**Information Architecture:**

* Tackles 2 problems:
  + Information overload – growing volume of information available to us results in a problem managing attention
  + Multiplication of Access Channel – proliferation of applications and electronic devices multiplied the number of channels through which we can access the same information
* “Places made of information” – Information products and services where people go for different tasks:
  + User experience is defined by familiar vocabulary (labels, menus, descriptions, …)
  + Different use of language makes them distinct which can help users be more efficient in accomplishing their goals
  + Information Architecture considers these info-rich spaces and their design to maximize effectiveness, specifically:
    - Connect information objects to users.
    - Identify concepts and pathways for access and navigation.
    - Create tools and systems for people to organize info.
    - Connect various information spaces, apps, platforms, and channels.

This requires skills from multiple disciplines (usability, design, information, …) -> Umbrella term User Experience (UX).

* Related Disciplines:
  + Usability Engineering – human computer interaction and how interfaces help users accomplish their tasks
  + Information Science – create organization and retrieval of information
  + Human Factors Engineering – ergonomics and physical factors in designing
  + Visual Design – aesthetics and communication of information
  + Interaction Design – how a system works in response to inputs

**Elements of Information Architecture:**

* Main Components:
  + Organization systems (How info is categorized) – Defines the shared characteristics of items and their grouping
    - The way things are organized impact their meaning – Well organized info is easier for people to find and work with.
    - Organizing things is hard because it requires us to deal with ambiguity, heterogeneity, different perspectives, politics, …
    - Can be of two types:
      * Exact Organization Schemes – Elements are divided into well-defined and mutually exclusive sections (Easy to design and maintain)
        + Alphabetical scheme (libraries, person directory, …)
        + Chronological scheme (calendars, task lists, …)
        + Geographical scheme (transportation, rental listings, …)
      * Ambiguous Organization Schemes – Important because people don’t always have an exact definition of what they are looking for and language and concepts are ambiguous in nature (Harder to design and maintain)
        + Categorical schemes (books, academic courses, newspapers, …)
        + Hierarchical schemes – organized by value from list to most or vice versa – Requires value to be assign to information (search engines, social networks, …)
        + Audience-specific schemes – when multiple audiences exist breaks info space in smaller audience-specific sub-spaces including more or less info depending on it (intranets, academical services, …)
  + Labelling systems (How info is represented)
    - Labels represent parts of info in info spaces
      * Textual (contextual links, headings, nav systems, …)
      * Iconic
    - Labels can be designed looking at
      * existing info env (what is used in other contexts or by others in similar contexts)
      * search logs (what language users use for searching)
  + Navigation systems (How you can move through the info) – Helps users move around
    - Provide context and a sense of control to the users as they explore
    - Should answers:
      * Where am I?
      * What can I do?
      * Where can I go from here?
    - Types of nav systems:
      * Global navigation systems:
        + Present in every page (often at the top)
        + Allows direct access to key areas of the system
        + Central element in the overall usability space
      * Local navigation systems:
        + Complement the global nav system
        + Allow users to explore and navigate the immediate subsection
      * Contextual navigation systems:
        + Nav through association of concepts that exist in the content being presented (context clues or habitual behaviour)
      * Supplemental navigation systems:
        + External to the basic hierarchy of a system
        + Complementary way of finding content and completing tasks

Sitemaps – broad view of the content in the system and facilitates random access for individual items

Breadcrumbs – dynamic and represent the path followed by the user or the location of the content within the hierarchy

Search – central mechanism for navigation, provides users direct access to the content they are looking for

* + - Advanced navigation Solutions:
      * Personalization – serve info based on user behaviour, navigation, profile, … (Recommendation panels, …)
      * Customization – give user control over content and nav options (ordering or labels in an email)
      * Visualization – provide nav mechanisms based on visual properties of info items (a visual search engine in a e-comerce website)
      * Social navigation – organize content access or navigation based on user input (upvotes in reddit, user tags in stack overflow)
  + Searching systems (How info can be searched for)
    - Advanced searching mechanism
    - Simple interface with a complex system
    - Requires evaluation and considering multiple issues:
      * Amount of content in the info space
      * Time and know-how to optimize the search system
    - Expected by users – users control vocab to find info and cut across the existing system
    - Multiple decisions:
      * Which content fields to index
      * What text processing to preform
      * Which info components to include in the search results
    - Complementary sorting options can be provided (relevance, alphabetical, …)
    - Faceted search – advances mechanism commonly used in stores and complex info contexts
    - Best practices:
      * Support autocomplete
      * Support query operators
      * Provide context
      * Allow querry reformulation
      * Support additional filters
      * Support alternative rankings
      * Customize snippets
      * Highlight matched search items

