



Chat Layout Module - Tài liệu chi tiết

Mô tả chi tiết giao diện Chat bao gồm LeftBar, CenterContent, RightBar, Menu và Socket

Cập nhật: 2026-01-19



Mục Lục

1. [Tổng quan](#)
2. [Cấu trúc thư mục](#)
3. [Layout Architecture](#)
4. [ChatWarper \(Main Entry\)](#)
5. [Menu Sidebar](#)
6. [LeftBar - Danh sách hội thoại](#)
7. [CenterContent - Khung chat](#)
8. [RightBar - Widget Panel](#)
9. [Socket & Real-time](#)
10. [Stores & State Management](#)

1. Tổng quan

1.1 Mục đích

Chat Layout là **màn hình chính** của ứng dụng, cung cấp:

- **Xem danh sách hội thoại** từ nhiều nền tảng
- **Nhắn tin** với khách hàng
- **Lọc** theo nhiều tiêu chí (nhân, nhân viên, thời gian, etc.)
- **Xem thông tin** chi tiết khách hàng
- **Sử dụng Widget** bên thứ 3
- **Real-time updates** qua WebSocket

1.2 Tech Stack

Công nghệ	Mục đích
Vue 3 Composition API	Framework
Pinia	State management
Socket.io	Real-time communication
Splitpanes	Resizable panels
vue-i18n	Đa ngôn ngữ
date-fns	Xử lý thời gian
lodash	Utility functions

2. Cấu trúc thư mục

```
src/views/ChatWarper/
├── ChatWarper.vue          # Main entry + Socket handler
├── Layout.vue               # Splitpanes layout (resizable)
└── Menu.vue                 # Sidebar menu với filters

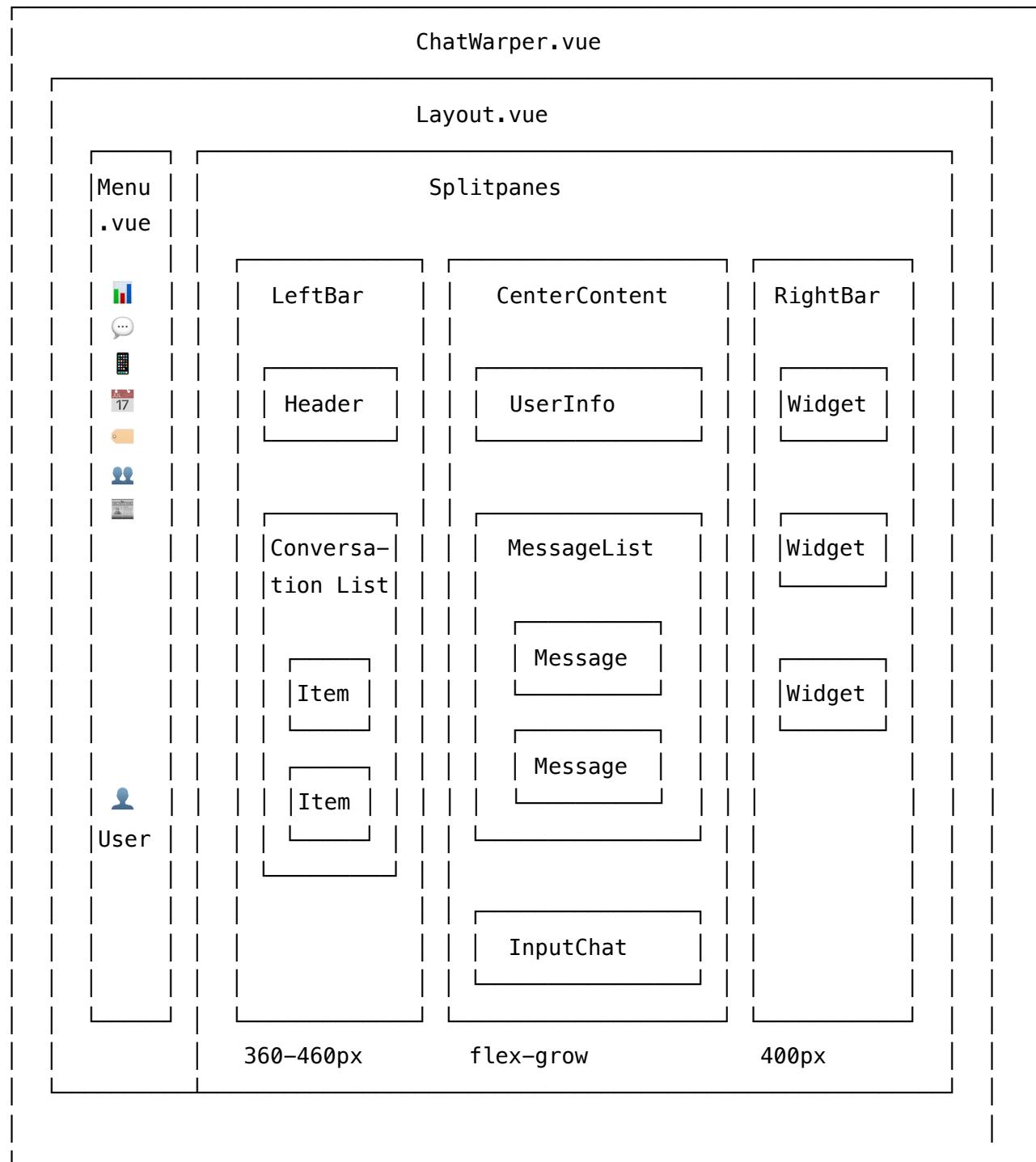
├── Menu/
│   ├── NavItem.vue          # Menu components
│   ├── FilterInteract.vue   # Item navigation
│   ├── FilterMessage.vue    # Lọc theo tương tác
│   ├── FilterPhone.vue      # Lọc theo tin nhắn
│   ├── FilterDate.vue       # Lọc theo SĐT
│   ├── FilterTag.vue        # Lọc theo ngày
│   ├── FilterNotTag.vue     # Lọc theo nhãn
│   ├── FilterStaff.vue      # Lọc không có nhãn
│   ├── FilterPost.vue       # Lọc theo nhân viên
│   ├── Attach.vue           # Lọc theo bài post
│   └── User.vue              # Menu đính kèm
                                # User dropdown

├── Chat/
│   ├── LeftBar.vue          # Chat area components
│   │   # Wrapper cột trái
│   ├── LeftBar/
│   │   ├── Header.vue        # Header với tabs + search
│   │   ├── Conversation.vue  # Danh sách hội thoại
│   │   └── Conversation/
│   │       ├── Item.vue       # Item hội thoại
│   │       ├── Avatar.vue     ...
│   │       └── Badge.vue
│   ...
│
│   ├── CenterContent.vue    # Wrapper cột giữa
│   ├── CenterContent/
│   │   ├── UserInfo.vue      # Thông tin user đang chat
│   │   ├── UserInfo/
│   │   │   ├── Info.vue       # Chi tiết thông tin
│   │   │   ├── Note.vue        # Ghi chú
│   │   │   └── Actions.vue    # Actions bar
│   │   ...
│
│   └── MessageList.vue      # Danh sách tin nhắn
        # Wrapper cột phải
        # MessageList/
```

```
|   |   |   └── MessageItem/  # Các loại tin nhắn
|   |   ├── TimeSplit.vue  # Phân cách thời gian
|   |   ├── UnReadAlert.vue # Đánh dấu chưa đọc
|   |   ├── ClientRead.vue # Avatar client đã đọc
|   |   ├── StaffRead.vue  # Avatar staff đã đọc
|   |   └── ...
|
|   └── InputChat.vue      # Wrapper input
|   └── InputChat/
|       ├── MainInput.vue  # Input chính
|       ├── ListLabel.vue  # Nhãn của hội thoại
|       ├── PreviewAttachment.vue # Preview file
|       ├── ScrollToBottomBtn.vue # Nút scroll xuống
|       ├── QuickReply.vue # Trả lời nhanh
|       └── ...
|
└── StaffReadModal.vue # Modal xem staff đã đọc
|
└── RightBar.vue        # Wrapper cột phải
└── RightBar/
    ├── WidgetSorting.vue # Sắp xếp widget
    ├── PostRightBar.vue  # Right bar cho Post
    └── PostAnalytic/     # Phân tích bài post
```

