Real-Time Web Applications

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

Agenda

- Was ist real time Kommunikation?
- Wie wurde das Problem in der Vergangenheit gelöst?
- Entwickeln einer Chat-Applikation:
    1. Schreiben und Anzeigen von Nachrichten
    2. Senden und Empfangen von Nachrichten (klassisches Client – Server Prinzip)
    3. Automatisches abholen der Nachrichten vom Server (polling)
    4. Serverseitige Push-Notifications
- Weitere Technologien

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

Real-time communication Definition

Real-time communications (RTC) is any mode of telecommunications in which all users can exchange information instantly or with negligible latency. In this context, the term "real-time" is synonymous with "live".

http://whatis.techtarget.com

Unterscheidung zu:

- Real-time Anwendungen:
  - Garantieren das Ausführen einer Operation in einer vorgegebenen Zeit
  - Nicht dasselbe wie High-Performance
  - Fahrassistenz Systeme/Autopiloten, Streaming
- Real-time über Netzwerke:
  - Spezielle Netzwerke können maximale Zustellzeiten garantieren
  - Gilt **nicht** für das Internet

Heutiges Thema:

„Soft-Real-Time”: wir wollen so **zeitnah** wie möglich und **verlässlich** auf Userinteraktionen oder serverseitige Ereignisse reagieren – **ohne** dabei jedoch eine bestimmte **Zustellzeit zu garantieren**

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

Use cases

- Online Office Suites, Cloud IDEs,…
- Visualisierungen (z.B. live-Diagramme)
- Finanz-Anwendungen
- Games
- Instant Messaging
- Nachrichten
- Liveticker
- …

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

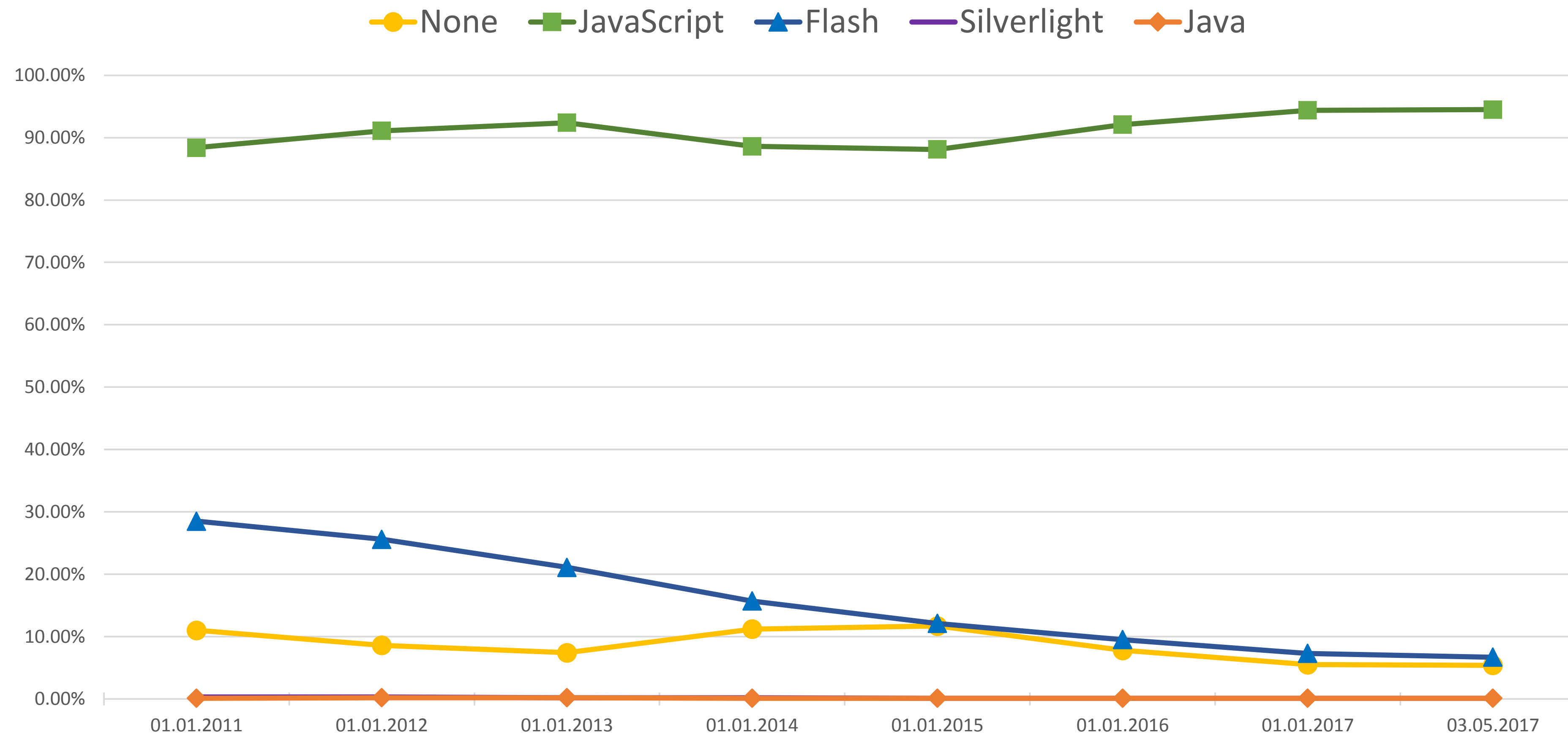
I‘m not sure I believe that there is such a thing as „realtime apps“ anymore. Apps either update instantly and smoothly, or they appear broken.

Max Williams – CEO Pusher

In der Vergangenheit

- HTML und HTTP vor HTML5:
  - **Post-back basiert (kann mittels AJAX umgangen werden)**
  - **Unidirektional (Client → Server)**
  - Keine Audio-/Video-Wiedergabe
  - Zeichnen von komplexen Objekten nicht möglich
  - 3D Zeichnungen nicht möglich
- Lösungen:
  Java Applets, Flash oder Silverlight Anwendungen

Clientseitiger Code: Marktentwicklung



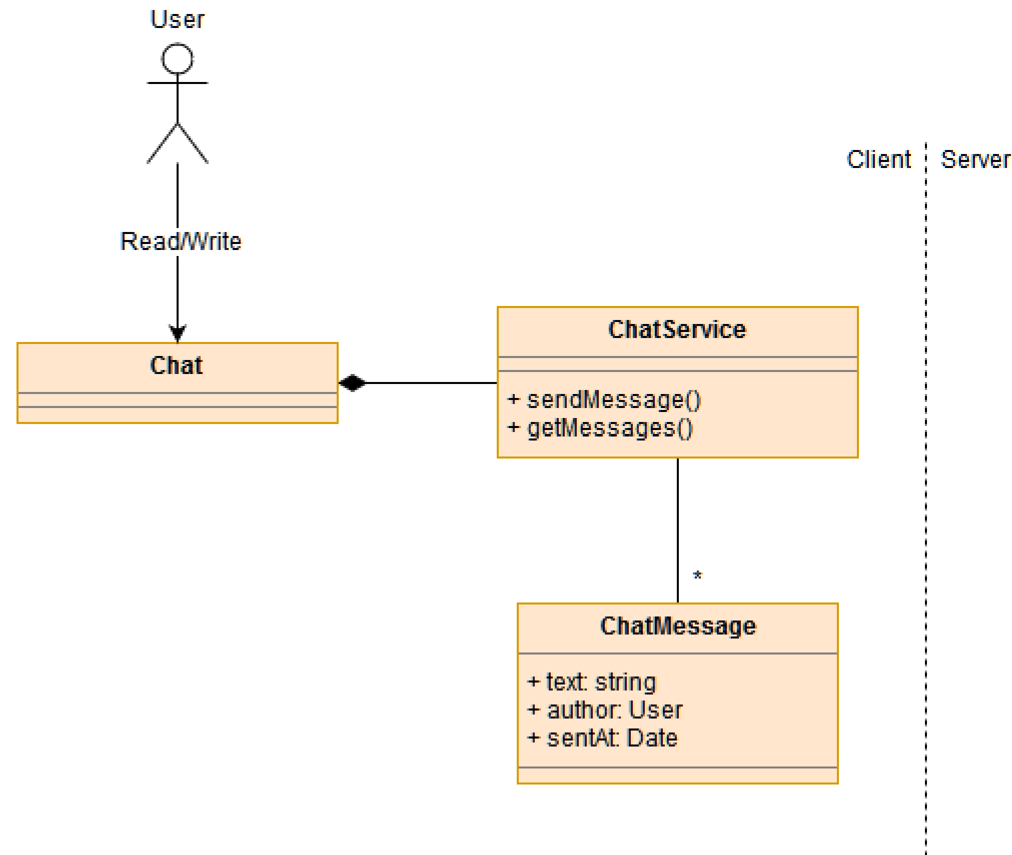
Quelle: w3techs.com

Technologie-Stack

	Layer	Technologie	Alternativen
Client-side	UI-Layer	Twitter Bootstrap	Foundation, Pure, HTML Kickstart, …
	Client side	Angular (4) mit TypeScript	React, Angular 1, Backbone, …
Server-side	“Real-Time” communication	SignalR Core	SignalR, Pusher, socket.io, …
	Web Framework	ASP.NET Core	ASP.NET 4.6, Spring, Node.js, …
	Runtime	.NET Core	.NET, Java, Node.js, …

