

Chat Interattiva con Angular

Kristian Boldini – Ahmed El Aidy

Introduzione ad Angular


- Sviluppato da Google nel 2016
- Basato su TypeScript




Caratteristiche

- Architettura a Componenti
- Command Line Interface (CLI)
- Reattività degli Stati
- Direttive e Binding dei Dati
- Dependency Injection (DI)
- Ampia libreria di componenti e moduli

WebChat

ChatterBox


Search

Channel 1

Channel 2

Channel 3

04/09/24 17:22


Lou

Marvin Berry

Hey guys, did either of you catch the football match last night?



Yes, I did! It was incredible, wasn't it?

04/09/24 17:33

Mark

Oh, I missed it. What happened?

04/09/24 17:33



Hey guys, did either of you catch the football match last night?

04/09/24 17:33

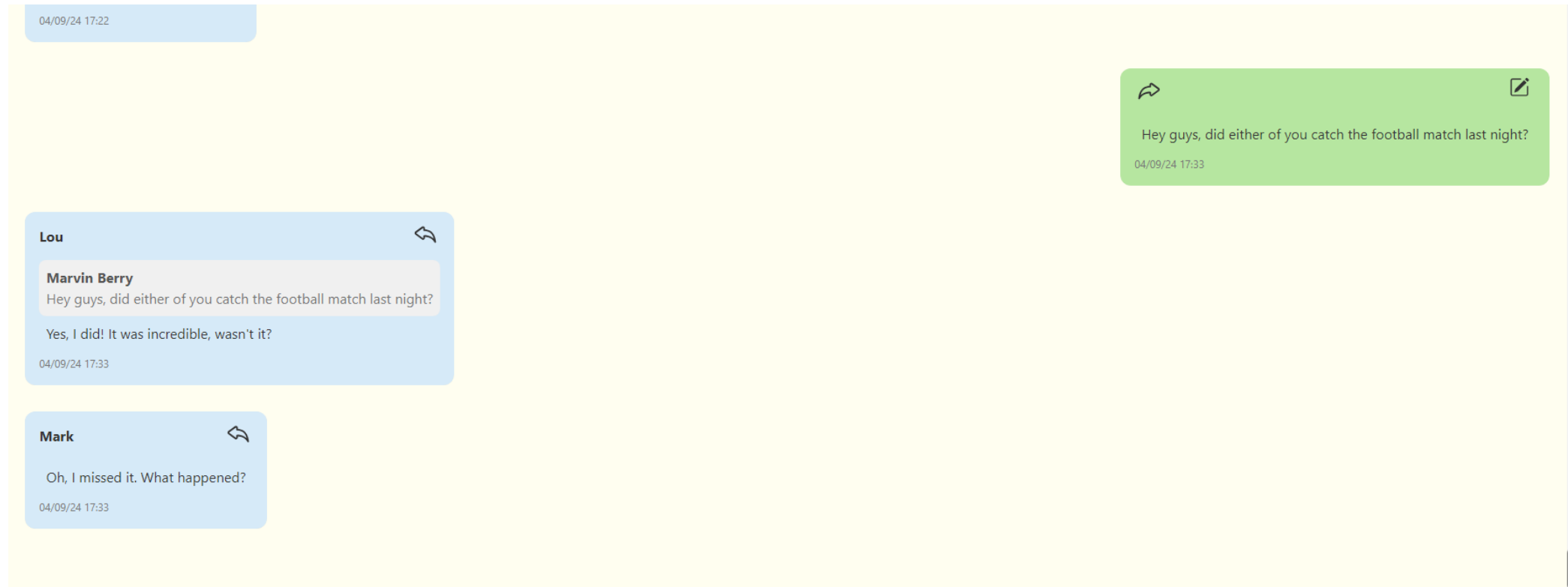
Type your message...

