Reddit Clone - Arquitectura Frontend Angular

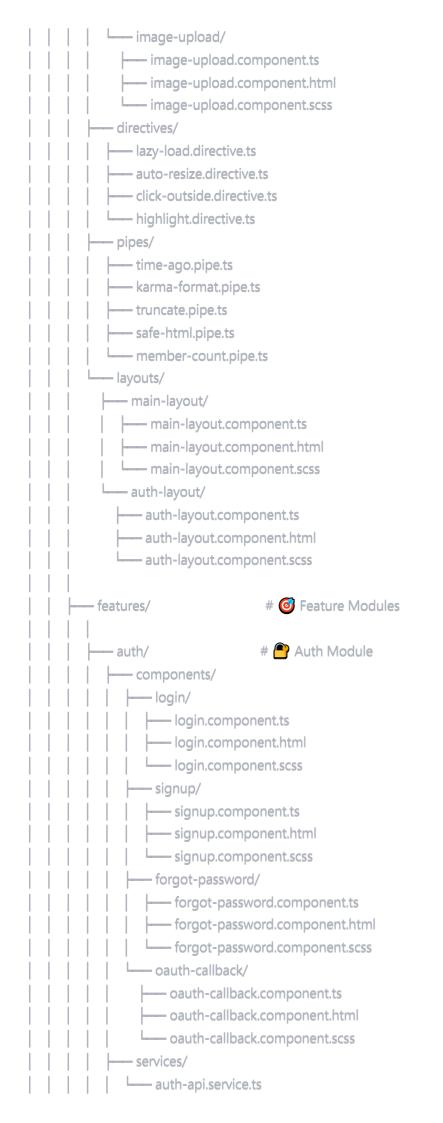
Tecnologías Frontend

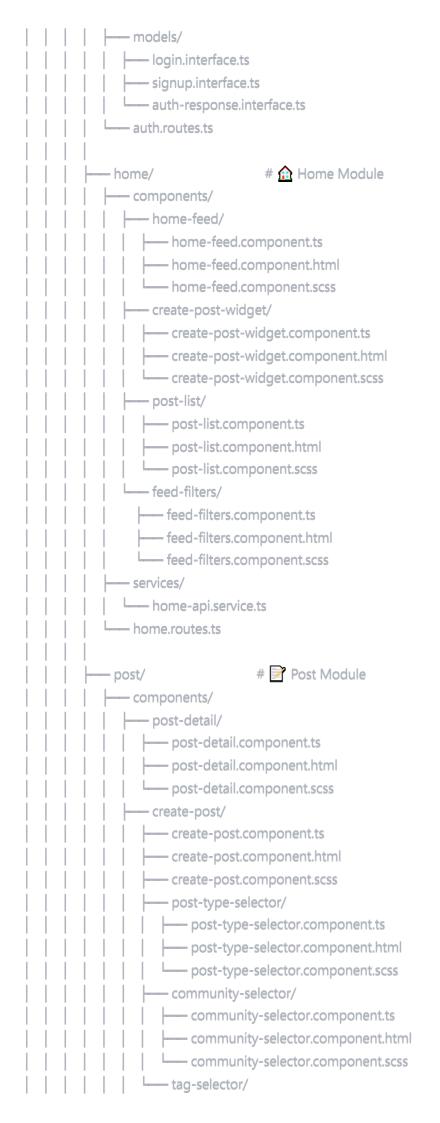
- Angular 17+ (Standalone Components + Signals)
- Angular Material (UI Components)
- Tailwind CSS (Utility-first CSS)
- NgRx (State Management)
- **RxJS** (Reactive Programming)
- **TypeScript** (Strict mode)

