BACKEND-WORKSHOP

Michael Fröhlich - michael-froehlich@cdtm.de

Tobias Dümmling - tobias.duemmling@cdtm.de

YOUR EXPECTATIONS?

WHY ARE WE DOING THE WORKSHOP?

- Almost every application has a back end
- Backend development is a black box
- Helping to understand key concepts of backend development
- Having a shared set of vocabulary

WHAT ARE WE DOING THE WORKSHOP?

- Develop a backend for a task application
- We will learn about:
  - Webservers
  - APIs
  - Databases
  - ...

AGENDA

FRIDAY 17:00–21:00

1. Introduction to Backend Development
2. Introduction to Development Tools
3. Python 101 (Quick and Dirty)
4. Hello World Server

AGENDA

SATURDAY 09:00–17:00

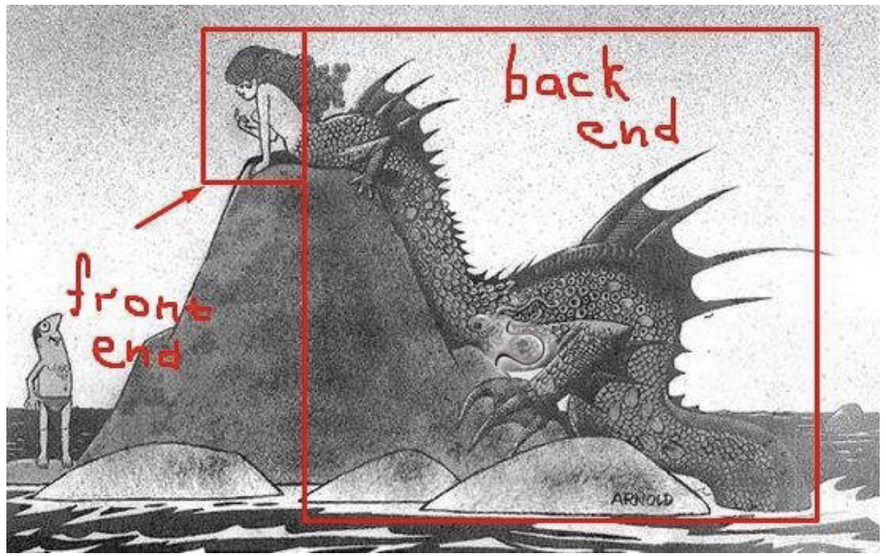
1. Object Modelling: What are our core objects?
2. APIs: HTTP, REST & JSON
3. Data Storage: Database and Basics of SQL
4. Data Storage: Handling big files

AGENDA

SUNDAY17:00–21:00

1. Refactoring: How to deal with growing projects?
2. Data Storage: Advanced SQL
3. Authentication: Sessions & Cookies

TWO PARTS:
FRONT-END & BACK-END



- What is the difference between front-end and back-end?

Back-End

The back-end, or the "server-side", is basically how the site works, updates and changes. This refers to everything the user can't see in the browser, like databases and servers.

In software engineering, **front end** (*frontend*) and **back end** (*backend*) distinguish between the separation of concerns between the presentation layer (the front end) – which is the interface between the user – and the data access layer (the back end). The front and back ends may be distributed among one or more systems.

Q: What is a back-end server?

A: QUICK ANSWER

A back-end server is a part of the back-end process, which usually consists of three parts: a server, an application and a database. The back end is where the technical processes happen, as opposed to the front end, which is usually where the user's interaction occurs. CONTINUE READING ▼