**Information Architecture Process:**

* Requires interdisciplinary teams (designers, software developers, content strategists, …)
* Has 4 general activities:
  + Research – understands user, content, and context.
  + Design – specify the info architecture (creating sitemaps, wireframes, …).
  + Implement – solutions that adhere to the design and specifications produced
  + Evaluate – improve the system throughout its life cycle

**Information Architecture Deliverables:**

* work as anchors between teams and different project phases
* visual diagrams contribute to define:
  + content components (what constitutes a unit of content and how these are grouped)
  + connections between components (how components are linked)
* can provide multiple views at different levels for different audiences
* Types:
  + Sitemaps – show the relationship between information elements such as pages and can be used to portray organization, navigation, and labelling systems
    - provides a condensed view for both developers and users
    - high-level sitemaps are usually a result of a top-down design process
    - can be uses to map specific areas or parts of a complex info environment
  + Wireframes – depict how an individual page or template should look from a structural and architectural perspective.
    - Need to consider issues such as location of nav systems, content hierarchy, …
    - Typically created for the system’s most important pages or screens
    - Good way to explore the impact of different screen sizes on screen layout
    - Level of fidelity can vary depending on the stage of the development lifecycle
  + Wireflows – combine wireframes and flowcharts to provide both a view of page-level layout ideas and document complex workflows and user tasks
    - Useful when documenting systems with few pages that dynamically change content and layout based on user interaction
    - Comon in mobile applications

**Database Indexes, Triggers and Transactions:**

* Mapping generalizations:
  + Superclass approach: media (id, type CHK {book,cd,dvd},…)
  + ER approach:
    - media(id,…)
    - book(id->media, …)
    - cd(id->media, …)
    - dvd(id->media, …)
  + Object oriented:
    - book(id, …)
    - cd(id, …)
    - dvd(id, …)
* Indexes:
  + Workload – study of the predicated system load including an estimate on the number and growth of tuples in each relation
  + Performance Indexes are applied to improve performance of select queries, but they add an overhead to execution
  + Two main types:
    - B-tree indexes – use a tree-like data structure that maintains data sorted and allow for search, order, range search in log time
    - Hash indexes – use a hash-function to map keys to values (only considered when there’s an equality operator, no sorts or ranges)
  + Can be created for more than one attribute
  + When an index is declared unique, multiple table rows with equal indexed values are not allowed (Null values are not considered equal)
  + PostgreSQL automatically creates indexes when:
    - A unique constraint
    - Primary key is defined for table
* Clustering:
  + Physical re-ordering of data in disk based on the index info (must be defined already).
  + One time operation – when the table is updated changes are not clustered
  + Clustering can be set to run periodically using cron
  + Good when multiple records are read together, and an index can group them – irrelevant when single rows are randomly accessed in a table
* Cardinality:
  + Uniqueness of data values contained in a particular column
  + Lower cardinality -> more duplicate values in a column
  + Used by the PostgreSQL planner to estimate the number of rows returned by a where clause which helps decide if and what indexes should be used
* Full Text Search
  + Text is broken into lexemes (normalized representation of words) stored in the data type tsvector
  + to\_tsvectors returns a tsvector with duplicates and stop words removed and the number of positions of each lexeme recorded
  + Weight – Sometimes we want to give more importance to some specific fields
* Triggers and User Defined Functions
  + Enforce integrity rules that cannot be achieved in a simpler way
  + Advantages of using stored procedures:
    - Reduce the number of round trips between the application and database server
    - Increase the application performance because user-defined functions are pre-compiled and stored in the database server
    - Be able to re-use in multiple applications
  + Disadvantages of using stored procedures:
    - Slow software dev because it requires skills that most software devs do not have
    - Make it difficult to manage version and hard to debug
    - Less portable code to other db management systems
* Transactions:
  + Bundle multiple steps into a single all-or-nothing operation ensuring data integrity with concurrent accesses
  + Can be done in different isolation levels (achieved mostly by locking access to tables)
  + We should aim to the less restrictive isolation level that still guaranties data consistency
  + Isolation levels are defined in terms of the problems (phenomena that can occur when concurrent transactions execute):
    - Dirty-read – a transaction read data written by a concurrent uncommitted transaction
      * T1 writes A and T2 reads A right after, before T1 has committed
    - Non-repeatable read – a transaction re-executes a query and finds data has been modified by another transaction
      * T1 reads A as 10, T2 modifies A to 20, T1 reads item again but now it’s 20 and not 10
    - Phantom read – a transaction re-executes a query and finds that the results have been changed by another transaction
      * T1 preforms Q1(A) and finds 2 rows available, T2 preforms Q2(C) (same condition) and inserts a new row, now if T1 repeats the query it’ll find 3 rows instead (phantom row)
    - Serialization anomaly – the results of committing a group of transactions is inconsistent with all possible orderings of running those transactions at a time