Estructura de Proyecto Angular

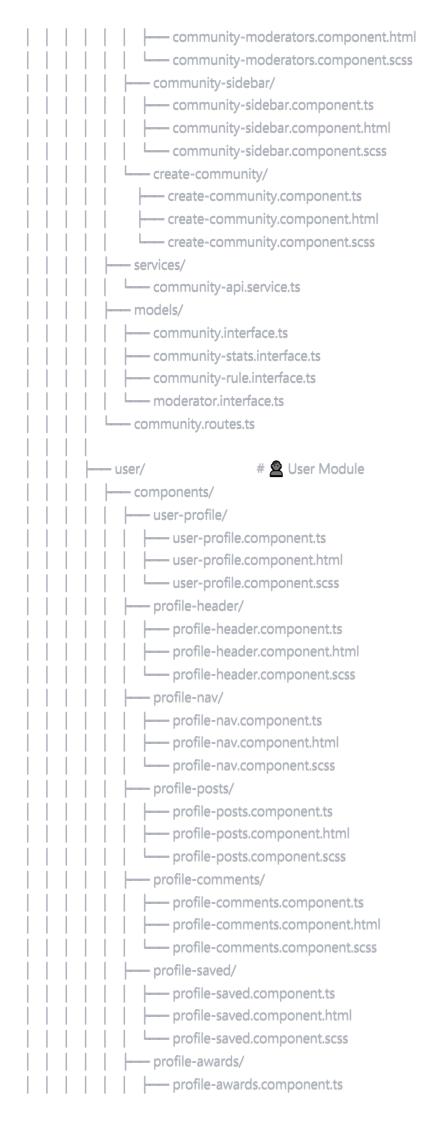


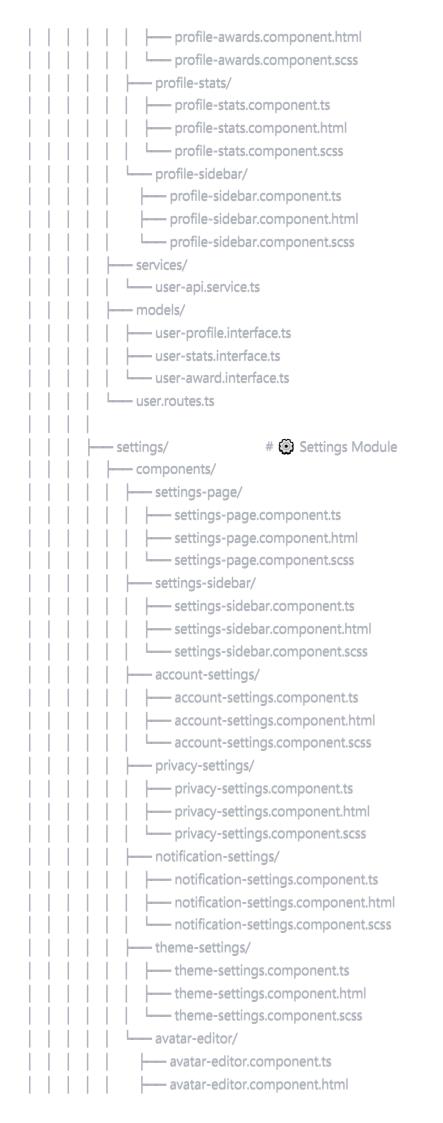
profile-dropdown.component.ts
profile-dropdown.component.html
profile-dropdown.component.scss
sidebar.component.html
sidebar.component.scss
community-list/
community-list.component.ts
community-list.component.html
community-list.component.scss
trending-topics/
trending-topics.component.ts
trending-topics.component.html
trending-topics.component.scss
post-card/
post-card.component.ts
post-card.component.html
post-card.component.scss
L vote-buttons/
vote-buttons.component.ts
vote-buttons.component.html
vote-buttons.component.scss
comment-tree/
comment-tree.component.ts
comment-tree.component.html
comment-tree.component.scss
comment-item/
comment-item.component.ts
comment-item.component.html
comment-item.component.scss
loading-spinner/
loading-spinner.component.ts
loading-spinner.component.html
loading-spinner.component.scss
error-message/
error-message.component.ts
error-message.component.html
error-message.component.scss
infinite-scroll/
infinite-scroll.component.ts
infinite-scroll.component.html
infinite-scroll.component.scss
rich-text-editor/
rich-text-editor.component.ts
rich-text-editor.component.html
rich-text-editor.component.scss