backend

1. The part of a software product that the user does not interact with.

28

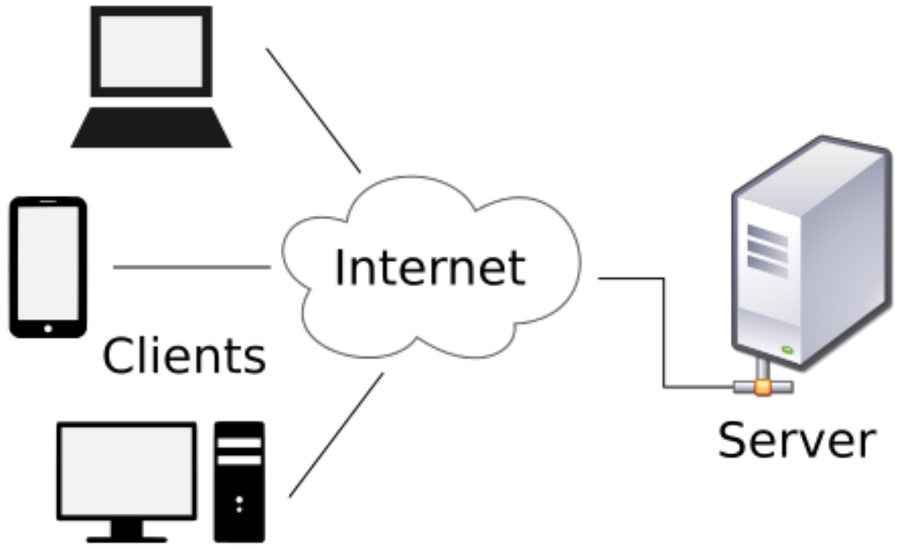
"Front-end" typically means the parts of the project a user interacts with--such as the graphical user interface or command line. It's a vague term, there isn't an exact definition.

"Back-end" means the parts that do the work, but the user is unaware of or cannot see. Databases, services, etc.

Think of it like a restaurant where you can't see the kitchen. As a customer you see the front-end--the decorations, menus, wait-staff. Meanwhile the kitchen and stockroom are out of view, but preparing food.