|  |  |  |  |
| --- | --- | --- | --- |
| **Isolation Level** | **Dirty Read** | **Non-Repeatable Read** | **Phantom** |
| **Read Uncommitted** | x | x | x |
| **Read Committed** |  | x | x |
| **Repeatable Read** |  |  | x |
| **Serializable** |  |  |  |

**Internet**

* Network of connected computers
* One of the first challenges faced was how to connect separate physical networks without dedicated links (point to point connections do not scale)
* Packet switching is a method where data is divided in small chunks and sent out separately – makes it possible to have multiple connections over the same link
* Internetworking:
  + Proliferation of different networking technologies and protocols became a problem when trying to connect different networks (each network was becoming an island, isolated from all others)
  + To overcome this problem and achieve a homogeneous service across heterogeneous networks both hardware and software were combined:
    - Hardware: routers
      * Core equipment used to connect networks using different physical technologies.
      * An internet is a set of networks connected by routers
    - Software: protocols (TCP/IP)
      * Establish message formats and message exchanging rules
      * Internet Protocol (IP):
        + Offers a virtual network hiding the underlying physical networks
        + Offers two services:

Addressing system (IP addresses)

Datagram structure (packets)

* + - * Transmission Control Protocol (TCP):
        + Offers a reliable and ordered delivery of packets between applications in different computers.
        + Handles problems not addressed in the lower layers (packet duplication and loss, packer ordering, communication delays, …)
      * Internet Services:
        + Domain Name System (DNS) – translates human-readable symbolic names to numeric addresses (IP)

Symbolic names are organized hierarchically

Right most element is the top-level domain (TLD) – there are different types of TLDs (countries and generic)

* + - * WHOIS Protocol – query/response protocol used to query databases that contain info about internet resources, such as domain names or IP addresses
        + Presents info in a human-readable format
        + Standard for querying domain name info