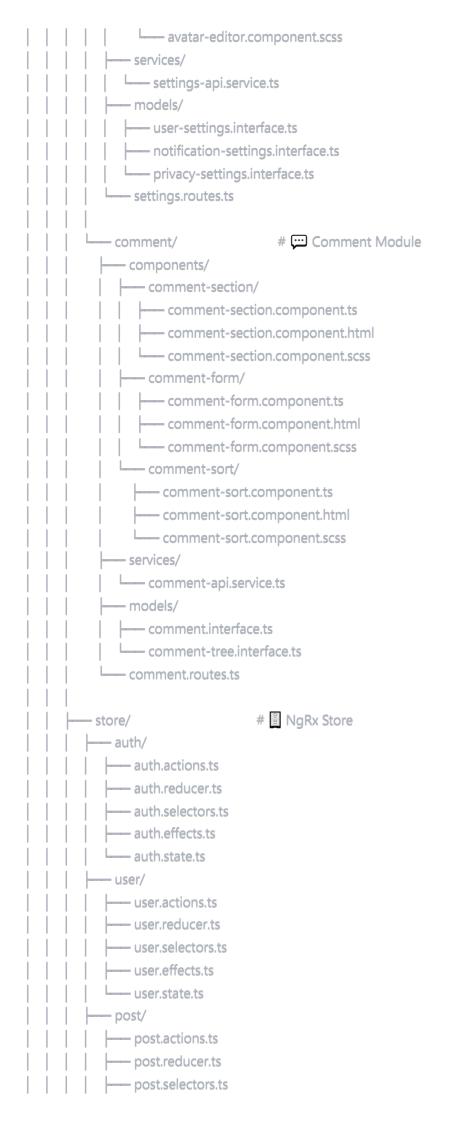












post.effects.ts	
post.state.ts	
— community/	
community.actions.ts	
community.reducer.ts	
community.selectors.ts	
community.state.ts	
— comment/	
— comment.actions.ts	
— comment.reducer.ts	
— comment.selectors.ts	
comment.effects.ts	
comment.state.ts	
ui.actions.ts	
— ui.reducer.ts	
ui.selectors.ts	
ui.state.ts	
app.reducer.ts	
app.state.ts	
environments/	
environment.ts	
environment.prod.ts	
environment.staging.ts	
assets/	
— icons/	
— images/	
fonts/	
scss/	
mixins.scss	
main.scss	
styles.scss	
— main.ts	
index.html	
angular.json	
— package.json	
— tailwind.config.js	
—— tsconfig.json	

Routing Structure

Main Routes (app.routes.ts)

```
typescript
export const routes: Routes = [
  path: ",
  component: MainLayoutComponent,
  children: [
   { path: ", redirectTo: '/home', pathMatch: 'full' },
   { path: 'home', loadChildren: () => import('./features/home/home.routes') },
   { path: 'r/:communityName', loadChildren: () => import('./features/community/community.routes') },
   { path: 'user/:username', loadChildren: () => import('./features/user.routes') },
   { path: 'post/:id', loadChildren: () => import('./features/post/post.routes') },
   { path: 'submit', loadChildren: () => import('./features/post/post.routes') },
   { path: 'settings', loadChildren: () => import('./features/settings/settings.routes'), canActivate: [AuthGuard] },
 },
  path: 'auth',
  component: AuthLayoutComponent,
  children: [
   { path: ", loadChildren: () => import('./features/auth/auth.routes') }
  ]
 { path: '**', redirectTo: '/home' }
];
```

Community Routes (community.routes.ts)

```
typescript
```

User Profile Routes (user.routes.ts)

Settings Routes (settings.routes.ts)

```
typescript
```

```
export const SETTINGS_ROUTES: Routes = [
  path: ",
  component: SettingsPageComponent,
  children: [
   { path: ", redirectTo: 'account', pathMatch: 'full' },
   { path: 'account', component: AccountSettingsComponent },
   { path: 'privacy', component: PrivacySettingsComponent },
   { path: 'notifications', component: NotificationSettingsComponent },
   { path: 'theme', component: ThemeSettingsComponent }
];
```

Component Mapping to SVG Designs

1. Home Feed (home-feed.component)

SVG Plantilla: [reddit_home_design]

```
typescript
@Component({
 selector: 'app-home-feed',
 standalone: true,
 imports: [CommonModule, PostListComponent, CreatePostWidgetComponent, SidebarComponent],
 template: `
  <div class="flex min-h-screen bg-gray-900">
   <!-- Sidebar Left -->
   <app-sidebar class="w-64 fixed left-0 top-16"></app-sidebar>
   <!-- Main Content -->
    <main class="flex-1 ml-64 mr-80 p-4">
     <app-create-post-widget class="mb-4"></app-create-post-widget>
     <app-post-list [posts]="posts$ | async"> </app-post-list>
    </main>
   <!-- Sidebar Right -->
   <aside class="w-80 fixed right-0 top-16 p-4">
     <app-trending-topics></app-trending-topics>
   </aside>
  </div>
})
```