3. Layout Architecture

3.1 Overall Structure



3.2 Layout.vue (Splitpanes)

Mục đích: Resizable panels cho LeftBar và Content

```

/** Cấu hình */
const MIN = 360 // Độ rộng tối thiểu cột trái (px)
const MAX = 460 // Độ rộng tối đa cột trái (px)

/** State */
const ready = ref(false) // Render flag
const width = ref(0) // Container width
const size = ref(0) // Left column size (%)

/** Computed */
const min = computed(() => round((MIN / width.value) * 100))
const max = computed(() => round((MAX / width.value) * 100))

/** Restore từ localStorage */
onBeforeMount(() => {
  const LOCAL_WIDTH = $localStorage.getItem('conversation_width')
  size.value = LOCAL_WIDTH
})

/** Lưu khi resize */
function onResized({ prevPane }) {
  if (!prevPane?.size) return
  size.value = prevPane.size
  $localStorage.setItem('conversation_width', round(size.value))
}

```

4. ChatWarper (Main Entry)

4.1 ChatWarper.vue

Mục đích: Entry point của Chat, xử lý socket và khởi tạo data

Template Structure:

```
<template>
  <div
    @dragover.prevent
    @drop="onDropFile"
    id="router__chat">
    >
    <!-- Hot Alerts -->
    <HotAlert :codes="['ALMOST_REACH_QUOTA_AI', 'LOCK_FEATURE', ...]" />

    <!-- Menu sidebar -->
    <menu />

    <!-- Main content -->
    <template v-if="view === 'CHAT'">
      <Layout>
        <template #left><LeftBar /></template>
        <template #right>
          <CenterContent />
          <RightBar />
        </template>
      </Layout>
    </template>

    <!-- Other views -->
    <template v-else-if="view === 'SEARCH'">
      <SearchPanel />
    </template>

    <template v-else-if="view === 'FRIEND_REQUEST'">
      <FriendRequest />
    </template>
  </div>
</template>
```

