https://hbase.apache.org/
Apache Parquet	https://en.wikipedia.org/wiki/Apache_Parquet
Apache Avro	https://avro.apache.org
Arc42	https://arc42.org/
Open Policy Agent	https://www.openpolicyagent.org/docs/latest/
API Gateway	https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html
aim:service accounts	https://cloud.google.com/iam/docs/understanding-service-accounts
MinIO Object Storage	https://min.io/product/overview
Culture	https://blog.sonatype.com/principle-based-devops-frameworks-three-ways
The Three Ways	https://blog.sonatype.com/principle-based-devops-frameworks-three-ways
CALMS (Culture, Automation, Lean, Measurement, Sharing)	https://blog.sonatype.com/principle-based-devops-frameworks-calms
Special IPv4 Ranges	https://en.wikipedia.org/wiki/IPv4#Special-use_addresses
awk	https://www.digitalocean.com/community/tutorials/how-to-use-the-awk-language-to-manipulate-text-in-linux
DNS	https://www.freshersemploy.com/dns-interview-questions-answers/
Kibana	https://logz.io/blog/grafana-vs-kibana/
Graphana	https://logz.io/blog/grafana-vs-kibana/
K8S / Kubernetes: [0/27]	https://classpert.com/kubernetes
What are network policies?	https://github.com/ahmetb/kubernetes-network-policy-recipes
what are podSelectors?	https://medium.com/@zwhitchcox/matchlabels-labels-and-selectors-explained-in-detail-for-beginners-d421bdd05362
what is a RBAC	https://www.cncf.io/blog/2018/08/01/demystifying-rbac-in-kubernetes/
ChatOps	https://www.pagerduty.com/blog/what-is-chatops/
Rugged DevOps	https://ruggedsoftware.org/
Google Fonts API	https://developers.google.com/fonts
Cookies	https://tools.ietf.org/id/draft-ietf-httpbis-rfc6265bis-03.html
Cookies / SameSite	https://web.dev/samesite-cookies-explained/
CSS preprocessors: LESS Sass	https://www.lambdatest.com/blog/css-preprocessors-sass-vs-less-vs-stylus-with-examples/
Sass	https://sass-lang.com/guide
CORS [0/5]	https://www.codecademy.com/articles/what-is-cors
Content Security Policy (CSP)	https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l
Bundlers	https://bundlers.tooling.report/
Webpack	https://webpack.js.org/concepts/
ECB, CBC, OFB, CFB, CTR - what they are	https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm//
HMAC	https://en.wikipedia.org/wiki/HMAC
what is spiffe	https://spiffe.io/docs/latest/spiffe/overview/
Resource Owner Password Credential Grant	https://www.netiq.com/documentation/access-manager-45-developer-documentation/pdfdoc/oauth-application-developer-guide/oauth-application-developer-guide.pdf
What is acr_values	https://ldapwiki.com/wiki/Acr_values
JWT	https://tools.ietf.org/html/rfc7519
RS256 vs HS256: What's the difference?	https://stackoverflow.com/questions/39239051/rs256-vs-hs256-whats-the-difference
JWKS	https://auth0.com/docs/tokens/json-web-tokens/json-web-key-sets
OWASP top 10 2017	https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/
CSRF / XSRF	https://owasp.org/www-community/attacks/csrf
DevSecOps	https://snyk.io/devsecops/
Software Composition Analysis (SCA) [0/2]	https://sudonull.com/post/27911-Forrester-Research-A-Comparison-of-Ten-Top-Software-Composition-Analysis-Vendors
Fix	https://www.parsonsmatt.org/2016/10/26/grokking_fix.html

Item	ref
How to create thread safe Singleton in Java	https://javarevisited.blogspot.com/2012/12/how-to-create-thread-safe-singleton-in-java-example.html
The Mostly Adequate Guide to Functional Programming	https://github.com/MostlyAdequate/mostly-adequate-guide
Scoped Packages	https://docs.npmjs.com/about-scopes
Babel	https://babeljs.io/
Spring & Spring Boot	https://www.baeldung.com/spring-interview-questions
Jest	https://jestjs.io/
Enzyme	https://enzymejs.github.io/enzyme/
Istambul	https://istanbul.js.org/
StrangeLoop	https://www.thestrangeloop.com/
The Backfire Effect	https://youarenotsosmart.com/2011/06/10/the-backfire-effect/
ACM ethics Code	https://ethics.acm.org/
Back-pressure Strategies	https://medium.com/@jayphelps/backpressure-explained-the-flow-of-data-through-software-2350b3e77ce7
OpenID Connect Core 1.0 incorporating errata set 1	https://openid.net/specs/openid-connect-core-1_0.html#IDToken