- Clientseitiges MVC Framework von Google
- Starke Konzentration auf Data-binding
  - →Ideal in Echtzeitumgebungen, da das Template automatisch aktualisiert wird, wenn neue Daten vom Server eintreffen

V1: Lokale Angular Anwendung



Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

V1: Lokale Angular Anwendung

```
<div class="chatMessage" *ngFor="let msg of chatMessages“
    [class.author]="msg.isAuthor(user)">
  <div class="chatUser">{{msg.username}}:</div>
  <span class="chatText">{{msg.text}}</span>
  <div class="chatDate">{{msg.sentAt | date: "HH:mm"}}</div>
</div>
```

```
sendMessage(): void {
  let message = new Message(this.user.username, this.user.sessionId,
    this.messageText, new Date());
  this.chatService.sendMessage(message);

  this.messageText = "";
}
```

V1: Lokale Angular Anwendung

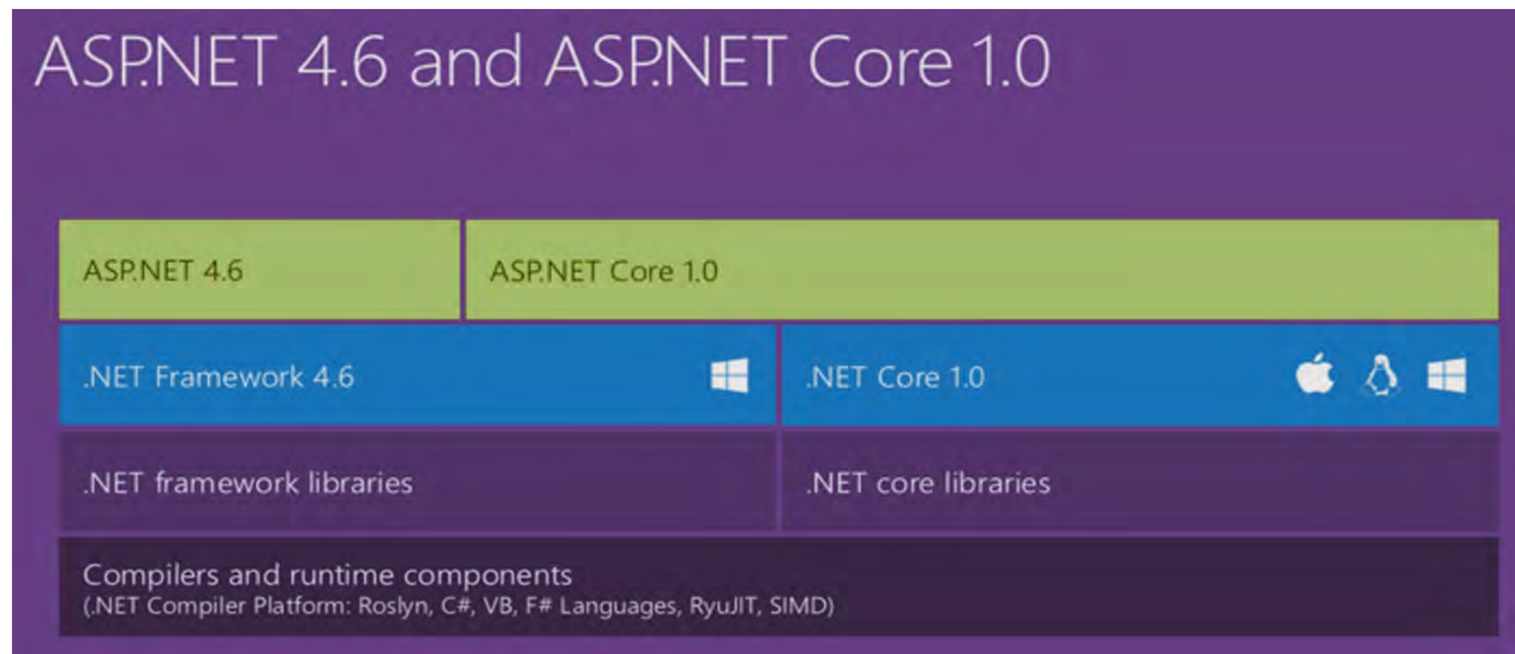
- Noch keine serverseitige Komponente
- Komponente, welche alle Nachrichten darstellt
- Nachrichten werden in einem Service gespeichert