4.2 Lifecycle

```
onMounted(async () => {
  // Khởi tạo extension logic (nếu dùng trong Chrome extension)
  initExtensionLogic()

  // Kết nối socket
  $socket.connect()

  // Lấy thông tin page và widgets
  await $main.getPageInfoToChat()

  // Kiểm tra quyền thông báo
  checkAllowNoti()

  // Lắng nghe focus/blur
  window.addEventListener('focus', checkFocusChatTab)
  window.addEventListener('blur', checkFocusChatTab)
})

onUnmounted(() => {
  // Đóng socket
  $socket.close()

  // Huỷ lắng nghe
  window.removeEventListener('focus', checkFocusChatTab)
  window.removeEventListener('blur', checkFocusChatTab)
})
```

4.3 Main Class

```
class Main {
  constructor(
    private readonly API_PAGE = container.resolve(N4ServiceAppPage),
    private readonly API_APP = container.resolve(N5AppV1AppApp)
  ) {}

  /** Lấy thông tin page để chat */
  @loading(commonStore)
  @error(container.resolve(CustomToast))
  async getPageInfoToChat() {
    // Kiểm tra đã vượt giới hạn gói chưa
    checkOverLimit()

    // Lấy danh sách page
    const pages = await this.API_PAGE.getPages()
    pageStore.setPages(pages)

    // Đánh dấu org có Zalo hay không
    this.markOrgHaveZalo()
  }

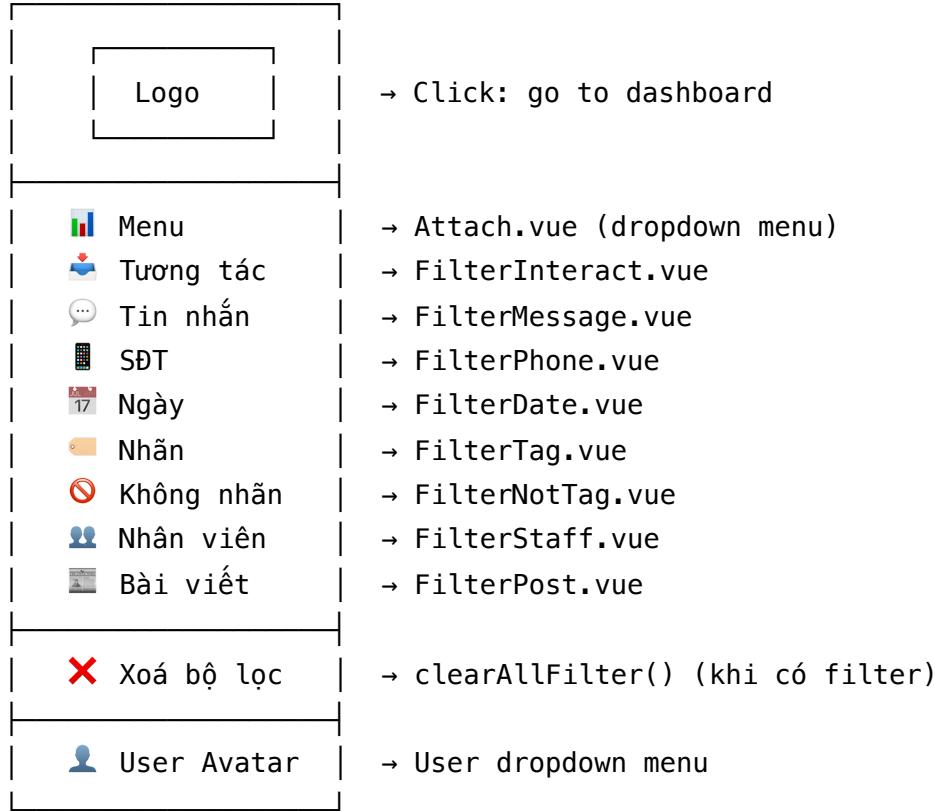
  /** Đánh dấu tổ chức có page Zalo */
  markOrgHaveZalo() {
    const HAS_ZALO = pageStore.active_page_list?.some(
      page => page?.page?.type === 'ZALO'
    )
    orgStore.has_zalo = HAS_ZALO
  }
}
```

5. Menu Sidebar

5.1 Menu.vue

Mục đích: Sidebar chứa các bộ lọc hội thoại

UI Structure:



5.2 Filter Components

Component	Icon	Filter Key	Description
FilterInteract.vue	InboxIcon	is_read , is_response	Đã đọc, Đã phản hồi
FilterMessage.vue	ChatDotIcon	unread_message , not_response_client	Chưa đọc, Chưa phản hồi
FilterPhone.vue	PhoneIcon	have_phone	Có/Không có SĐT
FilterDate.vue	DateIcon	time_range	Khoảng thời gian
FilterTag.vue	TagIcon	label_id	Lọc theo nhãn
FilterNotTag.vue	TagNotIcon	not_label_id	Trừ các nhãn
FilterStaff.vue	UserIcon	staff_id	Lọc theo nhân viên
FilterPost.vue	NewspaperIcon	post_id	Lọc bài viết FB