**The World Wide Web**

* Distributed info system over the internet designed to facilitate content sharing across different computer systems and technologies
* Architecture follows a standard client-server model where servers provide a function and clients initiate requests for those services
  + A Web Server is a program whose primary function is to deliver resources on clients’ requests – Only acts when requests arrive
    - Handles multiple clients at a time
  + Web Clients or Web Browsers are software applications capable of retrieving, presenting, and traversing information resources available on the World Wide Web
    - Initiate the communication session with servers
    - Communicate with web servers using the HTTP protocol
    - Issue multiple requests when preparing a single web page, one request per resource
    - Increasingly sophisticated
    - Structure:
      * User Interface (browser controls)
      * Browser Engine (mapping)
      * Rendering engine (HTML & CSS)
      * Networking (network calls)
      * JS Interpreter (execute javascript)
      * UI Backend (drawing widgets)
      * Data Persistence (saves data)
* Core technologies:
  + Uniform Resource Locator (URL)
    - Establishes a unique address for a WWW resource
    - Syntax:
      * Protocol, e.g. http://, ftp://, file://
      * Machine, e.g. [www.up.pt](http://www.up.pt) or 193.137.55.13
      * Port, e.g. 80 (default), 1000, 20
      * Resource path, directory path to the file
  + HyperText Transfer Protocol (HTTP)
    - Defines how web clients communicate with web servers to access web resources
    - Request-response – client issues a request and waits for the server to respond (Timeouts can occur)
    - Stateless – each request is treated as an independent transaction
      * Simpler design but requires additional information in each request
      * State must be maintained by web applications
        + Cookies – client-side pieces of data generated by the server and attached to the HTTP request
        + Sessions – server-side files with unique identifiers (session IDs), which can be passed in URLs or Cookies
    - Request Methods:
      * GET – Request resource
      * HEAD – Request only headers
      * POST – submits data to be processed to the identified source
      * PUT - Uploads data into the specified resource
      * DELETE – Deletes the specified resource
    - Status Codes – indicate if the request succeeded or if another action is required:
      * 1xx – Informational
      * 2xx – Success
      * 3xx – Redirection
      * 4xx – Client Error
      * 5xx – Server Error
  + HyperText Markup Language (HTML)
    - Defines the content and structure of hypertext documents

**Web Applications:**

* Software system based on web standards and technologies that is accessible through a web browser
  + Pros:
    - Platform independence
    - Easier updates & bug fixes
    - Only one version of the app
    - Access from anywhere
    - Reduced “piracy”
    - No installation hurdles
    - Devs can measure user interaction in real time
  + Cons:
    - Depends on network connectivity
    - Less sophisticated UIs
    - Limited hardware access
    - Reduced OS integration
    - Need to address browser versions
    - Harder to debug
    - Higher security risks
    - Infrastructure cost
  + Static Web Pages – In the early days of web pages were static files served directly from the filesystem (Pages are constructed at design time)
    - Fast
    - Rigid (no updates, no personalization and hard to maintain in scale.
  + Dynamic Web Pages – Instead of serving static files from the filesystem, software apps produce web pages when requested (Pages are constructed at run time)
    - The Common Gateway Interface (CGI) is a specification that defines how web requests and responses interact with an app
    - Sites emerged as a collection of multiple coherent web pages (shared functions and libraries)
    - Allows for a stateful experience to the user (e. g. shopping cart)
    - Common data layer implemented across pages
    - Libraries and frameworks to address repetitive and common tasks
    - Richer user interfaces (JavaScript, …)
    - Rendering can be:
      * Server-side rendering (SSR):
        + The document is prepared on the server and the result is sent to the client
        + Code execution is on the server
        + Complex documents with live information
        + Requires full round trips to the server for user interaction
      * Client-side rendering (CSR):
        + The server sends to the client the necessary elements to build the document (JS code in particular)
        + The client builds the document using JavaScript
        + Complex interactive document
        + Heavy on the client side
        + Not really hypertext
      * A mix of these techniques can be used (AJAX)
        + No need to reload full pages on each user interaction - Requests are made to the server from the document using JavaScript without loading a new document.
        + The server sends a response in the format of an HTML document, JSON, …
        + The HTML document is dynamically altered depending on the answer of the request.
        + N-to-1 mapping between app views and web pages
* Changes significantly over time due to technological advancements:
  + asynchronous interactions (AJAX)
  + New HTML5 standards
  + Broad adoption of mobile devices
* Three-Tier Architecture
  + Structured in three tiers:
    - Presentation
    - Business Logic
    - Data Management
  + Advantages:
    - Separation of concerns
    - Maintainability
  + Business Logic used to only be in the Web Servers but now it can be also found in the client.
* Multi-Page Web Applications
  + Implemented as a collection of multiple web pages
  + User interacts with the app nav through these pages
  + Each page is prepared by the server and only presentation details are sent to the browser
  + Advantages
    - REST style
    - Client independent
    - Consistency across browsers
    - Broad technological ecosystem
    - App logic is kept in the system
  + Disadvantages
    - Slow performance and responsiveness
    - Fragmented code
    - No way to deliver updates to an open web page
* Single-Page Web Applications
  + Implemented as a single web page
  + All necessary resources are loaded or dynamically added to the page (e.g. using Ajax).
  + App logic is pushed to the client (fat client architecture)
  + Advantages:
    - Improved user experience
    - Reduced bandwidth consumption
    - Decoupled client and server
    - Reusable server interfaces
    - Reusable client code
  + Disadvantages:
    - JavaScrip required
    - Breaks browser history
    - Increased browser dependency (version, features, performance, …)
    - No REST
    - Difficult to crawl and index
* Mixed Approach:
  + The two architectural styles can be mixed to combine the benefits of both
  + Use different web pages to setup the core structure of the app
  + Define data APIs to provide updates to the web pages and support different devices
  + Use asynchronous requests to improve performance and user experience
* Criteria to consider in choosing an architecture:
  + Usability – User friendly? Instant updates possible? Browser History?
  + Search / Share – Search engine friendly?
  + Linking – Mapping between URLs and views? 1-to-1?
  + Performance – Consistent across platforms? Optimized content loading? Client effort?
  + Productivity – Developer’s background? Modularity?
  + Testing – Modularity? What to test? Client dependency?

