Svelte 5 + Hono 🔥

Part 7: bulkbar, search, limit, pagination, sort

www.youtube.com/@GetterSethya

Setup bulkbar

```
export type BulkbarArgs<Data> = {
	data: Row<Data>[];
	table: TTable<Data>;
	isMutating: (value: boolean) => void;
};

export type TableBulkbar<Data> = {
	onAction: (args: BulkbarArgs<Data>) => void;
};



type DataTableProps = {
	//...
	bulkbar?: TableBulkbar<TData>;
};

let { data, columns, actions, bulkbar }: DataTableProps = $props();
let isMutating = $state(false);
```

```
{#if bulkbar && table.getFilteredSelectedRowModel().rows.length > 0}
    <div class="fixed bottom-[10%] flex flex-1 flex-row items-center justify-center self-center text-sm">
        <div class="flex w-[50vw] flex-row rounded-lg border border-foreground/20 bg-muted p-2.5
text-muted-foreground shadow-sm max-sm:w-[80vw] lg:w-[40vw]">
            <div class="flex w-1/2 flex-row items-center gap-1">
                <span class="text-nowrap">
                    {table.getFilteredSelectedRowModel().rows.length} of
{table.getFilteredRowModel().rows
                        .length}
                    row(s) selected
                </span>
                <Button onclick={() => table.resetRowSelection(true)} size="sm"
                    variant="outline">
                    <RefreshCwIcon size={ICON_SIZE} />
                    Reset
                </Button>
            </div>
            <div class="ms-auto flex w-fit lg:w-1/2">
                <Button onclick={() => {
                        const rows = table.getFilteredSelectedRowModel().rows;
                        bulkbar.onAction({ data: rows, table,
                            isMutating: (value) => {isMutating = value;}});
                    }}
                    size="sm"
                    variant="destructive"
                    class="ms-auto"
                >
                    Delete
                </Button>
            </div>
        </div>
    </div>
{/if}
```

Company table page

```
const bulkbar: TableBulkbar<CompanyEntity> = {
      onAction: ({data, table,isMutating}) => {
            console.log(data.map((row) => row.original));
            table.resetRowSelection();
      }
]};
```

```
<DataTable {actions} data={$companyQuery.data.result.items} {columns} {bulkbar} />
```

Bulkbar custom transition

```
export const flyAndScale = (node: Element, params: FlyAndScaleParams = { y: -8, x: 0, start: 0.95,
duration: 150 }): TransitionConfig => {
      const style = getComputedStyle(node);
      const transform = style.transform === 'none' ? '' : style.transform;
      const scaleConversion = (valueA: number, scaleA: [number, number], scaleB: [number, number])
=> {
            const [minA, maxA] = scaleA;
            const [minB, maxB] = scaleB;
            const percentage = (valueA - minA) / (maxA - minA);
            const valueB = percentage * (maxB - minB) + minB;
            return valueB;
      };
      const styleToString = (style: Record<string, number | string | undefined>): string => {
            return Object.keys(style).reduce((str, key) => {
                  if (style[key] === undefined) return str;
                  return str + `${key}:${style[key]};`;
            }, '');
      };
```

Bulkbar custom transition

```
return {
          duration: params.duration ?? 200,
          delay: 0,
          css: (t) => {
               const y = scaleConversion(t, [0, 1], [params.y ?? 5, 0]);
               const x = scaleConversion(t, [0, 1], [params.x ?? 0, 0]);
               const scale = scaleConversion(t, [0, 1], [params.start ?? 0.95, 1]);
               return styleToString({
                    transform: `${transform} translate3d(${x}px, ${y}px, 0)
scale(${scale})`,
                    opacity: t
               });
          },
          easing: cubicOut
     };
};
```

Bulkbar custom transition

```
      <div
            transition:flyAndScale
            class="fixed bottom-[10%] flex flex-1 flex-row items-center justify-center
self-center text-sm"
      >
```

Search Component

```svelte
<script lang="ts">
     let search = $state(page.url.searchParams.get('q') || '');
</script>
<div class="flex h-fit flex-row rounded-lg border border-border max-sm:w-full">
     <Input placeholder="Search..." bind:value={search}
          class="h-auto border-none py-0 focus-visible:ring-0 focus-visible:ring-offset-0"/>
     {#if search !== ''}
          <Button size="sm" variant="ghost">
               <XIcon size={ICON_SIZE} />
          </Button>
     {:else}
          <Button size="sm" variant="ghost" class="hover:bg-transparent">
               <SearchIcon size={ICON_SIZE} />
          </Button>
     {/if}
</div>
```

Search Component

```
function handleValueChange(value: string) {
	if (value === '') return handleClear();
	search = value;
	page.url.searchParams.set('q', search);
	goto(`${page.url.pathname}?${page.url.searchParams.toString()}`, {
		keepFocus: true,
		noScroll: true
	});
}
function handleClear() {
	search = '';
	page.url.searchParams.delete('q');
	goto(`${page.url.pathname}?${page.url.searchParams.toString()}`, {
		keepFocus: true,
		noScroll: true
	});
}
```