https://tinyurl.com/aka-local

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

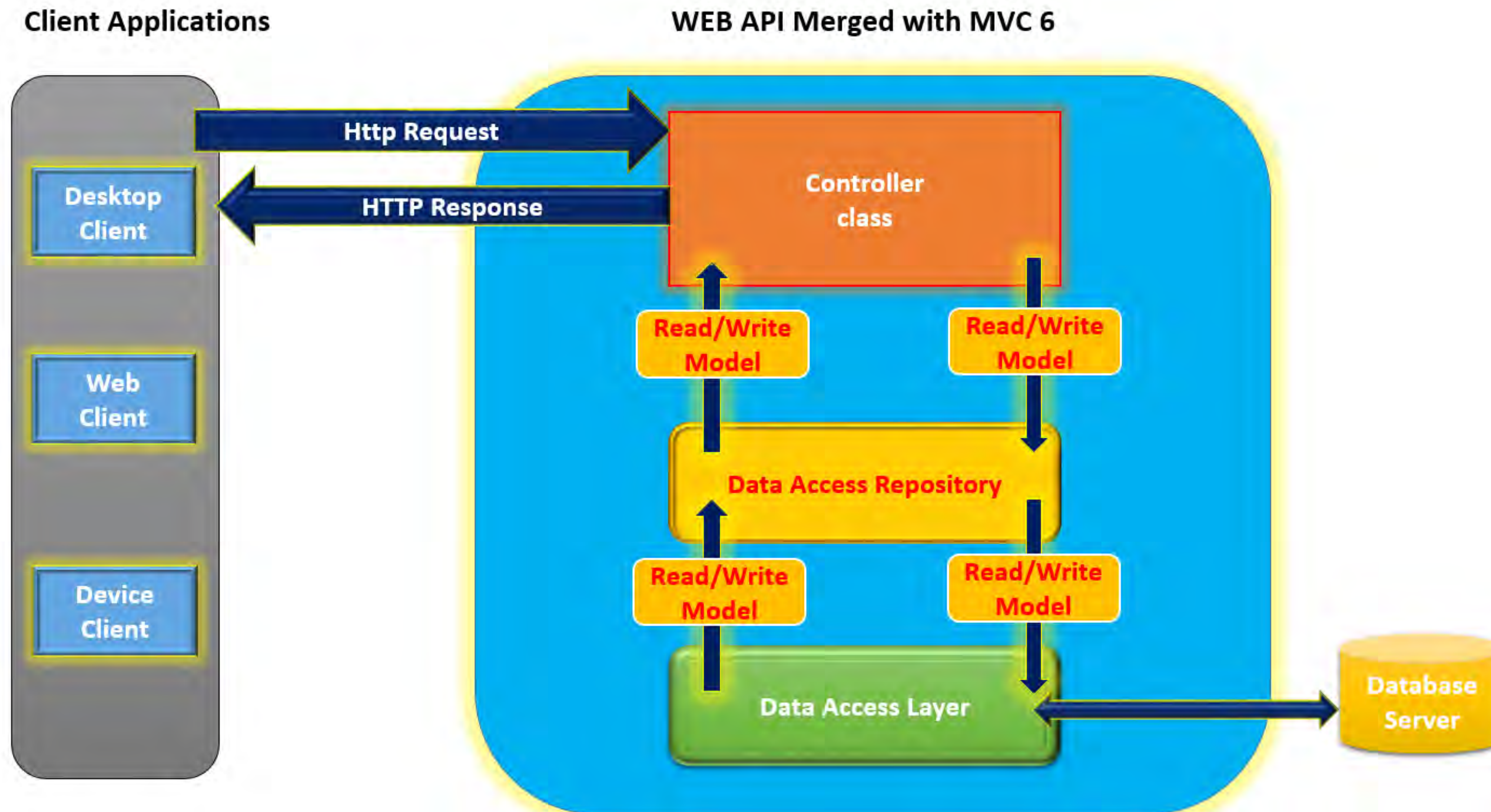
ASP.NET Core

- Open Source
- Multi-Plattform
- Komplette Neu-Entwicklung von ASP.NET
- .NET Framework und .NET Core kompatibel
- Zusammenlegung von MVC und Web API



Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

ASP.NET Core



Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

V2: Klassische Client-Server Anwendung

```
public class ChatController : Controller
{
  private static List<ChatMessage> messages = new List<ChatMessage>();
  [HttpGet]
  public IEnumerable<ChatMessage> ReceiveMessages()
  {
    return messages;
  }
  [HttpPost]
  public void SendMessage([FromBody] ChatMessage message)
  {
    if (message != null && !string.IsNullOrEmpty(message.Text))
    {
      messages.Add(message);
    }
  }
}
```

TypeScript Interfaces

Optional:
Um über die Grenzen zwischen Client und Server die Typsicherheit nicht zu verlieren generieren wir aus unseren C#-Klassen automatisiert TypeScript-Interfaces, welche wir in unseren TypeScript-Objekten implementieren:

```
public class ChatMessage
{
  public string Username { get; set; }

  public string SessionId { get; set; }

  public DateTime SentAt { get; set; }

  public string Text { get; set; }
}
```

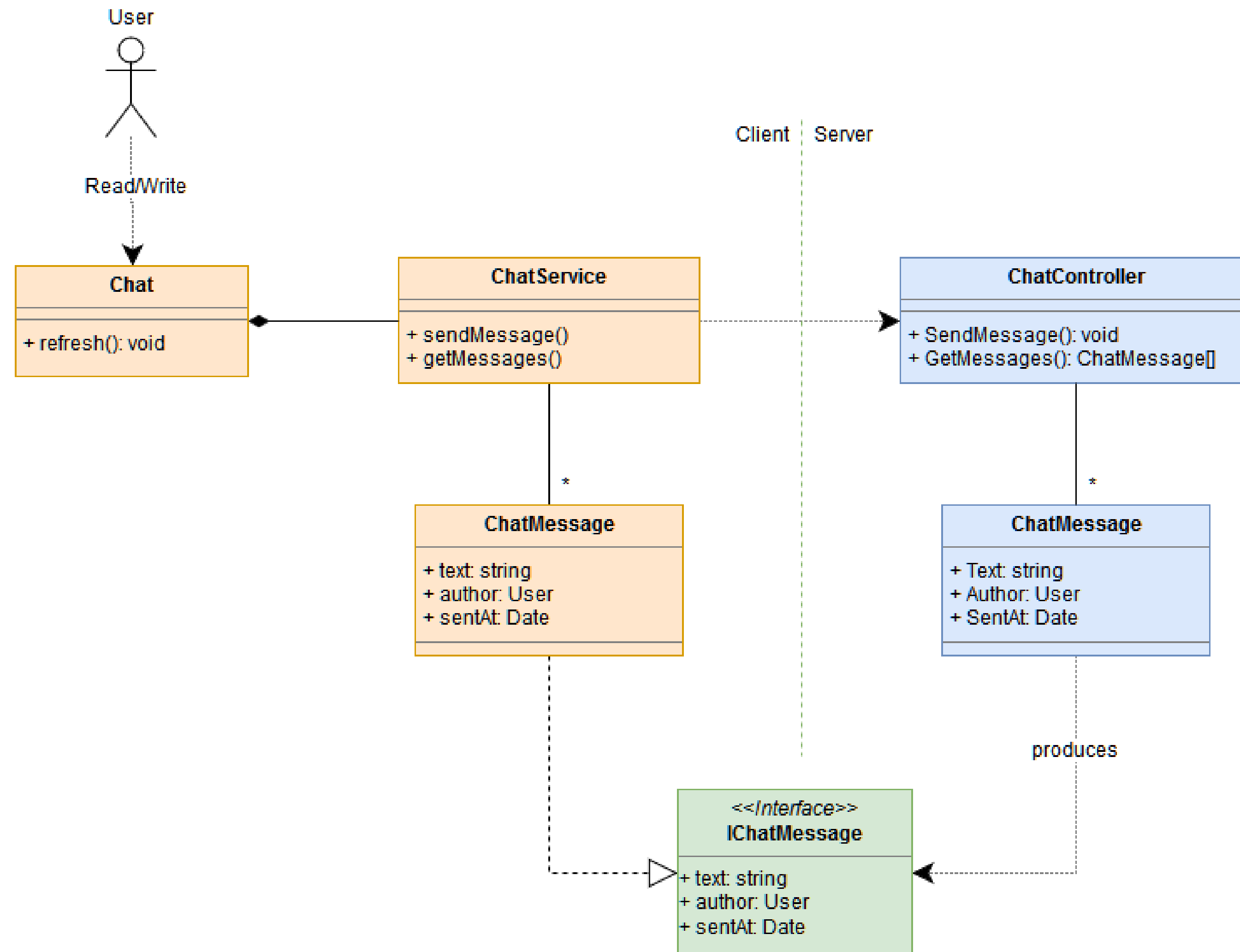
```
interface IChatMessage {
  username: string;

  sessionId: string;

  sentAt: Date;

  text: string;
}
```

V2: Klassische Client-Server Anwendung



V2: Klassische Client-Server Anwendung

- Neu: Serverseitige Komponente
- Controller mit Operationen:
  - Lesen von Nachrichten
  - Senden einer Nachricht
- Client liest Nachrichten auf Kommando und speichert sie lokal ab
- Senden einer Nachricht: geht direkt an den Server