5.3 Keyboard Shortcuts

```
watch(
  () => commonStore.keyboard_shortcut,
  (new_value) => {
    // Xoá bộ lọc
    if (new_value === 'clear_all_filter' && isFilterActive()) {
      $main.clearAllFilter()
    }

    // Map shortcut -> filter ref
    const MAP: Record<string, any> = {
      'filter_interact': filter_interact,
      'filter_message': filter_message,
      'filter_phone': filter_phone,
      // ...
    }

    // Dispatch dblclick event để mở filter
    const FILTER = MAP[new_value]
    const DBL_CLICK_EVENT = new MouseEvent('dblclick', {....})
    BUTTON_NEW?.dispatchEvent(DBL_CLICK_EVENT)
  }
)
```

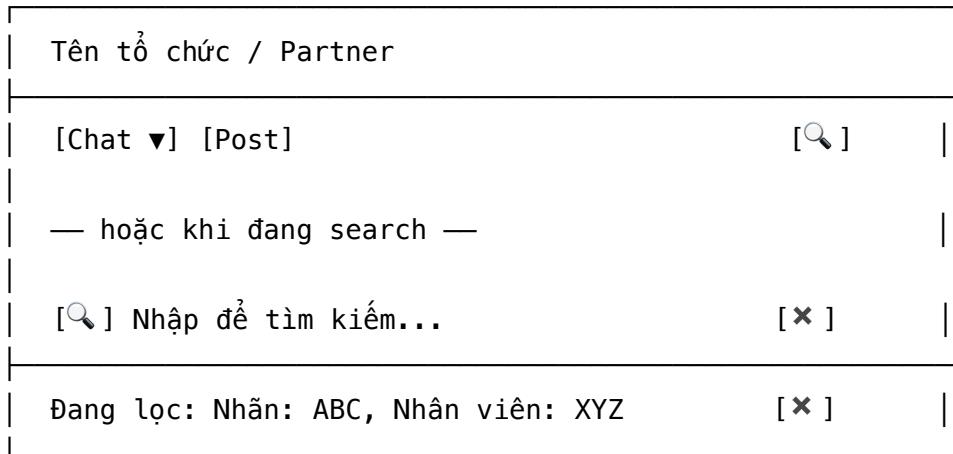
6. LeftBar - Danh sách hội thoại

6.1 LeftBar.vue

```
<template>
  <div
    id="chat_left-bar"
    class="bg-white h-full rounded-xl flex flex-col">
    <header />
    <Conversation />
  </div>
</template>
```

6.2 Header.vue

UI Structure:



Tabs:

Tab	Value	Description
Chat	CHAT	Hội thoại messenger/zalo
Post	POST	Comment Facebook posts

State:

```

/** Tab đang active */
type IActiveTab = 'CHAT' | 'POST'

/** Tìm kiếm (debounced) */
const search_conversation = ref<string>()
const onSearchConversation = debounce(value => {
  conversationStore.option_filter_page_data.search = value
}, 300)

/** Dữ liệu lọc hiển thị dạng text */
const filter = computed(() => {
  const RESULT: string[] = []

  // Lọc nhãn
  if (conversationStore.option_filter_page_data.label_id) {
    RESULT.push(`Nhãn: ${tags.value[id].title}`)
  }

  // Lọc nhân viên
  if (conversationStore.option_filter_page_data.staff_id?.length) {
    RESULT.push(`Nhân viên: ${staffNames.join(', ')}`)
  }

  // ... other filters

  return RESULT.join(', ')
})

```

6.3 Conversation.vue

Mục đích: Danh sách hội thoại với infinite scroll

Logic chính:

```
class Main {
    /** Lấy danh sách hội thoại */
    @loadingV2(conversationStore, 'is_loading')
    @error()
    async getConversation(is_first_time?: boolean, is_pick_first?: boolean) {
        // API call
        const { results } = await this.API_CONVERSATION.getConversations({
            page_ids: pageStore.selected_page_ids,
            ...conversationStore.option_filter_page_data,
            skip: is_first_time ? 0 : conversationStore.conversation_list.length,
            limit: 20,
        })

        // Append hoặc replace
        if (is_first_time) {
            conversationStore.conversation_list = results
        } else {
            conversationStore.conversation_list.push(...results)
        }

        // Auto select first
        if (is_pick_first) {
            this.selectDefaultConversation()
        }
    }

    /** Xử lý socket conversation */
    onRealtimeUpdateConversation({ detail }: CustomEvent) {
        const { conversation, message, event } = detail

        // Validate conversation theo filter
        if (!validateConversation(conversation, message)) return

        // Update hoặc thêm mới vào list
        const index = findIndex(
            conversationStore.conversation_list,
            c => c.fb_client_id === conversation.fb_client_id
        )

        if (index >= 0) {
            // Update existing
            conversationStore.conversation_list[index] = {
                ...conversationStore.conversation_list[index],
```

```

    ...conversation,
}
} else {
    // Thêm mới ở đầu
    conversationStore.conversation_list.unshift(conversation)
}
}

/** Load thêm khi scroll xuống */
loadMoreConversation($event: UIEvent) {
    const target = $event.target as HTMLElement
    const { scrollTop, scrollHeight, clientHeight } = target

    // Đến cuối list
    if (scrollTop + clientHeight >= scrollHeight - 100) {
        this.getConversation()
    }
}

/** Auto refresh khi focus lại sau thời gian dài */
autoRefreshPage() {
    if (differenceInHours(new Date(), last_focus_time) >= 3) {
        window.location.reload()
    }
}
}