Search Component

```
<Input
     oninput={() => handleValueChange(search)}
     placeholder="Search..."
     bind:value={search}
     class="h-auto border-none py-0 focus-visible:ring-0 focus-visible:ring-offset-0"
/>
{#if search !== ''}
     <Button onclick={handleClear} size="sm" variant="ghost">
          <XIcon size={ICON_SIZE} />
     </Button>
{:else}
     <Button size="sm" variant="ghost" class="hover:bg-transparent">
          <SearchIcon size={ICON_SIZE} />
     </Button>
{/if}
```

Search Component - Table company

```
<div class="flex flex-row items-center justify-end gap-2.5">
     <Search />
     <Button size="icon" variant="secondary" onclick={() => goto('/company/upsert')}>
          <PlusIcon size={ICON_SIZE} />
     </Button>
</div>
```

Search Component

```
npm install runed
```

```
    <Input
        oninput={useDebounce(() => handleValueChange(search), 500)}
        placeholder="Search..."
        bind:value={search}
        class="h-auto border-none py-0 focus-visible:ring-0 focus-visible:ring-offset-0"
    />
```

Limit component

```
export const LIMIT = [5, 10, 20, 50, 100] as const;
```

```
    type LimitProps = { value?: number[]; totalItems: number };

    let { value, totalItems }: LimitProps = $props();
    let limit = $state(Number(page.url.searchParams.get('limit') ?? 10));
    let pageNumber = $state(Number(page.url.searchParams.get('page') ?? 1));
```

Limit component

```
<Select.Root type={'single'} value={limit.toString()}>
      <Select.Trigger class="w-[100px] text-sm font-medium">
            Limit: {limit}
      </Select.Trigger>
      <Select.Content>
            {#if value}
                  {#each value as l}
                        <Select.Item value={l.toString()}>{l}</Select.Item>
                  {/each}
            {:else}
                  {#each LIMIT as l}
                        <Select.Item value={l.toString()}>{l}</Select.Item>
                  {/each}
            {/if}
      </Select.Content>
</Select.Root>
```

Limit component - utils.ts

```
export const recalculatePage = (currentPage: number, totalRow: number, deletedDataLength: number, limit:
number) => {
      let page = currentPage
      const totalItem = totalRow - deletedDataLength
      const totalPage = Math.ceil(totalItem / limit)
      const diffPage = totalPage - page
      if (diffPage < 0) {
            page = page - Math.abs(diffPage)
      }

      return page
}
```

Limit component

```
function handleValueChange(value: string) {
      limit = Number(value)
      const recal = recalculatePage(pageNumber, totalItems, 0, limit)
      page.url.searchParams.set('limit', limit.toString())
      page.url.searchParams.set('page', recal.toString())
      goto(`${page.url.pathname}?${page.url.searchParams.toString()}`, {
            keepFocus: true,
            noScroll: true
      })
}
```

Limit component

```
<Select.Root onValueChange={handleValueChange} type={'single'} value={limit.toString()}>
      <Select.Trigger class="w-[100px] text-sm font-medium">
            Limit: {limit}
      </Select.Trigger>
      <Select.Content>
            {#if value}
                  {#each value as l}
                        <Select.Item value={l.toString()}>{l}</Select.Item>
                  {/each}
            {:else}
                  {#each LIMIT as l}
                        <Select.Item value={l.toString()}>{l}</Select.Item>
                  {/each}
            {/if}
      </Select.Content>
</Select.Root>
```

Pagination component

```
type PaginationProps = {
      totalItems: number;
      page?: number;
};

let {
      totalItems,
      page = $bindable(Number(pageStore.url.searchParams.get('page') ?? 1))
}: PaginationProps = $props();
let perPage = $state(Number(pageStore.url.searchParams.get('limit') ?? 10));
```

Pagination component

```
$effect(() => {
      if (navigating) {
            perPage = Number(pageStore.url.searchParams.get('limit') ?? 10);
            page = Number(pageStore.url.searchParams.get('page') ?? 1);
      }
});
```

Pagination component

```
<Pagination.Root count={totalItems} bind:page {perPage} class="mx-0 w-fit">
	{#snippet children({ pages, currentPage })}
		<Pagination.Content>
			<Pagination.Item>
				<Pagination.PrevButton />
			</Pagination.Item>
			{#each pages as page (page.key)}
				{#if page.type === 'ellipsis'}
					<Pagination.Item>
						<Pagination.Ellipsis />
					</Pagination.Item>
				{:else}
					<Pagination.Item>
						<Pagination.Link {page} isActive={currentPage === page.value}>
							{page.value}
						</Pagination.Link>
					</Pagination.Item>
				{/if}
			{/each}
			<Pagination.Item>
				<Pagination.NextButton />
			</Pagination.Item>
		</Pagination.Content>
	{/snippet}
</Pagination.Root>
```

Table company

```
<div class="flex flex-row justify-between">
       <Limit totalItems={$companyQuery.data.result.meta.totalItems} />
       <Pagination totalItems={$companyQuery.data.result.meta.totalItems} />
</div>
```