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

https://tinyurl.com/aka-refresh

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

Polling

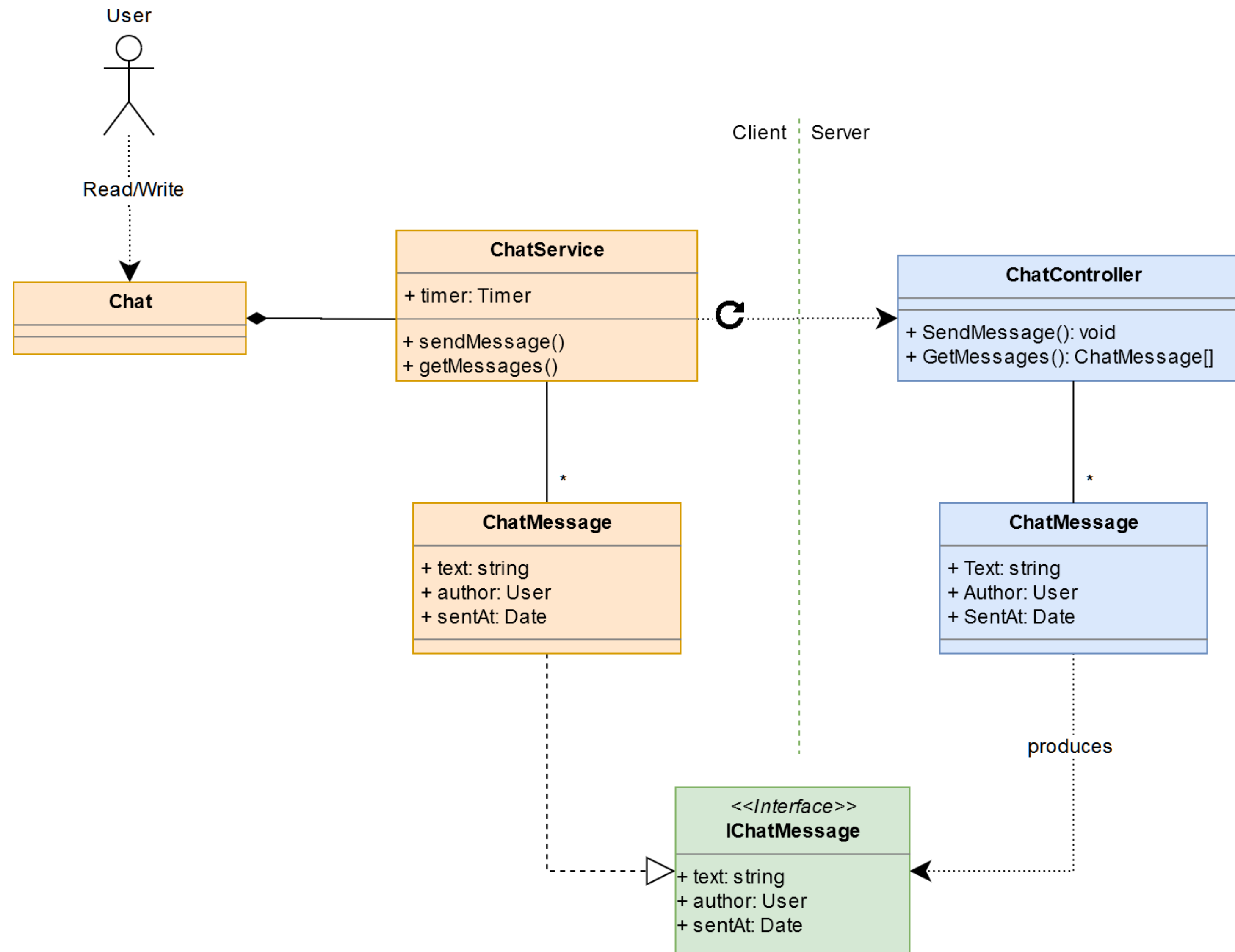
- HTTP basiert
- Simple/Traditionelle Variante
- In regelmäßigen Abständen wird der Server gefragt ob es etwas neues gibt
- Jede Verbindung geht vom Client aus
- Verursacht viel Traffic
- Hoher Ressourcenverbrauch

V3: Automatischer refresh (polling)

```
ngOnInit(): void {
    this.retrieveMessages();

    window.setInterval(() => this.retrieveMessages(), 1000);
}
```

V3: Automatischer refresh (polling)



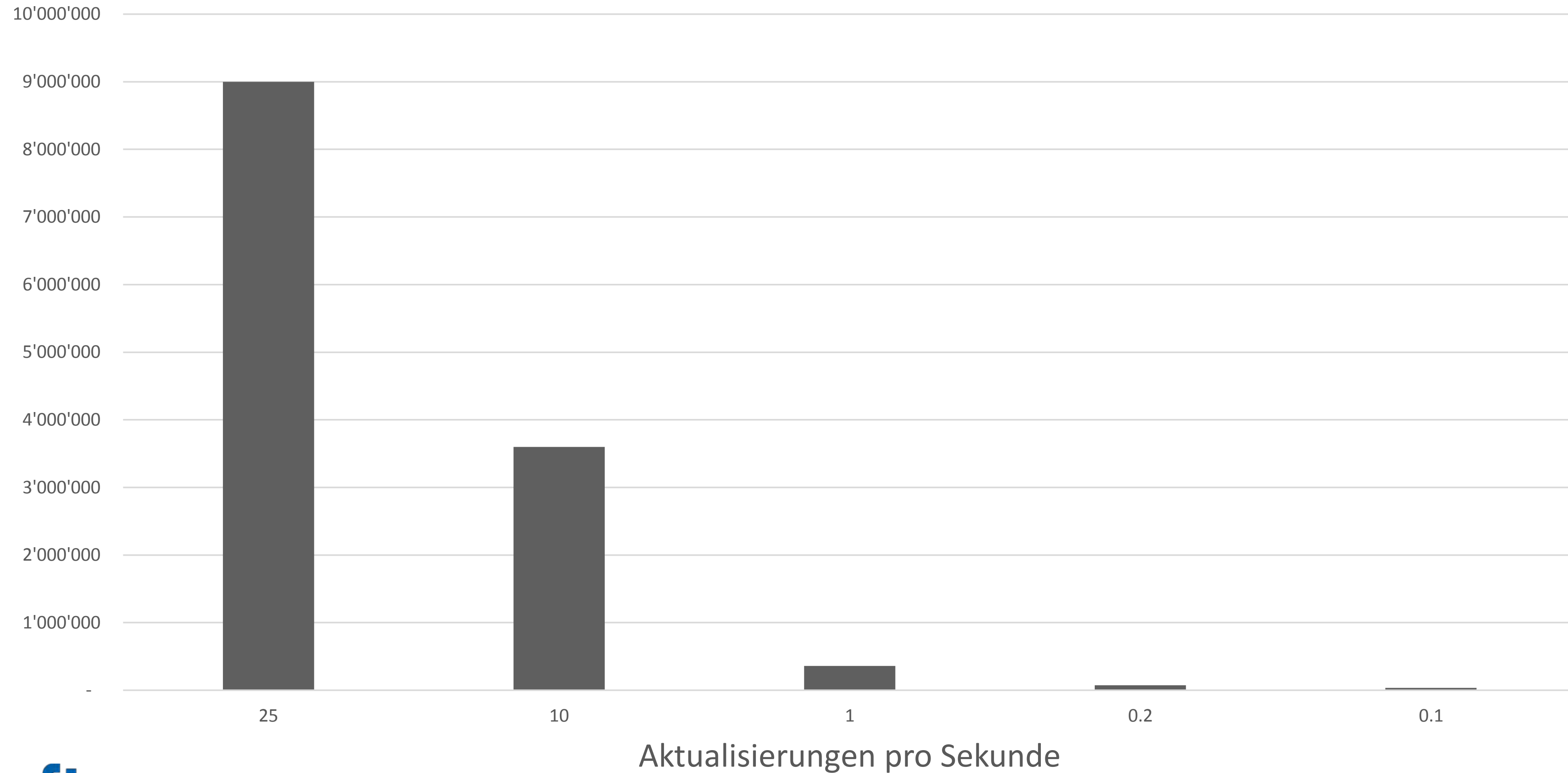
creasoft ag
Software: Planen. Entwickeln. Testen.

V3: Automatischer refresh (polling)

- Client:
    - Manueller Aufruf von GetMessages() wird durch einen Aufruf in einer Schleife ersetzt