18.2. Index of YouTube Videos

Item	ref
Webinar: Internal Building Blocks: Aggregates, Entities, Value Objects Part 3/5	https://youtu.be/RHg53wMflCc
GOTO 2018 • Old Is the New New • Kevlin Henney	https://youtu.be/AbgsfeGvg3E
Kevlin Henney - 1968	https://youtu.be/v-RkKohsF78
Developing Microservices with Aggregates by Chris Richardson	https://youtu.be/gR_EGN86fvg
Re Imagine The Hiring Process	https://youtu.be/TbZ57raC1TE
YOW! Conference 2018 - Cat Swetel - The Metrics You Should Use But Probably Don't	https://youtu.be/WEuHleDlpdc
DevOneConf 2018 - Siren Hofvander - Making Security an integrated part of the SW Dev Lifecycle	https://youtu.be/8NzkYsY7TOk
Modern Java: Change is the Only Constant by Mark Reinhold	https://youtu.be/SyVLioZmbOM
Refactoring Restify OAuth2	https://youtu.be/aHrwUnsPom0
Avoiding Microservice Megadisasters - Jimmy Bogard	https://youtu.be/gfh-VCTwMw8
Kevin Johnson & Jason Gillam - Next Gen Web Pen Testing - AppSecUSA 2016	https://youtu.be/rVnXzW31pWQ
What Doctors Can Teach Us on Continuous Learning - Johnny Graber	https://youtu.be/SuZZV74wTkw
Exploiting Network Printers	https://youtu.be/DwKzSO4yA_s
Attacks Against GSMA's M2M Remote Provisioning	https://youtu.be/rI9hvDDbdlE
Web Application Security Risks: A Look at OWASP Top Ten 2017 - Christian Wenz	https://youtu.be/avFR_Af0KGk
Od Aristotela k Newtonovi	https://youtu.be/w9ulyaauBIE

18.3. Index of SE-Radio Podcast Episodes

Item	ref
SE-Radio 238: Linda Rising on the Agile Brain	https://www.se-radio.net/2015/09/se-radio-episode-238-linda-rising-on-the-agile-brain/

18.4. Index of Concepts

Item	ref
Bottom Up	Not Yet Defined
Top Down	Not Yet Defined
Meet in the Middle	Not Yet Defined
Distributed consensus:	Not Yet Defined
Transactions	Not Yet Defined
MOSCOW	Not Yet Defined
ChatOps	https://www.pagerduty.com/blog/what-is-chatops/
Rugged DevOps	https://ruggedsoftware.org/
Tail Latency	Not Yet Defined
SLI	Not Yet Defined
SLO	Not Yet Defined
SLA	Not Yet Defined
DevSecOps	https://snyk.io/devsecops/
Zero Trust Network	Not Yet Defined
Regression to the Mean	Not Yet Defined
Fat Tail Distribution	Not Yet Defined
Data Driven Decision	Not Yet Defined
Shift Left	Not Yet Defined
Immutability	Not Yet Defined
I-Shaped vs T-shaped / Π-shaped	Not Yet Defined

Item	ref
Chaos Engineering	Not Yet Defined
Intrinsic vs Extrinsic Motivation	Not Yet Defined
Leading vs Lagging Indicators	Not Yet Defined
Declarative vs Imperative	Not Yet Defined
Strict vs Lazy Evaluation	Not Yet Defined
Back-pressure Strategies	https://medium.com/@jayphelps/backpressure-explained-the-flow-of-data-through-software-2350b3e77ce7
Overqualified Query	Not Yet Defined
Access Intent	Not Yet Defined
Skepticism	Not Yet Defined