```

Watchers:

```

// Load lại khi filter thay đổi
watch(
  () => option_filter_page_data,
  () => $main.loadConversationFirstTime(true, false, true)
)

// Load lại khi chuyển tab
watch(
  () => conversationStore.option_filter_page_data?.conversation_type,
  () => $main.loadConversationFirstTime(true, false, true)
)

// Load lại khi page list thay đổi
watch(
  () => pageStore.selected_page_list_info,
  () => $main.loadConversationFirstTime(true, true)
)

```

7. CenterContent - Khung chat

7.1 CenterContent.vue

```

<template>
  <div
    id="chat__center-content"
    class="h-full flex flex-col flex-grow"
  >
    <UserInfo />
    <MessageList />
    <InputChat :client_id="$route.query.user_id?.toString()" />
  </div>

  <!-- Modals -->
  <StaffReadModal />
  <ZaloPersonalModal ref="modal_zalo_personal_ref" />
</template>

```

7.2 UserInfo.vue

Hiển thị thông tin khách hàng đang chat:



7.3 MessageList.vue

Mục đích: Hiển thị danh sách tin nhắn

Layout đặc biệt: Sử dụng flex-col-reverse để scroll từ dưới lên

```
<div
  @scroll="onScrollMessage"
  :id="messageStore.list_message_id"
  class="flex flex-col h-full overflow-y-auto bg-[#0015810f]"
>
  <!-- Loading indicator -->
  <div
    v-if="is_loading"
    class="relative z-10"
  >
    <Loading />
  </div>

  <!-- Message list -->
  <div
    v-for="(message, index) of show_list_message"
    :key="message._id"
  >
    <!-- Time split -->
    <TimeSplit
      :before_message="show_list_message?.[index - 1]"
      :now_message="message"
    />

    <!-- Unread alert -->
    <UnReadAlert :index />

    <!-- Message item -->
    <div class="flex gap-1">
      <!-- Client avatar (for client messages) -->
      <ClientAvatar v-if="message.message_type === 'client'" />

      <!-- Message content -->
      <MessageItem :message="message" />

      <!-- Staff avatar (for page messages) -->
      <PageStaffAvatar v-if="message.message_type === 'page'" />

      <!-- Read receipts -->
      <ClientRead :time="message.time" />
    </div>

    <!-- Staff read indicators -->
```

```

<StaffRead :time="message.time" />
</div>

<!-- Sending messages (temp) -->
<div v-for="message of messageStore.send_message_list">
  <TempMessage :message="message" />
  <SendStatus :is_error="message.error" />
</div>
</div>

```

Message Types:

Type	Description	Component
client	Tin từ khách	ClientMessage
page	Tin từ page/staff	PageMessage
note	Ghi chú nội bộ	NoteMessage
activity	Hoạt động hệ thống	ActivityMessage
group	Group chat	GroupMessage
ad	Quảng cáo	AdMessage

Scroll Logic:

```

/** Scroll đến cuối */
function scrollToBottomLocal() {
  const element = document.getElementById(messageStore.list_message_id)
  if (!element) return
  element.scrollTop = element.scrollHeight
}

/** Load thêm tin nhắn khi scroll lên */
async function loadMoreMessage() {
  const element = document.getElementById(messageStore.list_message_id)
  if (!element) return

  // Đến đầu list (scroll lên)
  if (element.scrollTop <= 100) {
    await getMoreMessages()
  }
}

/** Xử lý socket tin nhắn mới */
function socketNewMessage({ detail }) {
  const { message, conversation } = detail

  // Thêm tin nhắn vào cuối list
  messageStore.message_list.push(message)

  // Scroll xuống nếu đang ở cuối
  if (isNearBottom()) {
    nextTick(() => scrollToBottomLocal())
  }
}

```

7.4 InputChat.vue

Wrapper cho input components:

```

<template>
  <div
    v-if="client_id"
    id="chat__input-chat"
    class="flex flex-col gap-2"
  >
    <!-- Nút scroll xuống -->
    <ScrollToBottomBtn />

    <!-- Reply comment (cho Post) -->
    <ReplyComment v-if="messageStore.reply_comment?.root_comment_id" />

    <!-- List labels -->
    <ListLabel v-else />

    <!-- Preview attachment -->
    <PreviewAttachment />

    <!-- Main input area -->
    <MainInput v-if="conversationType !== 'POST'" />
  </div>
</template>

```

MainInput.vue components:

- TextArea với emoji picker
- Attachment (file, image, video)
- Quick reply dropdown
- Send button