https://tinyurl.com/aka-polling

Requests pro Stunde bei 100 Benutzern

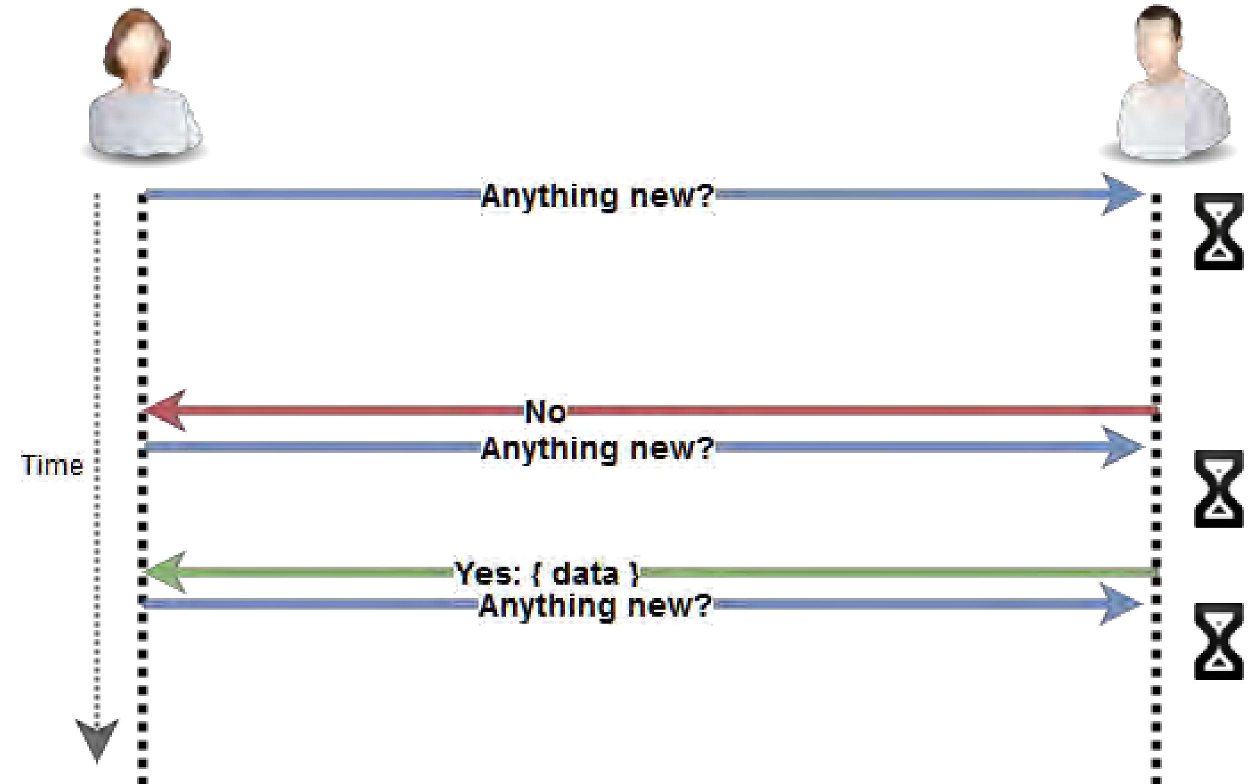


Real-Time Technologien

- Polling
- Long Polling
- Server-Sent Events
- Forever Frame
- WebSockets

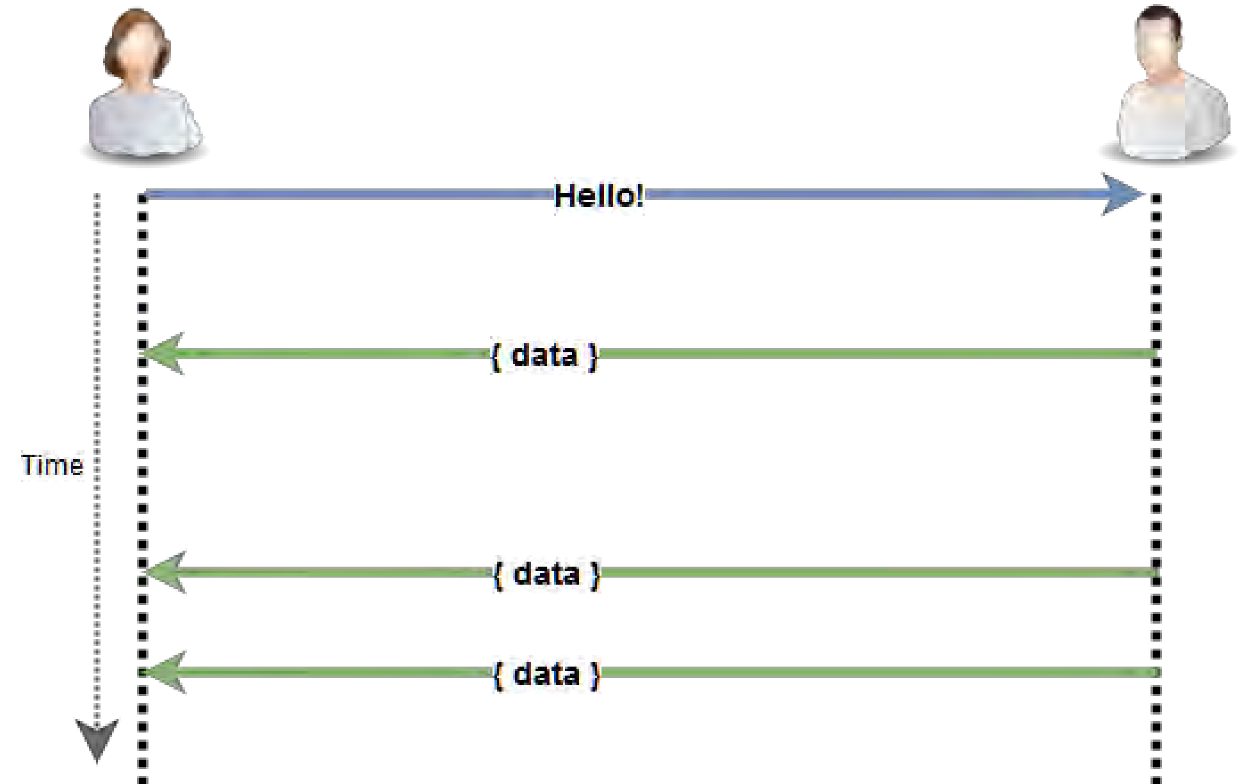
Long Polling

- HTTP basiert
- Client initiiert Verbindung
- Server antwortet nicht direkt auf jeden Request
  - Erst wenn Daten bereitstehen, wird eine Antwort gesendet
  - Oder nach einem Timeout
- Nach jeder Response (ob Daten oder Timeout) setzt der Client einen neuen Request ab



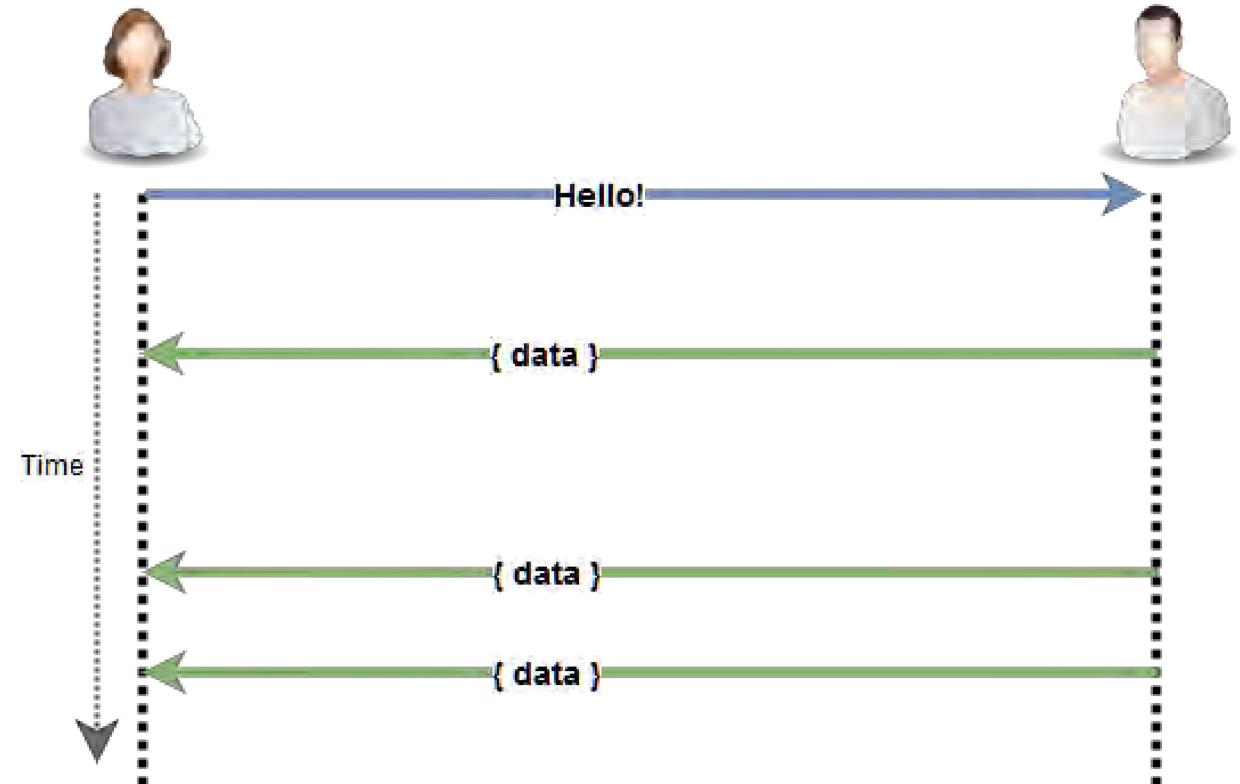
Server-Sent Events

- HTTP-basiert
- Unidirektionale Kommunikation
- Client öffnet Verbindung zum Server und behält diese offen
- Server nützt die offene Verbindung um Daten an den Client zu senden
- Client → Server Kommunikation läuft auf einem eigenen Kanal mittels AJAX-Requests