2. Post Detail (post-detail.component)

SVG Plantilla: (reddit_post_detail)

```
typescript
@Component({
 selector: 'app-post-detail',
 standalone: true,
 imports: [CommonModule, PostContentComponent, CommentSectionComponent, VoteButtonsComponent],
 template: `
  <div class="flex min-h-screen bg-gray-900">
   <main class="flex-1 max-w-4xl mx-auto p-4">
    <!-- Breadcrumb -->
    <nav class="mb-4 text-gray-400">
     <a [routerLink]="['/r', post.communityName]">r/{{post.communityName}}</a> > {{post.title}}
    </nav>
    <!-- Post Content -->
    <article class="bg-gray-800 rounded-lg p-6 mb-6">
     <div class="flex">
      <app-vote-buttons [targetId]="post.id" [voteCount]="post.voteCount"></app-vote-buttons>
      <app-post-content [post]="post" [showFullContent]="true"> </app-post-content>
     </div>
    </article>
    <!-- Comments -->
    <app-comment-section [postId]="post.id"></app-comment-section>
   </main>
   <!-- Sidebar -->
   <aside class="w-80 p-4">
    <app-community-sidebar [communityName]="post.communityName"> </app-community-sidebar>
   </aside>
  </div>
```

3. Create Post (create-post.component)

SVG Plantilla: (reddit_create_post)

})



```
@Component({
 selector: 'app-create-post',
standalone: true,
imports: [CommonModule, ReactiveFormsModule, RichTextEditorComponent, CommunitySelector],
template: `
  <div class="min-h-screen bg-gray-900">
   <header class="bg-gray-900 border-b border-gray-700 p-4">
    <h1 class="text-2xl font-bold text-white">Create a post</h1>
   </header>
   <div class="flex max-w-6xl mx-auto p-4 gap-6">
    <main class="flex-1">
     <form [formGroup]="postForm" (ngSubmit)="onSubmit()" class="bg-gray-800 rounded-lg p-6">
      <!-- Community Selection -->
      <app-community-selector formControlName="communityId"></app-community-selector>
      <!-- Post Type Tabs -->
      <div class="flex gap-2 my-4">
        <button type="button"
            [class.bg-blue-600]="postType === 'text'"
            class="px-4 py-2 rounded-lg border"> Post</button>
        <button type="button"
            [class.bg-blue-600]="postType === 'image'"
            class="px-4 py-2 rounded-lg border"> ■ Image</button>
        <button type="button"
            [class.bg-blue-600]="postType === 'link'"
            class="px-4 py-2 rounded-lg border">  Link</button>
      </div>
      <!-- Title Input -->
      <input formControlName="title"
          placeholder="Title"
          class="w-full p-3 bg-gray-700 rounded-lg mb-4">
      <!-- Content Editor -->
      <app-rich-text-editor formControlName="content"></app-rich-text-editor>
      <!-- Tags -->
      <app-tag-selector formControlName="tags"></app-tag-selector>
```

<button type="button" class="px-6 py-2 border border-gray-600 rounded-lg">Save Draft</button>

<button type="submit" class="px-6 py-2 bg-blue-600 text-white rounded-lg">Post</button>

<!-- Submit -->

</div>

<div class="flex justify-end gap-4 mt-6">

4. Login Page (login.component)

SVG Plantilla: reddit_login_page



```
@Component({
 selector: 'app-login',
standalone: true,
imports: [CommonModule, ReactiveFormsModule],
template: `
  <div class="min-h-screen bg-black flex">
   <!-- Left side with features -->
   <div class="flex-1 flex flex-col justify-center items-center p-8">
    <h1 class="text-4xl font-bold text-orange-500 mb-4">Join the conversation</h1>
    Connect with communities around your interests
    <div class="space-y-4 w-full max-w-md">
     <div class="bg-gray-800 rounded-lg p-4 flex items-center">
      <div class="w-8 h-8 bg-green-500 rounded-full mr-3"> </div>
      <div>
       Share your knowledge
       Post and comment in communities
      </div>
     </div>
     <!-- More features... -->
    </div>
   </div>
   <!-- Right side with login form -->
   <div class="flex items-center justify-center p-8">
    <div class="w-full max-w-md bg-gray-800 rounded-2xl p-8">
     <div class="text-center mb-6">
      <div class="w-16 h-16 bg-orange-500 rounded-full mx-auto mb-4 flex items-center justify-center">
       <span class="text-white text-2xl font-bold">R</span>
      </div>
      <h2 class="text-2xl font-bold text-white">Welcome back</h2>
      Sign in to your account
     </div>
     <form [formGroup]="loginForm" (ngSubmit)="onSubmit()">
      <div class="space-y-4">
       <input formControlName="email"</pre>
           type="email"
           placeholder="Email or Username"
           class="w-full p-3 bg-gray-700 rounded-lg text-white">
       <input formControlName="password"
           type="password"
           placeholder="Password"
           class="w-full p-3 bg-gray-700 rounded-lg text-white">
```

```
<div class="flex items-center justify-between">
      <label class="flex items-center text-gray-300">
       <input type="checkbox" class="mr-2">
       Remember me
      </label>
      <a href="#" class="text-blue-400 text-sm">Forgot password?</a>
     </div>
     <button type="submit" class="w-full bg-blue-600 text-white py-3 rounded-lg font-medium">
     Log In
     </button>
     <div class="text-center">
      <span class="text-gray-400">OR</span>
     </div>
     <button type="button" class="w-full border border-gray-600 text-white py-3 rounded-lg flex items-center just
      <span class="mr-2">G</span> Continue with Google
     </button>
    </div>
   </form>
   Don't have an account?
    <a routerLink="/auth/signup" class="text-blue-400 font-medium">Sign Up</a>
   </div>
 </div>
</div>
```