Choose File

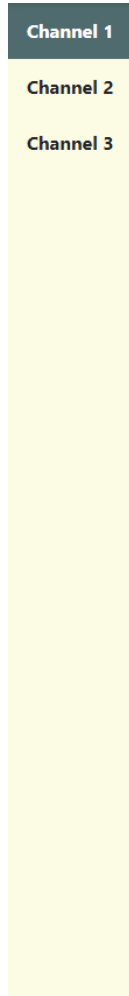
Send

Message List Component

- Gestione e visualizzazione dei messaggi dei canali



Channel list Component



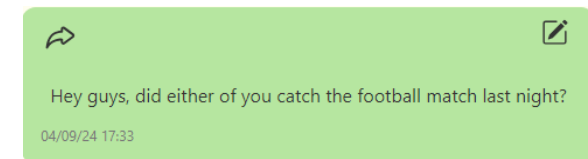
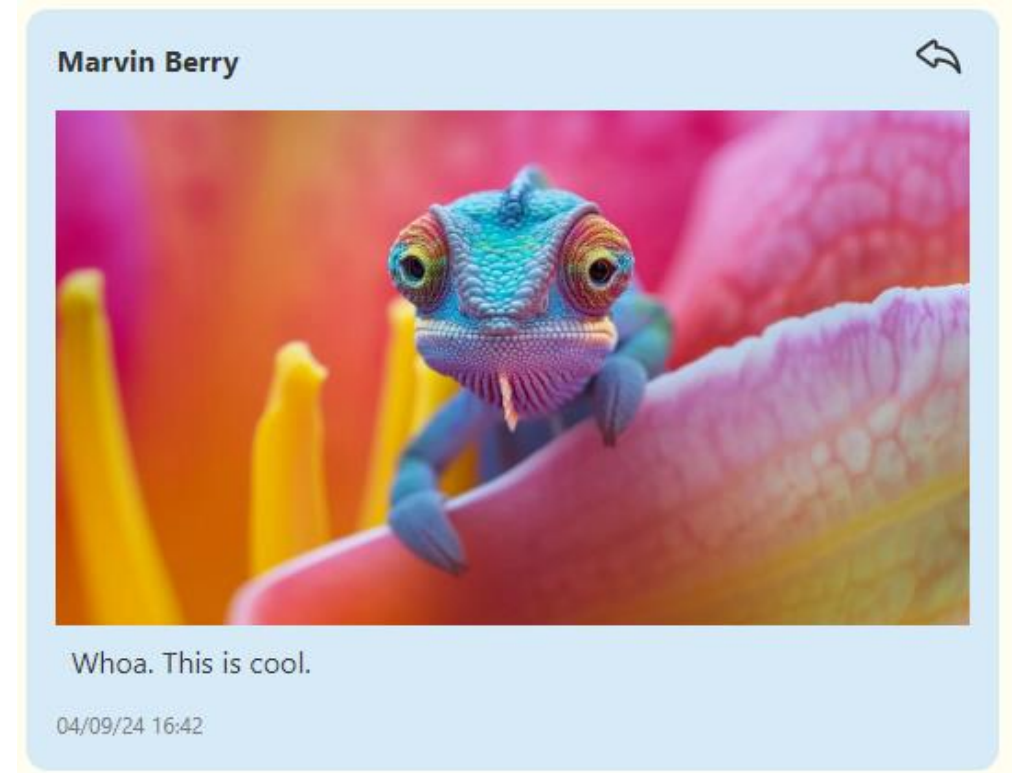
- Gestione e visualizzazione dell'elenco canali

```
getChannels(): void {  
  this.service.getChannels()  
    .subscribe(channels => {  
    this.channels = channels  
  });  
}
```

```
<div class="channel-list" *ngIf="channels.length > 1 && showChannels">  
  <div *ngFor="let channel of channels" class="channel-item" (click)="changeChannel(channel.id)"  
    [class.selected]="currentChannel?.id == channel.id">  
    {{ channel.name }}  
    <span *ngIf="unreadMessageCounts[channel.id] > 0" class="badge">  
      {{ unreadMessageCounts[channel.id] }}  
    </span>  
  </div>  
</div>
```

Message Component

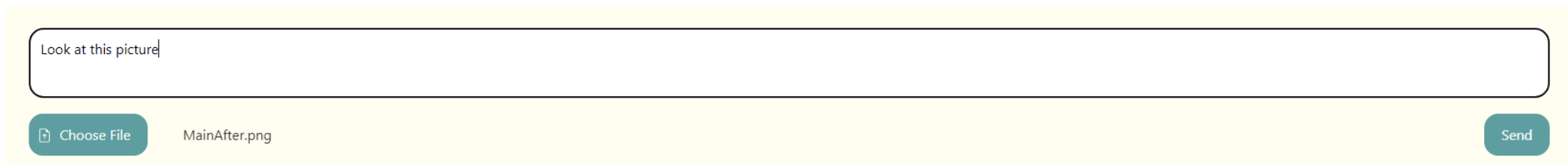
- Gestione e visualizzazione di un messaggio



Input Component

- Consente di comporre e inviare messaggi con testo e allegati
- Rispondere ai messaggi
- Modificare i messaggi

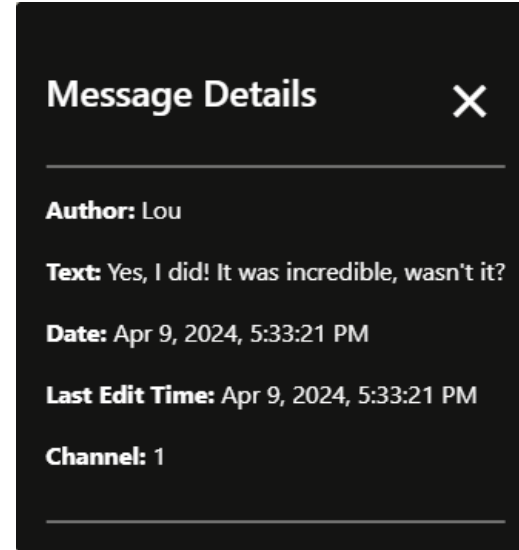
```
sendMessage() {  
  this.messageSent.emit({ text: this.messageText, file: this.selectedFile });  
  this.messageText = '';  
  this.selectedFile = null;  
  this.messageToReply = undefined;  
  this.clearFileInput();  
}
```



A UI mockup of a message input component. It features a large, rounded rectangular text input field with a thin black border. Inside the field, the text "Look at this picture" is visible, followed by a vertical cursor. Below the input field, there is a horizontal bar containing three elements: a teal button with a file icon and the text "Choose File", the filename "MainAfter.png" in a standard font, and a teal button with the text "Send". The entire input area is set against a light yellow background.