8. RightBar - Widget Panel

8.1 RightBar.vue

Mục đích: Hiển thị widgets bên phải

```
<template>
  <div
    id="chat__right-bar"
    class="w-[400px] h-full flex-shrink-0"
  >
    <!-- Post mode -->
    <PostRightBar v-if="conversation_type === 'POST'" />

    <!-- Chat mode -->
    <template v-else>
      <!-- AI Journey widget -->
      <AiJourney />

      <!-- Widget list -->
      <div v-for="widget of pageStore.widget_list">
        <div class="rounded-lg bg-white overflow-hidden">
          <!-- Widget header (toggle) -->
          <button
            @click="toggleWidget(widget)"
            class="w-full py-2.5 px-3"
          >
            {{ widget.snap_app?.name }}
            <ChevronDownIcon :class="{ '-rotate-90': !widget.is_show }" />
          </button>

          <!-- Widget content (iframe) -->
          <div
            v-if="widget.is_show"
            class="flex-grow"
          >
            <iframe
              :id=`widget-${widget._id}`
              :src="widget.url"
              class="w-full h-full"
              allow="microphone; camera; autoplay; speaker"
            />
          </div>
        </div>
      </div>
    </template>
    <!-- Widget sorting mode -->
    <WidgetSorting v-show="view === 'sorting'" />
  </template>
```

```

<!-- Dropdown menus -->
<Dropdown ref="widget_dropdown_ref" />
<Dropdown ref="change_mode_dropdown_ref" />
</div>
</template>

```

8.2 Widget Token Flow

```

/** Khởi tạo token cho widget */
async function getTokenOfWidget(
  new_val?: ConversationInfo,
  old_val?: ConversationInfo
) {
  // Không cần refresh nếu cùng page
  if (new_val?.page_id === old_val?.page_id) return

  // Gọi API lấy widget token
  const { token } = await API_APP.getWidgetToken({
    page_id: new_val?.page_id,
    client_id: new_val?.fb_client_id,
  })

  // Lưu token
  conversationStore.list_widget_token = { data: token }

  // Gửi token đến các iframe
  pageStore.widget_list?.forEach(widget => {
    sendEventToIframe(widget, { token })
  })
}

/** Gửi event đến iframe */
function sendEventToIframe(widget, payload) {
  const iframe = document.getElementById(`widget-${widget._id}`)
  if (!iframe) return

  iframe.contentWindow?.postMessage(payload, '*')
}

```

9. Socket & Real-time

9.1 Socket Connection

```
// ChatWarper.vue
onMounted(() => {
  $socket.connect()

  // Lắng nghe events
  $socket.on('new_message', handleSocketEvent)
  $socket.on('update_conversation', handleSocketEvent)
  $socket.on('update_message', handleSocketEvent)
})

onUnmounted(() => {
  $socket.close()
})
```

9.2 Socket Event Handler

```
function handleSocketEvent(socket_data: {
  conversation?: ConversationInfo
  message?: MessageInfo
  staff?: StaffSocket
  event?: SocketEvent
  update_message?: MessageInfo
  update_comment?: FacebookCommentPost
}) {
  const { conversation, message, event } = socket_data

  // Refresh thời gian hội thoại
  debounceRefreshConversationTime()

  // Validate theo filter
  if (!validateConversation(conversation, message)) return

  // Dispatch custom events để các component con xử lý
  window.dispatchEvent(
    new CustomEvent('chatbox_socket_conversation', {
      detail: socket_data,
    })
  )

  window.dispatchEvent(
    new CustomEvent('chatbox_socket_message', {
      detail: socket_data,
    })
  )

  // Trigger thông báo
  triggerAlert(conversation)
}
```

9.3 Validation Conversation

```
/** Kiểm tra hội thoại có thỏa mãn điều kiện lọc không */
function validateConversation(
    conversation?: ConversationInfo,
    message?: MessageInfo
): boolean {
    const filter = conversationStore.option_filter_page_data

    // Kiểm tra page được chọn
    if (!pageStore.isSelectedPage(conversation?.page_id)) return false

    // Kiểm tra conversation type (CHAT/POST)
    if (filter.conversation_type !== conversation?.conversation_type) return false

    // Kiểm tra label filter
    if (filter.label_id) {
        const hasLabel = conversation?.labels?.includes(filter.label_id)
        if (!hasLabel) return false
    }

    // Kiểm tra staff filter
    if (filter.staff_id?.length) {
        const hasStaff = filter.staff_id.includes(conversation?.assigned_staff_id)
        if (!hasStaff) return false
    }

    // ... other filters

    return true
}
```

9.4 Notifications

```
/** Gửi thông báo cho nhân viên */
function triggerAlert(conversation?: ConversationInfo) {
    // Check quyền thông báo
    if (!commonStore.is_allow_noti) return

    // Check không phải tin nhắn của mình
    if (conversation?.last_message?.staff_id === currentStaffId) return

    // Gửi thông báo
    pushWebNoti(conversation)
    ringBell()
}

/** Web notification */
function pushWebNoti(conversation?: ConversationInfo) {
    if (Notification.permission !== 'granted') return

    new Notification(conversation?.client_name || 'Tin nhắn mới', {
        body: conversation?.last_message?.text,
        icon: conversation?.client_avatar,
    })
}

/** Phát âm thanh */
function ringBell() {
    const audio = new Audio('/notification.mp3')
    audio.play()
}
```