18.5. Index of Books

Item	ref
Gene Kim, Patrick Debois, John Willis, Jez Humble: The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations	Not Yet Defined
Daniel Kahneman: Thinking, Fast and Slow	Not Yet Defined
Nassim Nicholas Taleb: The Black Swan, Second Edition: The Impact of the Highly Improbable:	Not Yet Defined
Nassim Nicholas Taleb: Fooled by Randomness: The Hidden Role of Chance in Life and in the Markets	Not Yet Defined
Nassim Nicholas Taleb: Antifragile: Things That Gain from Disorder	Not Yet Defined
Nassim Nicholas Taleb: Skin in the Game: Hidden Asymmetries in Daily Life	Not Yet Defined
Barry O'Reilly: Unlearn: Let Go of Past Success to Achieve Extraordinary Results	Not Yet Defined
Stuart Firestein: Failure: Why Science Is so Successful	Not Yet Defined
Nate Silver: The Signal and the Noise: Why So Many Predictions Fail - but Some Don't	Not Yet Defined
Philip Tetlock , Dan Gardner: Superforecasting: The Art and Science of Prediction	Not Yet Defined
Daniel Coyle: The Culture Code: The Secrets of Highly Successful Groups	Not Yet Defined
Andy Greenberg: Sandworm: A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers	Not Yet Defined
Philip E. Tetlock: Expert Political Judgment: How Good is it? How can We Know?	Not Yet Defined
Steven Pinker: The Stuff of Thought: Language as a Window into Human Nature	Not Yet Defined
Simon Conway Morris (Editor): The Deep Structure of Biology: Is Convergence Sufficiently Ubiquitous to Give a Directional Signal	Not Yet Defined
Robert M. Sapolsky: Behave: The Biology of Humans at Our Best and Worst	Not Yet Defined
Irvin D. Yalom: The Schopenhauer Cure: A Novel	Not Yet Defined
Michael A Britt: Psych Experiments: From Pavlov's dogs to Rorschach's inkblots, put psychology's most fascinating studies to the test	Not Yet Defined

18.6. Index of Patterns

Item	ref
File Transfer	Not Yet Defined
Shared Database	Not Yet Defined
Remote Procedure Invocation	Not Yet Defined
Messaging	Not Yet Defined
Message Channel	Not Yet Defined
Message	Not Yet Defined
Pipes and Filtres	Not Yet Defined
Message Router	Not Yet Defined
Message Translator	Not Yet Defined
Message Endpoint	Not Yet Defined
API Gateway	Not Yet Defined
Circuit Breaker	Not Yet Defined
CQRS	http://codebetter.com/gregyoung/2010/02/16/cqrs-task-based-uis-event-sourcing-agh/
Event Sourcing	Not Yet Defined
Sidecar	Not Yet Defined
Backend-for-Frontend	Not Yet Defined
Strangler Pattern	Not Yet Defined
Saga	Not Yet Defined
Event Sourcing	Not Yet Defined

Item	ref
CQRS	Not Yet Defined
CRUD	Not Yet Defined

18.7. Index of AWS Related Topics

Item	ref
EC2 (Amazon Elastic Compute Cloud)	Not Yet Defined
AMI	Not Yet Defined
ELB CLB ALB	Not Yet Defined
Route 53	Not Yet Defined
Beanstalk	Not Yet Defined
RDS	Not Yet Defined
API Gateway	https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html
AWS Lambda	Not Yet Defined
DynamoDB	Not Yet Defined
Kinesis	Not Yet Defined
EFS	Not Yet Defined
SNS	Not Yet Defined
S3	Not Yet Defined
EBS	Not Yet Defined
CloudWatch	Not Yet Defined
CloudFormation	Not Yet Defined
CloudFront	Not Yet Defined
Secrets Manager	Not Yet Defined
VPC	Not Yet Defined
Wavelength	Not Yet Defined
boto3	Not Yet Defined
aws-shell	Not Yet Defined
SAM	Not Yet Defined
AWS S3	Not Yet Defined

18.8. Index of GCP Related Topics

Item	ref
Big Table	Not Yet Defined
Big Query	https://cloud.google.com/files/BigQueryTechnicalWP.pdf
GCS bucket	Not Yet Defined