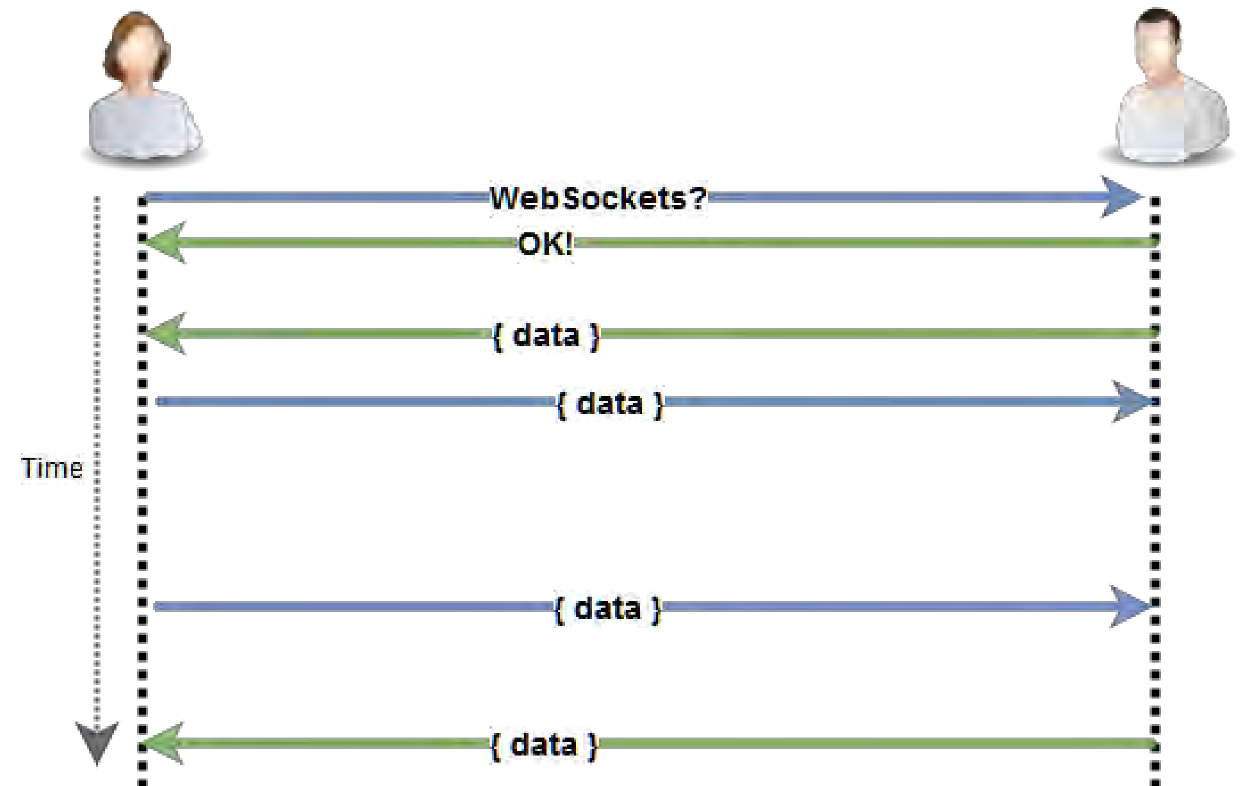
Forever Frame

- HTTP basiert
- Unidirektionale Kommunikation
- Unsichtbares iframe auf Webseite
- Inhalt des iframes wird in “chunks” geliefert
  - HTTP 1.1 Feature
  - Dafür gedacht, wenn Response-Size nicht bekannt ist
- Jede einzelne Response enthält eine kodierte Nachricht vom Server an den Client
- Client → Server Kommunikation läuft auf einem eigenen Kanal mittels AJAX-Requests



WebSockets

- TCP basiert
- Bidirektionale Kommunikation
- Anfrage bezüglich WebSockets wird an Server gesendet
- Protokoll wird von HTTP zu WS gewechselt
  (spezieller Support in Server und Client nötig)
- Frames werden über die offene Verbindung in beide Richtungen ausgetauscht



Browser Support

	Long polling	Server-Sent Events	Forever Frame	WebSockets
IE 8	✓		✓	
IE 11	✓		✓	✓
Edge 15	✓		✓	✓
Firefox 53	✓	✓		✓
Chrome 58	✓	✓		✓
Chrome 57 (Android)	✓	✓		✓
Safari 10	✓	✓		✓
iOS Safari 10	✓	✓		✓

Server Support

	Long polling	Forever Frame	Server-Sent Events	WebSockets
IIS 7.5	✓	✓	✓	
IIS 8	✓	✓	✓	✓
Tomcat 6	✓	✓	✓	
Tomcat 7	✓	✓	✓	✓
Apache 2.3	✓	✓	✓	
Apache 2.4	✓	✓	✓	✓
Node.js	✓	✓	✓	✓

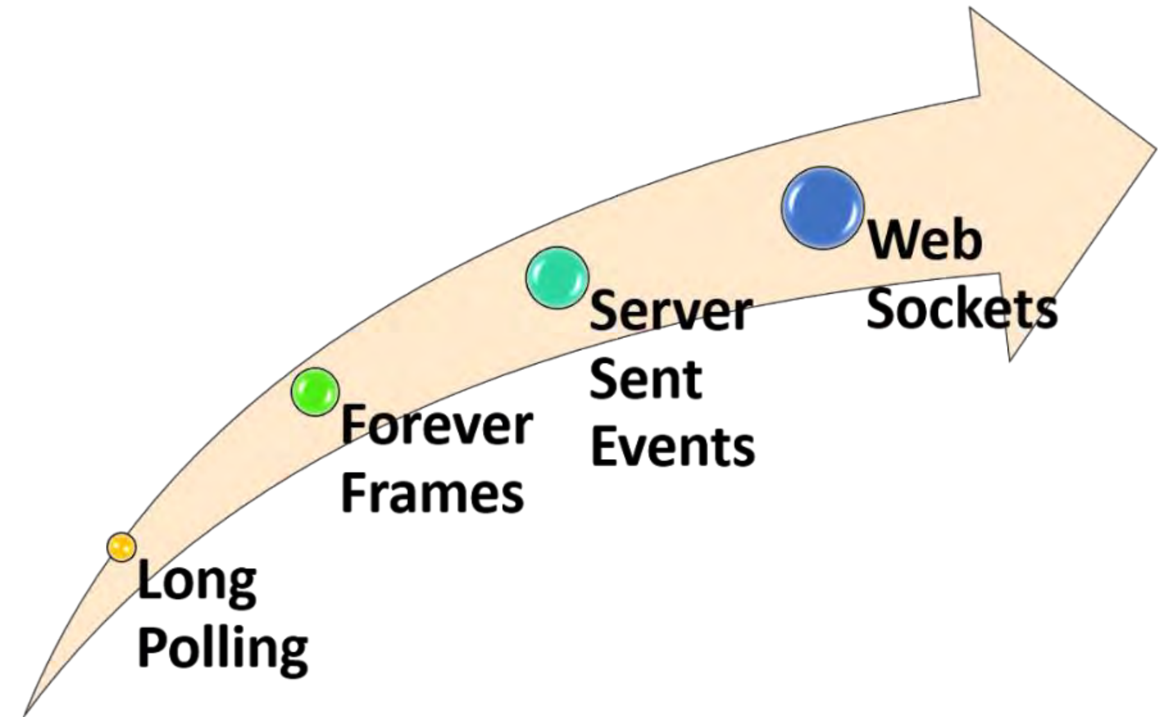
Übersicht

	Long-polling	Server-sent Events	Forever Frame	WebSockets
Client Support	Alle Browser	Aktuelle nicht-MS Browser	Aktuelle MS Browser	Alle aktuellen Browser
Server Support	Alle	Alle gängigen	Alle gängigen	Alle gängigen
Richtung	Client → Server Client ← Server	Client ← Server	Client ← Server	Client ↔ Server
Reaktionszeit	Delay zwischen dem Timeout einer Verbindung und dem nächsten Verbindungsaufbau	Default: max. 3s (konfigurierbar)	Leicht zeitverzögert (abfragen von Daten im iFrame)	"Real-time"