10. Stores & State Management

10.1 Related Stores

Store	Purpose
useConversationStore	Danh sách hội thoại, hội thoại đang chọn, filter

Store	Purpose
useMessageStore	Danh sách tin nhắn, tin nhắn đang gửi
usePageStore	Danh sách page, page đang chọn
useChatbotUserStore	Thông tin user đang đăng nhập
useOrgStore	Thông tin tổ chức
useWidgetStore	Widget state

10.2 useConversationStore

```
{
  // State
  conversation_list: [],           // Danh sách hội thoại
  select_conversation: null,       // Hội thoại đang chọn
  option_filter_page_data: {       // Filter options
    conversation_type: 'CHAT',
    search: '',
    label_id: null,
    staff_id: [],
    time_range: null,
    // ...
  },
  count_conversation: { chat: 0, post: 0 },
  list_widget_token: null,

  // Getters
  isCurrentStaffAdmin(): boolean,

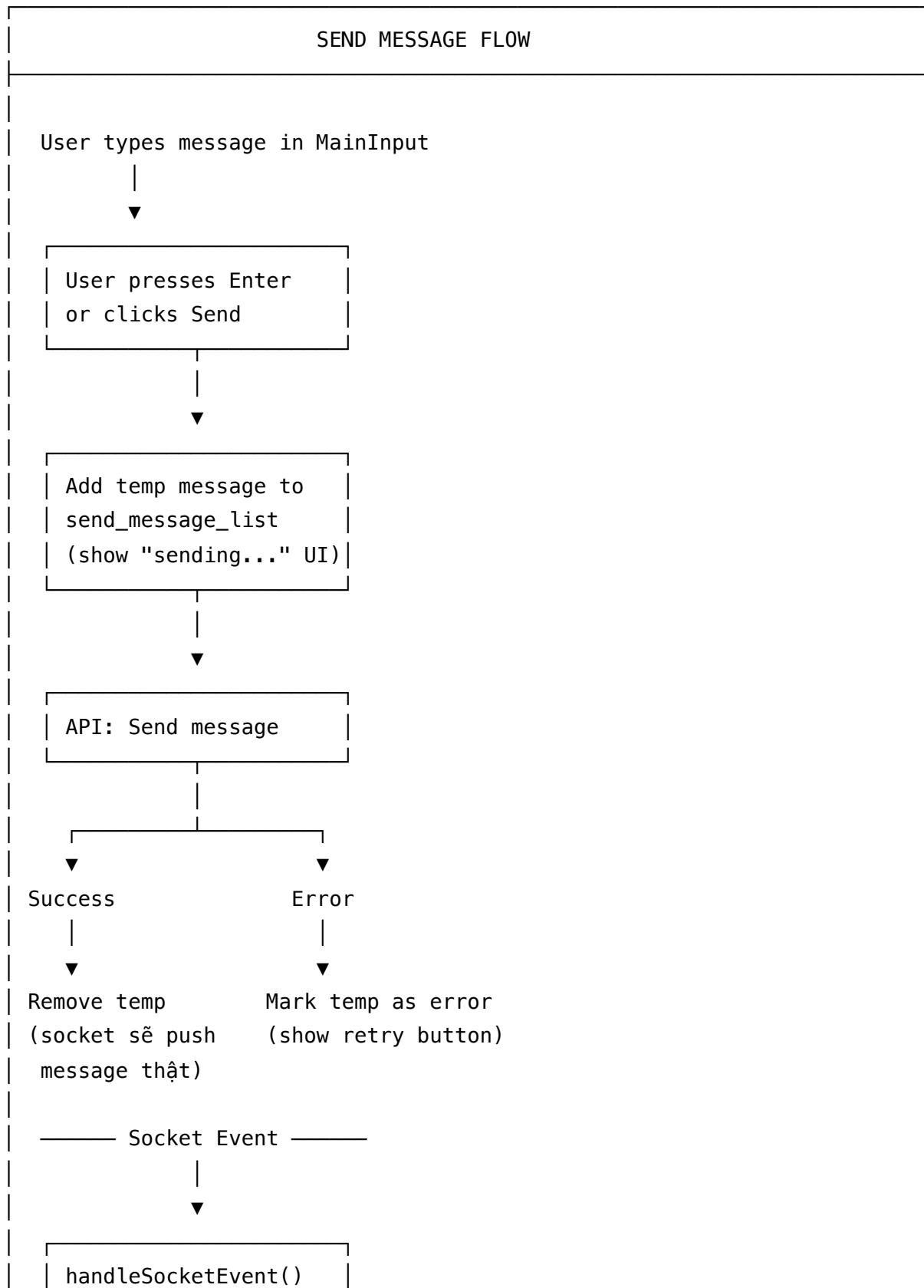
  // Actions
  setSelectConversation(conversation),
  updateConversation(conversation),
}
```