Core Services

Auth Service

})

```
@Injectable({ providedIn: 'root' })
export class AuthService {
 private readonly API_URL = environment.apiUrl;
 private currentUserSubject = new BehaviorSubject < User | null > (null);
 public currentUser$ = this.currentUserSubject.asObservable();
 constructor(private http: HttpClient, private router: Router) {
  // Check for stored user on app init
  const storedUser = localStorage.getItem('currentUser');
  if (storedUser) {
   this.currentUserSubject.next(JSON.parse(storedUser));
 login(credentials: LoginRequest): Observable < AuthResponse > {
  return this.http.post<AuthResponse>(`${this.API_URL}/auth/login`, credentials)
   .pipe(
    tap(response => {
     localStorage.setItem('token', response.token);
     localStorage.setItem('currentUser', JSON.stringify(response.user));
      this.currentUserSubject.next(response.user);
    })
   );
 }
 signup(userData: SignupRequest): Observable < AuthResponse > {
  return this.http.post<AuthResponse>(`${this.API_URL}/auth/signup`, userData);
 logout(): void {
  localStorage.removeItem('token');
  localStorage.removeItem('currentUser');
  this.currentUserSubject.next(null);
  this.router.navigate(['/auth/login']);
 isAuthenticated(): boolean {
  return !!localStorage.getItem('token');
 }
 getCurrentUser(): User | null {
  return this.currentUserSubject.value;
```

API Service



```
@Injectable({ providedIn: 'root' })
export class ApiService {
 private readonly API_URL = environment.apiUrl;
 constructor(private http: HttpClient) {}
// Generic HTTP methods
 get<T>(endpoint: string, params?: HttpParams): Observable<T> {
  return this.http.get<T>(`${this.API_URL}${endpoint}`, { params });
 post<T>(endpoint: string, body: any): Observable<T> {
  return this.http.post<T>(`${this.API_URL}${endpoint}`, body);
 put<T>(endpoint: string, body: any): Observable<T> {
  return this.http.put<T>(`${this.API_URL}${endpoint}`, body);
 delete < T > (endpoint: string): Observable < T > {
  return this.http.delete < T > (`${this.API_URL}${endpoint}`);
}
// Specific API methods
 getPosts(page: number = 0, size: number = 10): Observable < PagedResponse < Post >> {
  const params = new HttpParams()
   .set('page', page.toString())
   .set('size', size.toString());
  return this.get < Paged Response < Post >> ('/posts', params);
 getPost(id: string): Observable < Post > {
  return this.get<Post>(`/posts/${id}`);
 createPost(post: CreatePostRequest): Observable < Post > {
  return this.post < Post > ('/posts', post);
 }
// Community methods
 getCommunity(name: string): Observable < Community > {
  return this.get < Community > (`/communities/${name}`);
}
 getCommunityPosts(name: string, page: number = 0): Observable < PagedResponse < Post >> {
  const params = new HttpParams()
```

```
.set('page', page.toString())
  .set('size', '10');
 return this.get < PagedResponse < Post >> (`/communities/${name}/posts`, params);
// User methods
getUserProfile(username: string): Observable < UserProfile > {
 return this.get < UserProfile > (`/users/${username}`);
getUserPosts(username: string, page: number = 0): Observable < PagedResponse < Post >> {
 const params = new HttpParams()
  .set('page', page.toString())
  .set('size', '10');
 return this.get < Paged Response < Post > (`/users/${username}/posts`, params);
// Vote methods
vote(targetId: string, targetType: 'post' | 'comment', voteType: 'upvote' | 'downvote'): Observable < VoteResponse > {
 return this.post<VoteResponse>('/votes', { targetId, targetType, voteType });
}
// Comment methods
getComments(postId: string): Observable < Comment[] > {
 return this.get < Comment[] > ('/posts/${postId}/comments');
createComment(comment: CreateCommentRequest): Observable < Comment > {
 return this.post < Comment > ('/comments', comment);
```