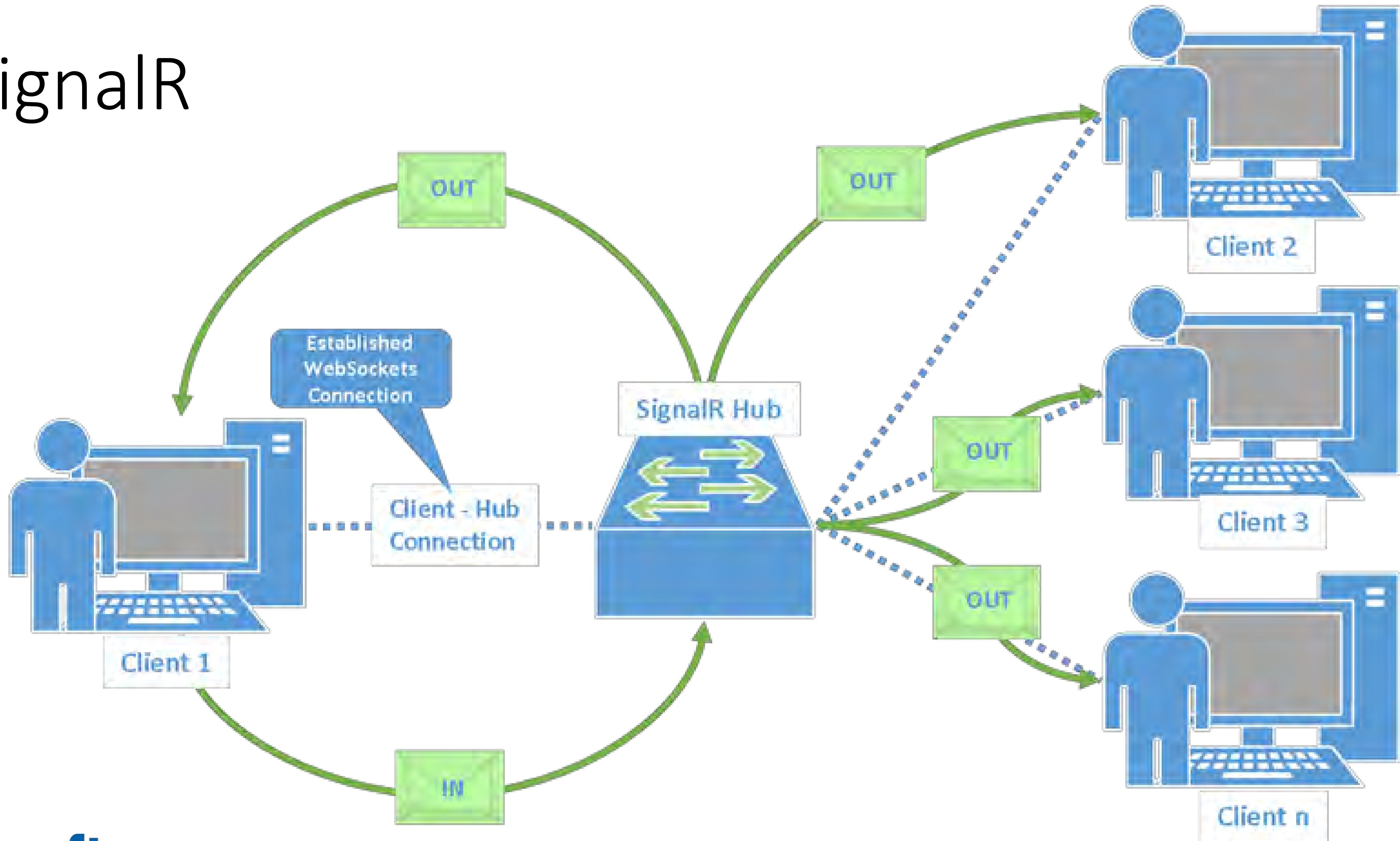
Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

SignalR

- Abstraktion und Vereinheitlichung der vorgestellten Technologien
- Client und Server kommunizieren über einen „Hub“
- Zu verwendende Technologie wird beim Verbindungsaufbau ausgehandelt



SignalR



V4: Server-Client Kommunikation mit SignalR

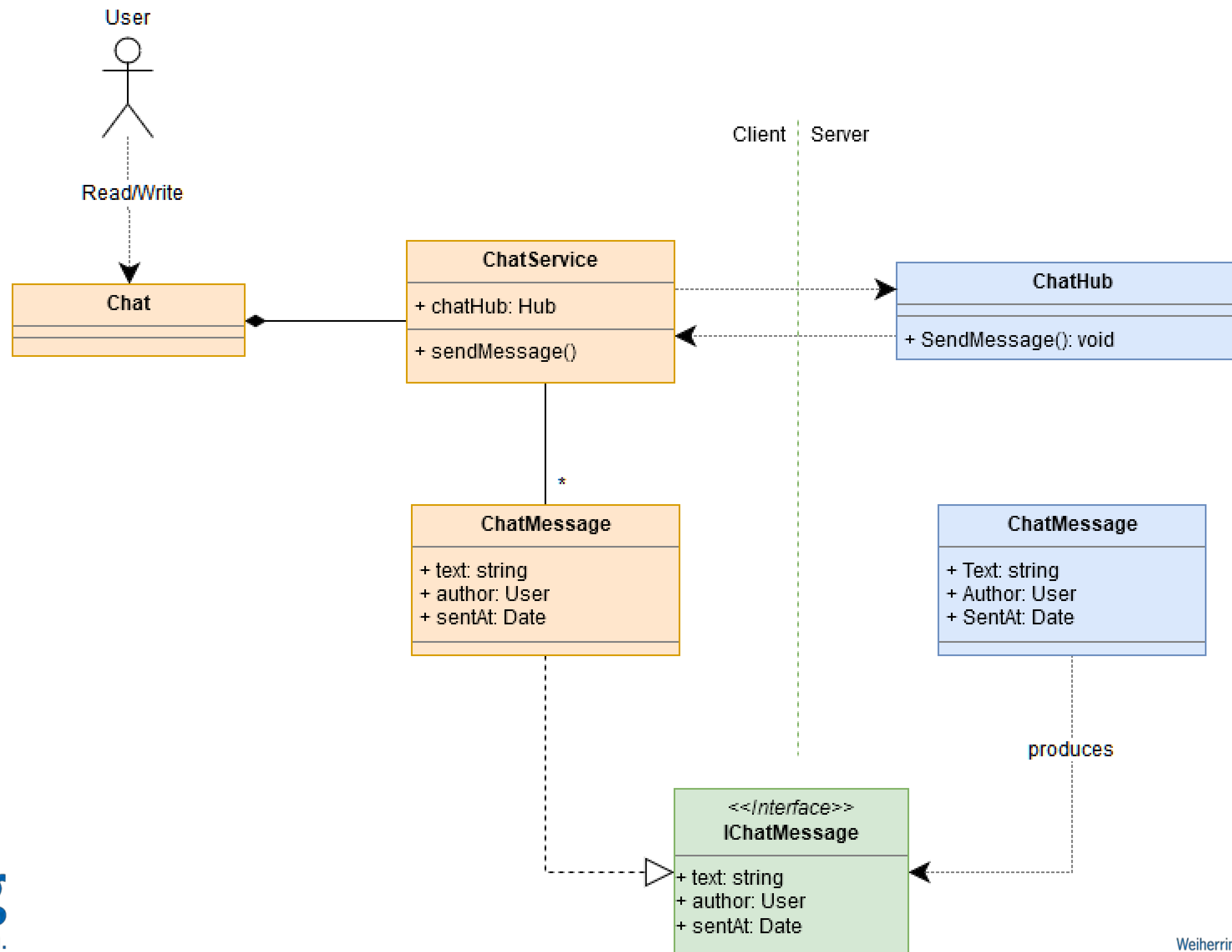
Client

```
export class ChatService {
  constructor(private http: Http) {
    $.connection.chatHub.client
       .broadcast = this.receive;
  }
  sendMessage(msg: Message): void {
    $.connection.chatHub.server
      .sendMessage(msg);
  }
  receive = (msg: ChatMessage): void => {
    this.component.messages.push(msg);
  };
}
```