Message Details Component

- Visualizzazione dei dettagli di un messaggio



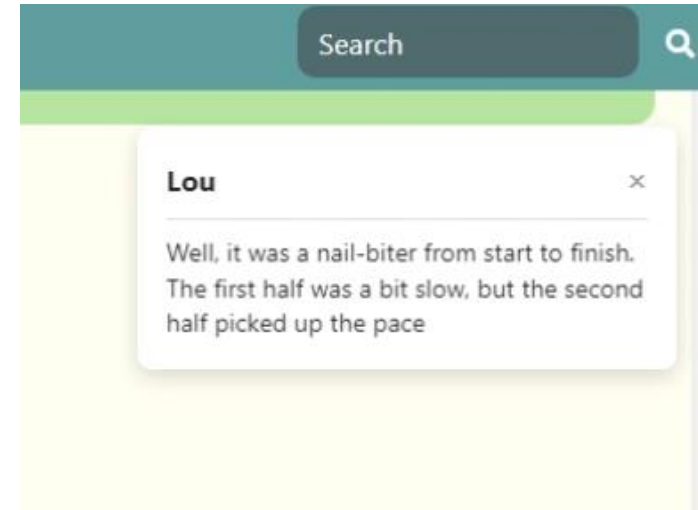
Toast Component

- Visualizzazione di brevi notifiche

```
@Injectable()
export class ToastService {
  timer: any;
  status: BehaviorSubject<any> = new BehaviorSubject<any>(null);
  private readonly duration = 4000;
  constructor() { }

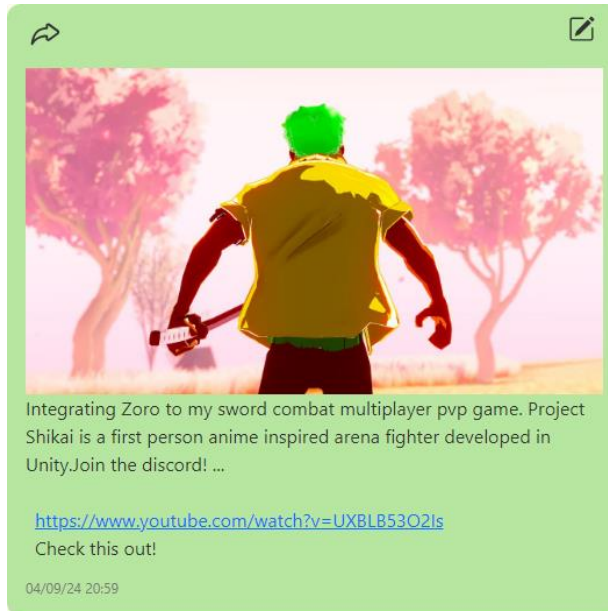
  showToast(msg: Message){
    if(this.timer){
      clearTimeout(this.timer);
    }

    this.status.next(msg);
    this.timer = window.setTimeout(()=>{
      this.status.next(null);
    },this.duration)
  }
}
```



Link Preview Component

- Visualizzazione preview di un URL



```
export class LinkPreviewService {  
  ⚡ constructor(private http: HttpClient) { }  
  
  private getLinkPreview(url: string): Observable<LinkPreview> {  
    return this.http.get(url, { responseType: 'text' })  
      .pipe(  
        map((html: string) => this.extractOpenGraphData(html, url)),  
        catchError(() => EMPTY)  
      );  
  }  
  
  private extractOpenGraphData(html: string, url: string): LinkPreview {  
    const $ = cheerio.load(html);  
    const title = $('meta[property="og:title"]').attr('content');  
    const description = $('meta[property="og:description"]').attr('content');  
    const image = $('meta[property="og:image"]').attr('content');  
    return new LinkPreview(title, description, image, url);  
  }  
  
  fetchLinkPreview(messageBody: string): Observable<LinkPreview> | undefined {  
    const link = this.extractLink(messageBody);  
    if (link) {  
      return this.getLinkPreview(link);  
    }  
    return undefined;  
  }  
  
  private extractLink(message: string): string | null {  
    const urlRegex = /(https?:\/\/\/[^\s]+)/g;  
    const matches = message.match(urlRegex);  
    return matches && matches.length > 0 ? matches[0] : null;  
  }  
}
```