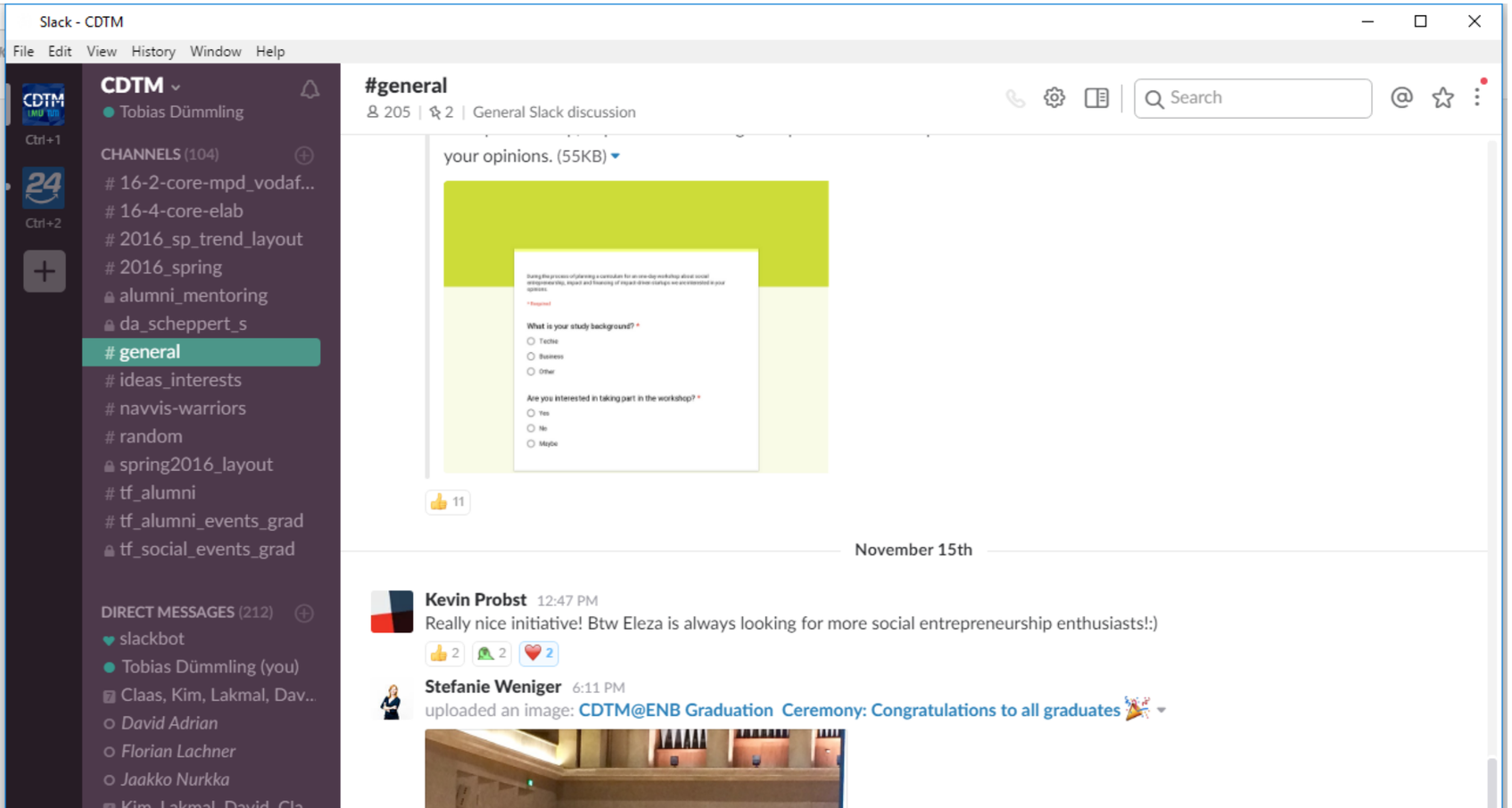
CLIENT-SERVER MODEL

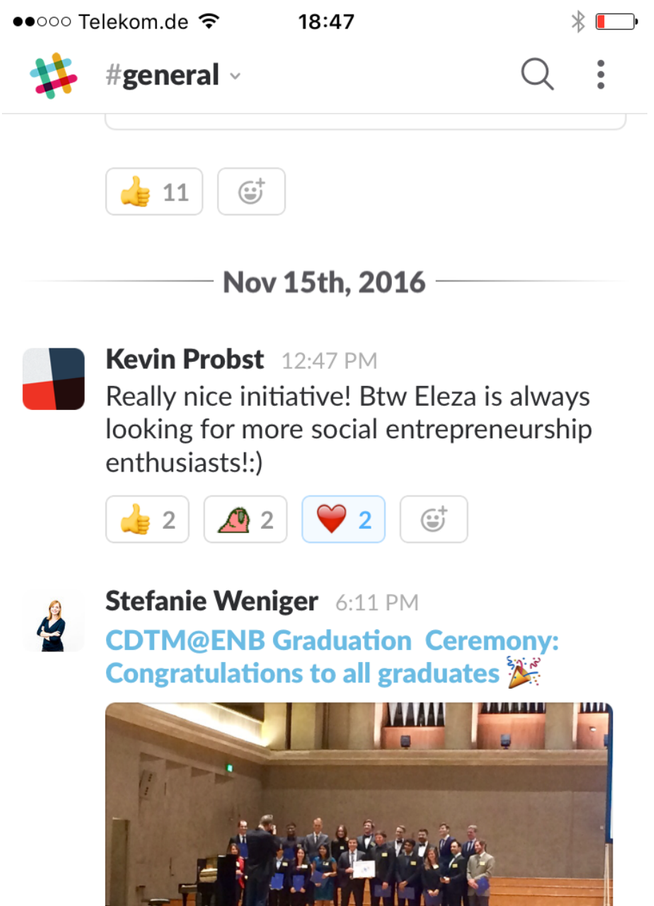
- Most of the time Front-End & Back-End refers to a Client-Server model
- Backend (Server)
  - Offers a *service*
  - e.g. data access / storage
- Frontend (Client)
  - Uses one or more services
  - presents information
  - handles user interaction



THE TWO PARTS – THE FRONTEND

- The Front end is providing a user friend representation of the service you want to offer