Server

```
public class ChatHub : Hub
{
    public void SendMessage(
        ChatMessage message)
    {
      this.Clients.All.broadcast(message);
    }
}
```

V4: Server-Client Kommunikation mit SignalR



V4: Server-Client Kommunikation mit SignalR

- Server:
  - Ersetzen des ASP.NET Controllers durch einen SignalR Hub
  - Nachrichten werden direkt verteilt
- Client:
  - Einbinden des auto-generierten JavaScript Codes für den Hub
  - Aufrufen von sendMessage() und setzen des broadcast() Callbacks anstelle von regulären Service-calls

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

https://tinyurl.com/aka-final

Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

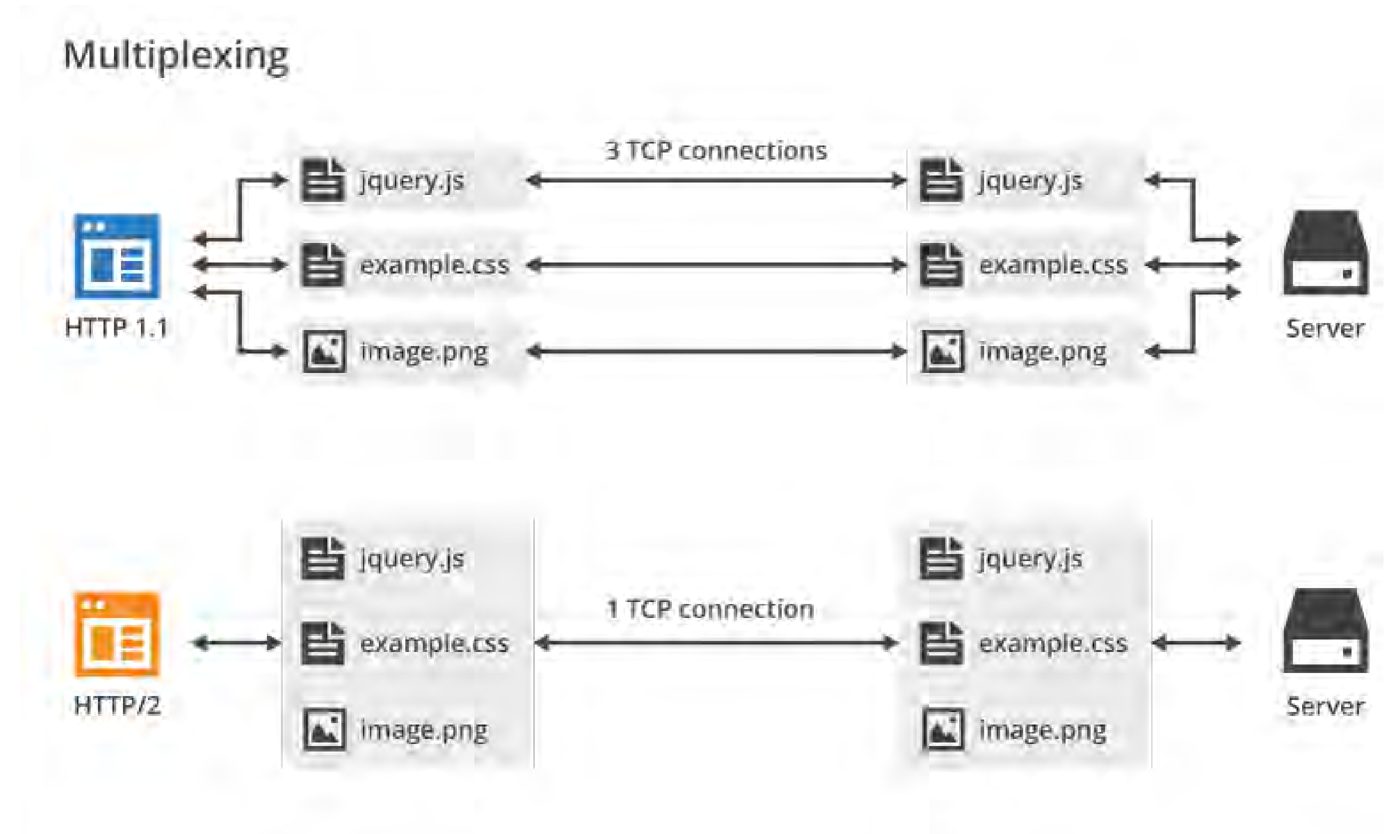
WebRTC

- „Web Real-Time Communication“
- Offener Standard für End to End Verbindungen
- Kommunikation mittels RTP
- Hauptanwendungszweck: Streaming
- Benötigt Server mit eigenem Protokoll für den Verbindungsaufbau

Browser	Support
IE 11	
Edge (ab 15)	✓
Firefox	✓
Chrome	✓ (Browser prefix)
Opera	✓ (Browser prefix)
Safari	

HTTP/2

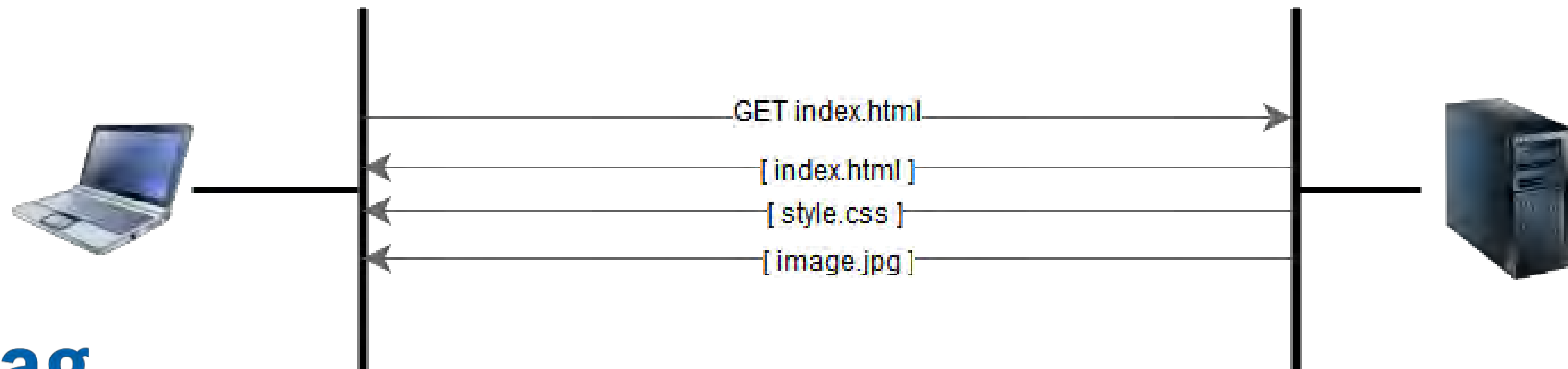
- Erweiterung des HTTP Standards
- Binäres Format
- Erlaubt komprimieren der gesamten Nachricht
- Multiplexing Support
  → Vereinfacht Long polling
- Server side push



HTTP/2 - Server side push

HTTP 1.1:

HTTP/2:



Weiherring 115 . FL 9493 Mauren . T +423 375 8200 . www.creasoft.li

Demo: High Frequency Updates

https://tinyurl.com/aka-circles