**Web Development Frameworks:**

* Generic software foundation over which custom app-specific code can be written
* Include multiple libraries (used by the app-specific code to support specific features) in addition to tools and rules on how to structure and use these components.
* Control the application flow and call app-specific code.
* Advantages:
  + Implementation speed
  + Tested proven solutions
  + Access to expertise and off-the-shelf solutions
  + Maintenance (updates, patches, …)
* Disadvantages:
  + Reduced independence
  + Lower performance
  + Dependence on external entities
  + Technological lock-in
* What to consider:
  + Team expertise on language, libraries, and framework
  + Existing code base
  + Licensing model
  + Maturity
  + Community support
  + …
* Types:
  + Micro Frameworks – focused on routing HTTP request to call back, commonly used to implement HTTP APIs.
  + Full-Stack Frameworks – feature-full frameworks that includes routing, templating, data access and mapping, plus many more packages
  + Component Frameworks – collections of specialized and single-purpose libraries that can be used together.
* Core components:
  + Request Routing – match incoming HTTP request to code.
    - Handled independently from app code using request routing
    - Clean URLs are an important part of a web app
  + Template Engine – structure and separate presentation from logic.
  + Data Access – uniform data access, mapping, and configuration.
* Common components:
  + Security - protection against common web security attacks
  + Sessions – session management and configuration
  + Error Handling – capture and manage app-level errors
  + Scaffolding – quickly generate CRUD interfaces based on data model

**Web Accessibility:**

* Web sites should be designed to ensure everyone including users with disabilities can use them.
  + Access to information and communication technologies is a human right
  + Promote equal access and equal opportunity to people with diverse abilities
  + Supports social inclusion for people with disabilities as well as others, such as older people, people in rural areas or people in developing countries
  + Mandatory by law for public services in many countries
* Common Disabilities:
  + Vision - blindness, low vision, colour-blindness
  + Hearing – deafness
  + Motor – inability to use a mouse, limited fine motor control skills
  + Cognitive – learning disabilities, inability to remember information
* Assistive Technologies:
  + Screen readers
  + Screen magnification
  + Tet enlargers
  + Alternative input devices (voice recognition, eye tracking)
* Accessibility Guidelines:
  + Design forms for users using assistive technologies
  + Do not use colour alone to convey information
  + Enable users to skip repetitive nav links
  + Provide text equivalents for non-text elements
  + Do not require stylesheets for content readability
  + Ensure that scripts allow accessibility