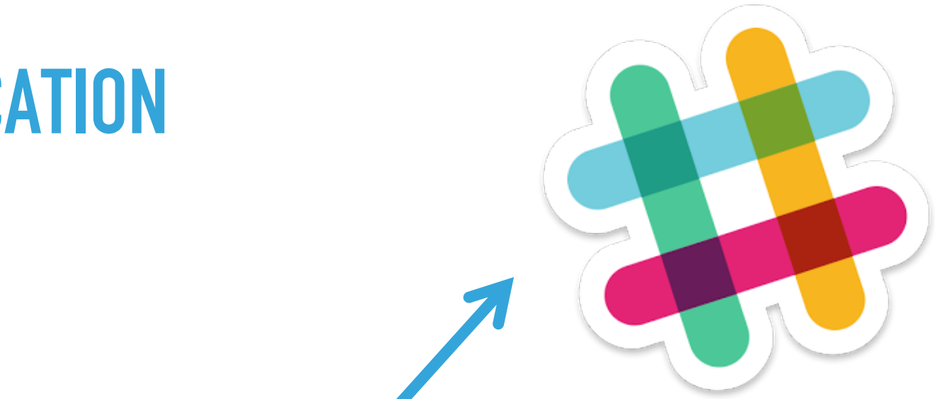
THE TWO PARTS – THE BACKEND

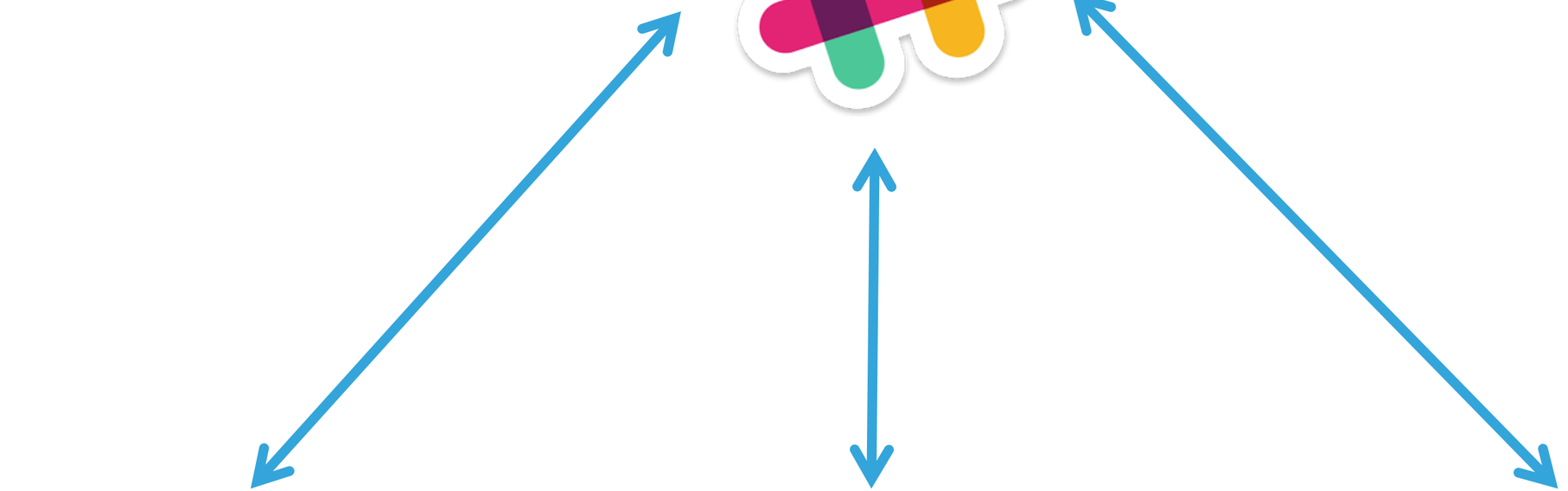
- The Backend provides raw information to the front-end

```
{
    "attachments": [
        {
            "fallback": "Required plain-text summary of the attachment.",
            "color": "#36a64f",
            "pretext": "Optional text that appears above the attachment block"
            "author_name": "Bobby Tables",
            "author_link": "http://flickr.com/bobby/",
            "author_icon": "http://flickr.com/icons/bobby.jpg",
            "title": "Slack API Documentation",
            "title_link": "https://api.slack.com/",
            "text": "Optional text that appears within the attachment",
            "fields": [
                {
                    "title": "Priority",
                    "value": "High",
                    "short": false
                }
            ],
            "image_url": "http://my-website.com/path/to/image.jpg",
            "thumb_url": "http://example.com/path/to/thumb.png",
            "footer": "Slack API",
            "footer_icon": "https://platform.slack-edge.com/img/default_applic
            "ts": 123456789
```