10.3 useMessageStore

```
{  
  // State  
  message_list: [],           // Tin nhắn của hội thoại hiện tại  
  send_message_list: [],      // Tin nhắn đang gửi (temp)  
  list_message_id: 'message-list', // DOM id  
  reply_comment: null,        // Reply comment (for Post)  
  modal_zalo_personal_ref: null, // Modal ref  
  message_data: null,         // Message cho modal  
  
  // Actions  
  addSendMessage(message),  
  removeSendMessage(id),  
  updateSendMessageStatus(id, status),  
}
```

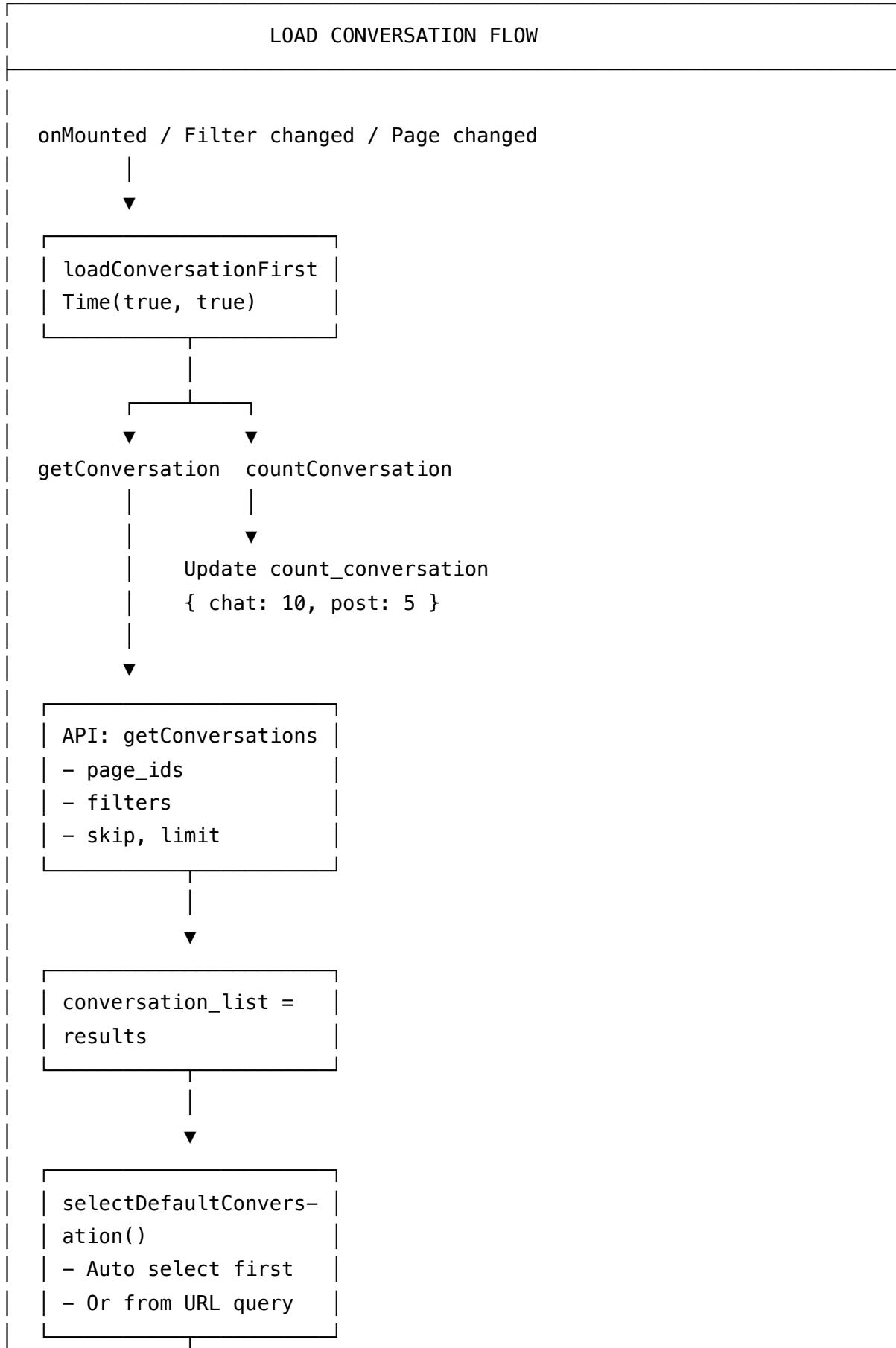
Flow Diagrams

Send Message Flow



```
|   | → Add real message    |
|   | → Update conversation |
|   | → Scroll to bottom   |
|   |
```

Load Conversation Flow



Load messages for
selected conversation

Related Files

- **Entry:** `src/views/ChatWarper.vue`
- **Layout:** `src/views/ChatWarper/Layout.vue`
- **Stores:** `src/stores/conversation.ts` , `src/stores/message.ts`
- **Socket:** `src/utils/helper/Socket.ts`
- **APIs:** `src/utils/api/N4Service/Conversation.ts` , `src/utils/api/N4Service/Message.ts`

Note: Chat Layout sử dụng kiến trúc event-driven với WebSocket cho real-time updates. Các component con lắng nghe custom events từ ChatWarper để cập nhật UI.