**Web Usability:**

* User interfaces should be effective
* Factors that effect the user’s experience with the application:
  + Ease of Learning
  + Efficiency of Use
  + Memorability
  + Error Frequency and Severity
  + Subjective Satisfaction
* Challenges in Web Usability
  + Wide range of users both in experience and in expectations
  + Wide range of platforms and technologies
  + User’s experience is formed on other web sites
  + Web users have a very low tolerance for poor usability
  + Other products are only a click away
* Popular Interaction Patterns:
  + Lazy registration - let the user take actions before forcing them to register
  + Progressive disclosure – show most relevant features or information to the user and delay further details or complex features until requested
  + Breadcrumbs – clearly show the path from the front page to the current location
  + Account registration – require only necessary information
  + Required field marker – make obvious the required fields in any form
* Discount Usability:
  + Simplify User Testing
  + Narrowed-down prototypes
    - Paper prototyping – one the fastest and cheapest techniques that can be employed in a user-centered design process - It’s cheaper and easier to make changes before any code has been written
    - A/B testing – randomized controlled experiments to evaluate the effects of two different versions (A and B) of a given web site, page or interaction.
      * Pros:
        + Measures actual user behaviour
        + Can measure very small performance differences
        + Results in objective measures
        + Cheap
      * Cons:
        + Requires a clear well-defined goal
        + Not everything is measured
        + Requires fully implemented designs
  + Heuristic Evaluation

**Information retrieval:**

* Deals with representation, storage, organization of and access to information items
* Includes:
  + Document and query modelling
  + Web search
  + Text classification
  + System architecture
  + User interfaces
  + Data visualization
  + Filtering
* Motivation – RDBS provide set-based or data retrieval
  + Limitations:
    - There is no linguistic support
    - Difficult to search for multiple keywords
    - Degraded performance when dealing with many documents
    - No ranking of results (e.g., order by relevance)
  + The key goal of an IR system is to retrieve all items that are relevant to a user query representing an information needed, while retrieving as few non relevant items as possible – Relevance
* Search Engine Architecture:
  + Indexing process – offline, when collecting changes
    - Test Acquisition – identifies and stores documents for indexing
    - Text transformation – transforms documents into index terms or features
    - Index Creation – takes terms and creates data structures to support fast searching
  + Querying process – online, in response to user queries
    - User interaction – supports creation and refinement of queries, display of results
    - Ranking – use query and index to generate ranked list of results
      * Estimating each document’s relevance for a given user query and context is done using various sources of information, usually called signals.
      * Web search engines use hundreds of signals, also called features with two types:
        + Static signals – can be computed during the indexing process

Origin of the document

References to the document

Age of the document

…

* + - * + Query-dependent signals – only available at query time

Number of query terms

Time of day

Query terms in document

…

* + - * Signal can also be divided according to their source (document-based, collection-based, user-based, …)
    - Evaluation – monitors and measures effectiveness and efficiency
* Term-Document Matrix – basic concept that represents the relation between indexed terms and collection documents
* Term Weighting – Terms are not equally useful for describing a document
  + Quantify the importance of a given index term for describing the content of a document
  + Term Frequency can be used as an estimation of the term importance for a given document however it can be easily manipulated
  + Inverse Document Frequency – important but less intuitive measure
    - Terms that appear in fewer documents of a collection have more discriminative power, thus are given a higher weight (specificity of a term)
  + TF-IDF – combination of TF and DTF (tf-idf(term,document,collection) = tf(term, document) x idf(term,collection))
  + Vector Space Model – binary weights are too limiting, proposes a framework in which partial matching is possible.
    - Document and queries are represented as unary vectors in a n-dimendional space – the similarity between 2 different docs is obtained using the cosine between the two vectors
* Link-based signals
  + The web is a directed graph
  + The hyperlinks pointing to a given page are a measure of the quality of a page
  + PageRank
    - Computed iteratively
    - All nodes start with the same initial value
    - The score of each none is distributed to the documents that it links to until the score of each node converges
* Retrieval Efficiency
  + The goal is to process user queries with minimal requirements of computational resources
  + The inverted index is a word-based data structure built to speed up access
    - The vocabulary is the set of all different words
    - For each word the index stores the document which contains that word
  + Full inverted index – the basic index is not suitable for answering phrase or proximity queries hence we need to add the position of each word in each document to the index
* Retrieval Evaluation
  + Determine how well a system is performing and evaluate changes
  + Compare the performance of a system to others
  + Challenging since it can’t be measured by objective measures
  + Precision and Recall
    - Consider:
      * R, set of relevant documents in the collection
      * A, set of documents in the retrieved answer
    - We can define the two core measures in IR evaluation,
      * Precision is the fraction of the retrieved documents that are relevant
      * Recall is the fraction of the relevant documents that are retrieved