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past' , daywindow,
            totaltweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    for k in j.entities.urls:
```

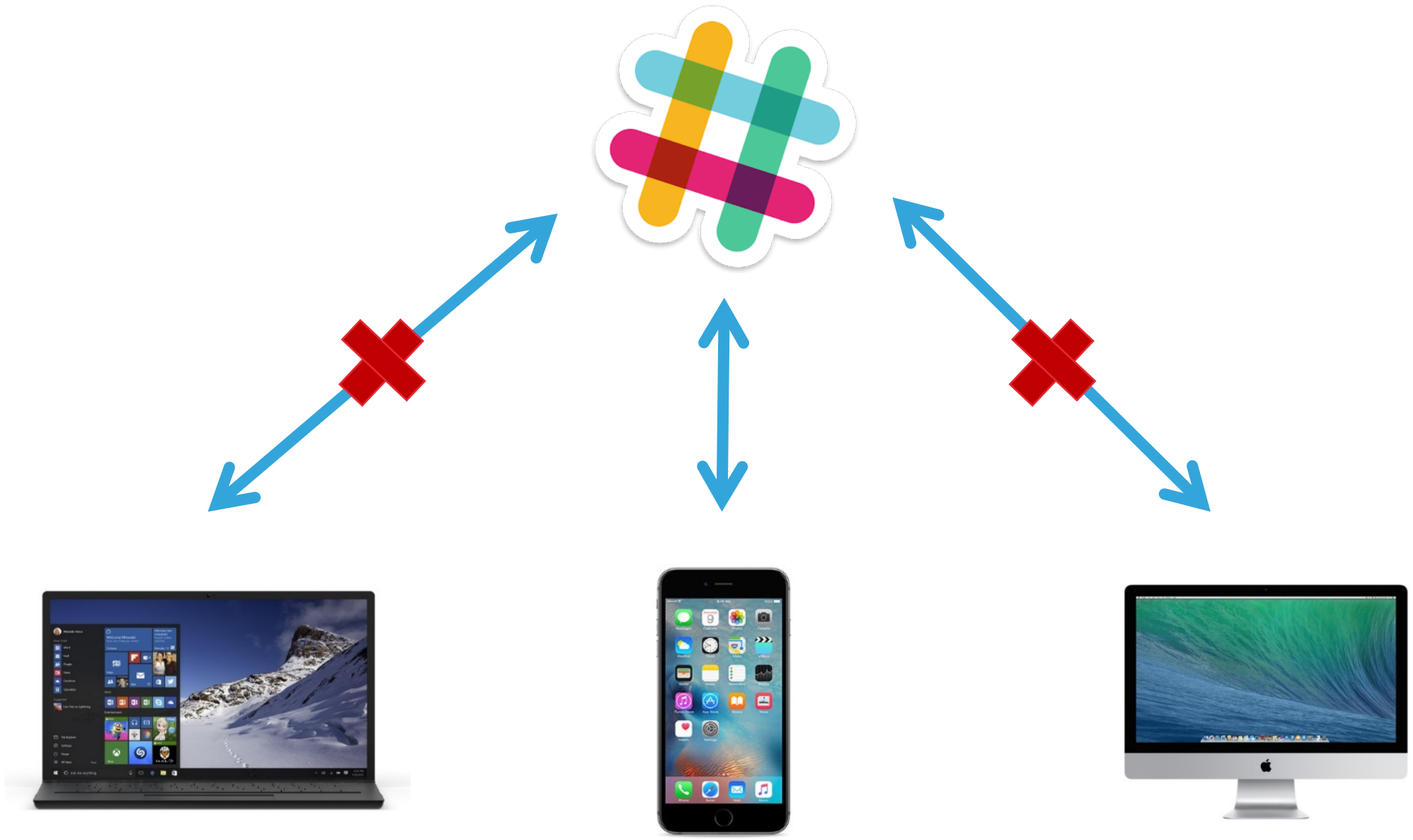
THE COMMUNICATION





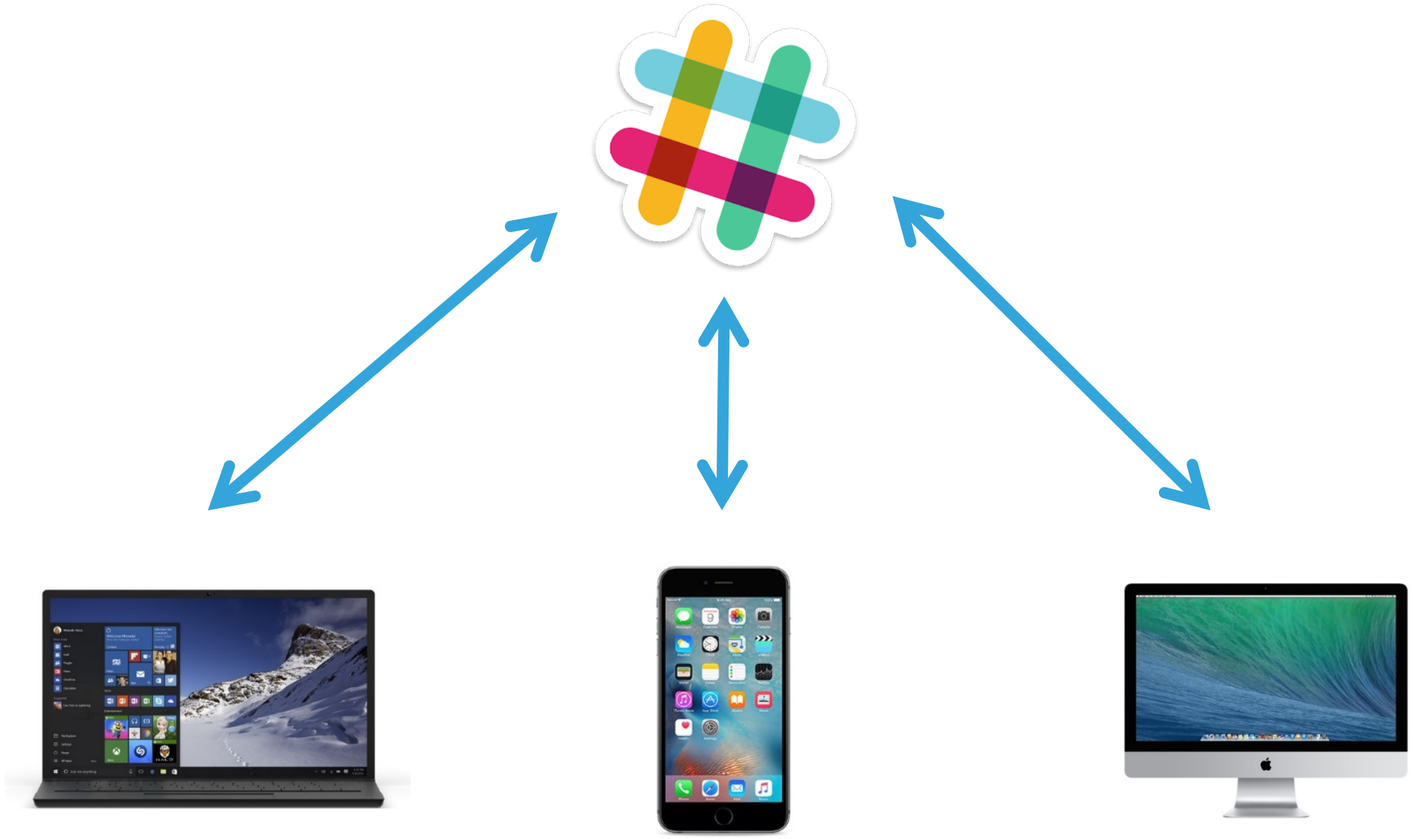
WHY IS THERE A SEPARATION BETWEEN FRONT AND BACK-END

1. Security Reasons: The separation ensures that every user of the service is only able to access the information he is authorized to see



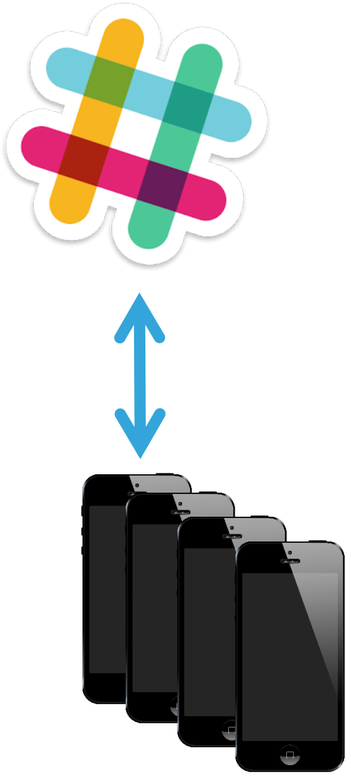
WHY IS THERE A SEPARATION BETWEEN FRONT AND BACK-END

2. Synchronisation: Its easy to synchronize information throughout several devices



WHY IS THERE A SEPARATION BETWEEN FRONT AND BACK-END

3. Scaling: If you keep the **representation** of the information and the **source** of the information separated it is easier to scale them accordingly to increasing demand



WHY IS THERE A SEPARATION BETWEEN FRONT AND BACK-END

4. Technology: Providing information and presenting it can involve different technologies (Programming languages / tools / frameworks etc.).
   They can be developed independent of each other in different teams.

WHY IS THERE A SEPARATION BETWEEN FRONT AND BACK-END

5. Maintenance: Its easier to maintain the program if you separate the two core concerns: **presenting** information and **providing** the data that is needed.

   It allows to exchange either part without affecting the other one.

Example: Two different (front-end) web-applications using the same back-end

- https://mail.google.com
- https://inbox.google.com