SearchParams Hooks

```
export class SearchParams {
	searchParams = () => new URLSearchParams(page.url.searchParams);
	get page() {
		return Number(this.searchParams().get('page') || '1');
	}

	get limit() {
		return Number(this.searchParams().get('limit') || '10');
	}

	get sort() {
		return this.searchParams().get('sort') ?? 'created_at';
	}

	get order() {
		return this.searchParams().get('order') ?? 'DESC';
	}

	get search() {
		return this.searchParams().get('q') ?? '';
	}
}
```

Table company

```
const searchParams = new SearchParams();
queryKey: ['company', searchParams.page, searchParams.limit, searchParams.sort, searchParams.order, searchParams.search]

query: {
       page: searchParams.page.toString(),
       limit: searchParams.limit.toString(),
       sort: searchParams.sort,
       order: searchParams.order,
       q: searchParams.search
}


afterNavigate(() => $companyQuery.refetch());
```

Table company

```
const bulkbarMutation = createMutation({
    mutationFn: async (data: Row<CompanyEntity>[]) => {
        for (const company of data.map((d) => d.original)) {
            await $client.company.id[':id'].$delete(
                { param: { id: company.id.toString() } },
                { init: { headers: { Authorization: localStorage.getItem(JWT.ACCESS_TOKEN) || '' } } }
            );
        }
    },
    onSuccess: () => {
        $companyQuery.refetch();
        toast.success('Company deleted successfully');
    },
    onError: (err) => {
        console.error(err);
        toast.error('Error when deleting data');
    }
});
```

Sort Component

```
<script lang="ts">
	import type { ComponentProps } from 'svelte';
	import { Button } from '../button';
	import { ArrowUpDown } from 'lucide-svelte';
	import { cn } from '@/utils';

	type TableSortProps = ComponentProps<typeof Button> & {
		value: string;
		key: string;
	};

	let { variant = 'ghost',class:className, value, key, ...restProps }: TableSortProps = $props();

</script>

<Button
    {variant}
    class={cn(["w-fit hover:bg-transparent justify-start p-0 hover:bg-transparent",className])} {...restProps}>
       {value}
       <ArrowUpDown class="ml-2 size-4" />
</Button>
```

Company table

```
{
      accessorKey: 'name',
      header: () => {
            return renderComponent(Sort, {
                  value: 'Name',
                  key: 'name',
            });
      },
      enableHiding: false
},
```

Sort Component

```
        function handleClick() {
                const searchParams = new URLSearchParams(page.url.searchParams)
                const currentOrder = searchParams.get('order') ?? 'DESC'
                page.url.searchParams.set('sort', key)
                page.url.searchParams.set('order', currentOrder === 'ASC' ? 'DESC' : 'ASC')

                goto(`${page.url.pathname}?${page.url.searchParams.toString()}`, {
                        keepFocus: true,
                        noScroll: true
                })
        }
```

```
<Button
    onclick={handleClick}
    {variant}
    class={cn(["w-fit hover:bg-transparent justify-start p-0 hover:bg-transparent",className])} {...restProps}
>
       {value}
       <ArrowUpDown class="ml-2 size-4" />
</Button>
```

Custom Cell - utils.ts

```ts
export const idDateFormat = new Intl.DateTimeFormat('id-ID', {
       dateStyle: 'short'
})

export const idTimeFormat = new Intl.DateTimeFormat('id-ID', {
       hour12: false,
       timeStyle: 'medium'
})

export const idDateTimeFormat = new Intl.DateTimeFormat('id-ID', {
       hour12: false,
       timeStyle: 'long',
       dateStyle: 'medium'
})
```

Custom Cell - date-compact

```
type CompactDateProps = {dateString: number}
let { dateString }: CompactDateProps = $props()
const dateObj = $derived(new Date(dateString))
```

```
<Tooltip.Root>
	<Tooltip.Trigger>
		<div class="flex flex-col">
			<span class="text-sm">{idDateFormat.format(dateObj)}</span>
			<span class="text-sm text-foreground/50">{idTimeFormat.format(dateObj).replaceAll('.', ':')}</span>
		</div>
	</Tooltip.Trigger>
	<Tooltip.Content>
		<span class="text-foreground/50">{idDateTimeFormat.format(dateObj)}</span>
	</Tooltip.Content>
</Tooltip.Root>
```

Custom Cell - date-compact

```
{
        id: 'Updated',
        accessorKey: 'updated_at',
        header: () =>
                renderComponent(Sort, {value: 'Updated',key: 'updated_at'}),
        cell: ({ row }) =>
                renderComponent(CompactDate, {
                        dateString: row.original.updated_at
                }),
        size: 1
},
{
        id: 'Created',
        accessorKey: 'created_at',
        header: () =>
                renderComponent(Sort, {
                        value: 'Created',
                        key: 'created_at'
                }),
        cell: ({ row }) =>
                renderComponent(CompactDate, {
                        dateString: row.original.created_at
                }),
        size: 1
},
```