Tailwind Configuration

tailwind.config.js

```
module.exports = {
 content: ['./src/**/*.{html,ts}'],
 darkMode: 'class',
 theme: {
  extend: {
   colors: {
     reddit: {
      orange: '#ff4500',
      blue: '#0079d3',
      dark: '#1a1a1b',
      gray: {
       100: '#f8f9fa',
       200: '#edeff1',
       300: '#d7dadc',
       400: '#878a8c',
       500: '#818384',
       600: '#343536',
       700: '#272729',
       800: '#1a1a1b',
       900: '#030303'
   fontFamily: {
     sans: ['Inter', 'system-ui', 'sans-serif']
   spacing: {
    '18': '4.5rem',
     '88': '22rem'
 plugins: [
  require('@tailwindcss/forms'),
  require('@tailwindcss/typography')
```

Responsive Design Strategy

Breakpoints

```
// Mobile-first approach
$mobile: 640px; //sm
$tablet: 768px; // md
$desktop: 1024px; // lg
$large: 1280px; //xl
// Layout adjustments
@media (max-width: $tablet) {
 .sidebar { display: none; }
 .main-content { margin: 0; width: 100%; }
 .mobile-nav { display: block; }
@media (max-width: $mobile) {
 .post-card { padding: 1rem; }
 .header { height: 3rem; }
 .vote-buttons {
  flex-direction: row;
  gap: 0.5rem;
```

State Management with NgRx

App State Structure

```
export interface AppState {
 auth: AuthState;
 user: UserState:
 posts: PostState;
 communities: CommunityState;
 comments: CommentState;
 ui: UIState:
export interface AuthState {
 user: User | null;
 token: string | null;
 isAuthenticated: boolean;
 loading: boolean;
 error: string | null;
export interface PostState {
 posts: Post[];
 currentPost: Post | null;
 loading: boolean;
 error: string | null;
 pagination: {
  page: number;
  totalPages: number;
  hasMore: boolean;
};
```



Testing Strategy

Component Testing Example

```
typescript
```

```
describe('PostCardComponent', () => {
 let component: PostCardComponent;
 let fixture: ComponentFixture < PostCardComponent >;
 beforeEach(async () => {
  await TestBed.configureTestingModule({
   imports: [PostCardComponent, HttpClientTestingModule],
   providers: [provideMockStore()]
  }).compileComponents();
  fixture = TestBed.createComponent(PostCardComponent);
  component = fixture.componentInstance;
 });
 it('should display post content correctly', () => {
  const mockPost: Post = {
   id: '1',
   title: 'Test Post',
   content: 'Test Content',
   voteCount: 10,
   // ... more properties
  };
  component.post = mockPost;
  fixture.detectChanges();
  expect(fixture.nativeElement.querySelector('.post-title').textContent).toBe('Test Post');
});
});
```

Development Phases

Fase 1: Core Structure (Semanas 1-2)

- 1. **Project setup** con Angular 17, Tailwind, NgRx
- 2. Auth layout + Login/Signup components
- 3. Main layout + Header + Sidebar
- 4. **Basic routing** y guards

Fase 2: Home & Posts (Semanas 3-4)

- 5. Home feed component
- 6. **Post card** component con voting

- 7. Post detail page
- 8. Create post form

Fase 3: Communities (Semanas 5-6)

- 9. **Community page** con todas las pestañas
- 10. Community header y sidebar
- 11. Community navigation

Fase 4: User Profiles (Semanas 7-8)

- 12. **User profile** con todas las pestañas
- 13. **Profile dropdown** component
- 14. **Settings pages** completas

Fase 5: Comments & Advanced (Semanas 9-10)

- 15. Comment system anidado
- 16. Real-time features con WebSocket
- 17. Infinite scroll y lazy loading
- 18. **Mobile responsive** optimization

Fase 6: Polish & Testing (Semanas 11-12)

- 19. PWA features
- 20. Testing completo
- 21. Performance optimization
- 22. Documentation

¿Te parece bien esta arquitectura de Angular? ¿Quieres que detalle algún componente específico o empecemos con la implementación de alguna fase?