Сучасні каркаси web-додатків

Стан застосунку та автентифікація

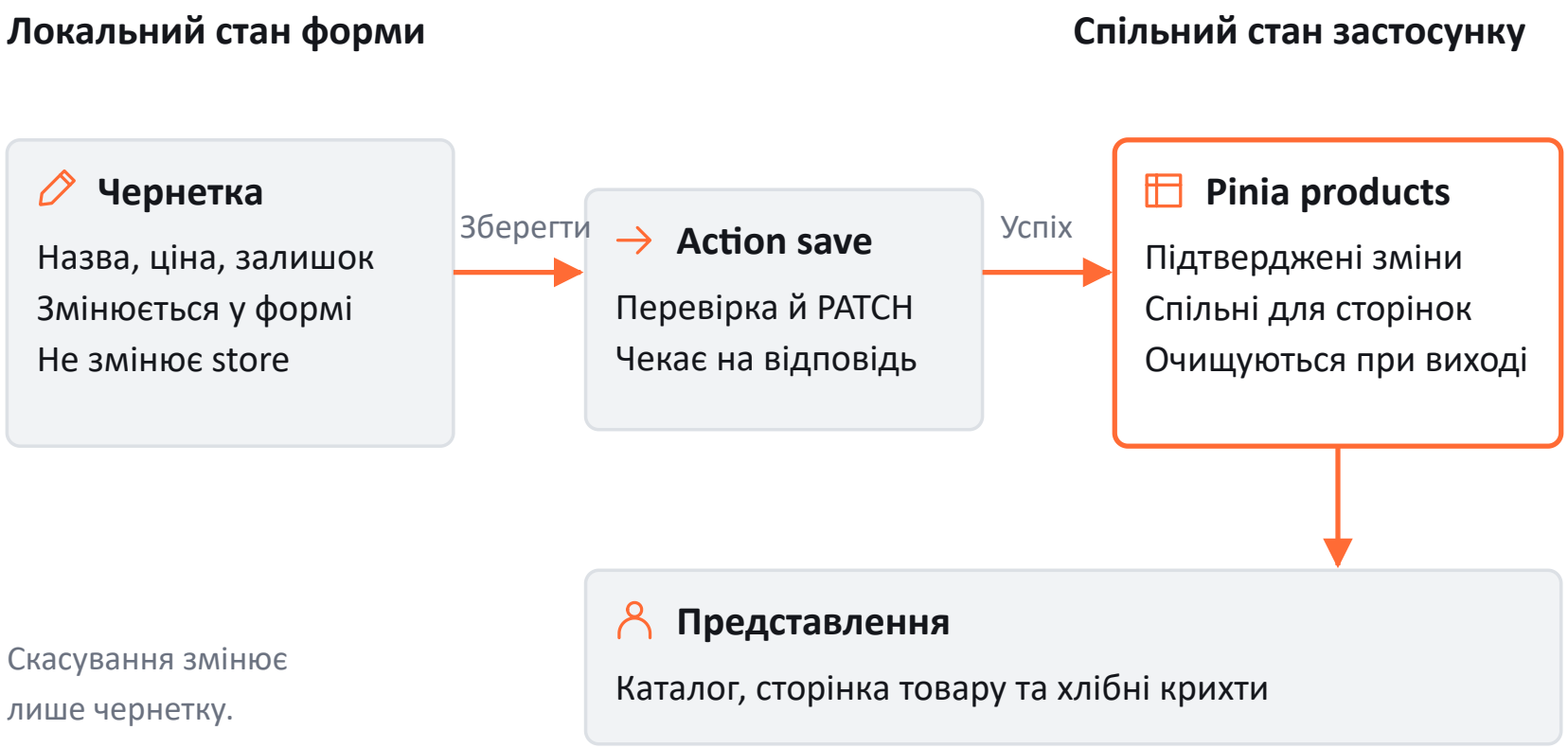
Pinia, приватні маршрути та редагування товарів

Лекція 7 · Лісняк Андрій Олександрович

Хто володіє станом

Поняття	Зміст
Компонент	Поля форми, помилки валідації, ознака незбережених змін.
URL	Ідентифікатор товару, пошук, сторінка списку.
Store	Перевірений сеанс і підтверджені зміни, спільні для сторінок.
Сервер	Облікові записи, перевірка токена та правила доступу до API.

Від чернетки до спільних даних



Скасування змінює форму; успішне збереження оновлює спільний стан.

Composable, provide/inject і store

Поняття	Зміст
Composable	Повторно використовує логіку; стан усередині виклику зазвичай окремий.
provide/inject	Передає залежність нащадкам певного піддерева компонентів.
Pinia store	Має ідентифікатор, спільний екземпляр у Pinia та підтримку DevTools.
Поєднання	Store містить спільний стан; composable керує життєвим циклом запиту.

Setup Store у Pinia

Поняття	Зміст
defineStore(id, setup)	Оголошує функцію use…Store; екземпляр створюється для конкретної Pinia.
ref()	Повернений реактивний стан сховища.
computed()	Похідне значення — getter.
Функція	Дія — action; може виконувати асинхронний запит.
Повернення і reset	Власний реактивний стан повертають; reset Setup Store пишуть явно.

Підключення Pinia перед Router

main.js

```
import { createApp } from 'vue'
import App from './App.vue'
import { pinia } from './stores/index.js'
import { router } from './router/index.js'
import './styles/main.css'
import './styles/router.css'
import './styles/auth.css'

createApp(App).use(pinia).use(router).mount('#app')
```

Реактивний стан сеансу

auth.js · рядки 6–14

```
export const useAuthStore = defineStore('auth', () => {
  const accessToken = ref(readToken())
  const user = ref(null)
  const status = ref('idle')
  const error = ref('')
  const isAuthenticated = computed(() =>
    status.value === 'authenticated' && Boolean(accessToken.value && user.value),
  )
  const isBusy = computed(() => ['restoring', 'signing-in'].includes(status.value))
```

Стан застосунку та автентифікація

storeToRefs зберігає реактивний зв’язок

ProductEditView.vue · рядки 12–18

```
const route = useRoute()
const auth = useAuthStore()
const edits = useProductsStore()
const { saving, error: saveError } = storeToRefs(edits)
const validId = computed(() => /^[1-9]\d*$/.test(String(route.params.id)))
const resource = computed(() => validId.value ? `/products/${route.params.id}?delay=500` : null)
const { data, error, isLoading, refetch } = usePrivateResource(resource)
```

Автентифікація та доступ

Поняття	Зміст
Автентифікація	Сервер установлює, хто виконує запит.
Авторизація	Сервер перевіряє право на конкретну операцію чи ресурс.
Guard	Керує переходом до сторінки в браузері.
Приватний API	Перевіряє Bearer-токен незалежно від guard і видимості кнопок.
Навчальна модель	Усі користувачі демо входять до редактора; ролі не моделюються.

Контракт DummyJSON

Поняття	Зміст
POST /auth/login	Логін і пароль → accessToken та дані профілю.
GET /auth/me	Bearer-токен → перевірений профіль користувача.
GET /auth/products	Приватне читання списку або /:id.
PATCH /auth/products/:id	Імітована зміна чотирьох полів; результат повертається у відповіді.
Початкові дані	Новий GET повертає початковий товар; сервер зміни не зберігає.

Вхід: запит і підтвердження

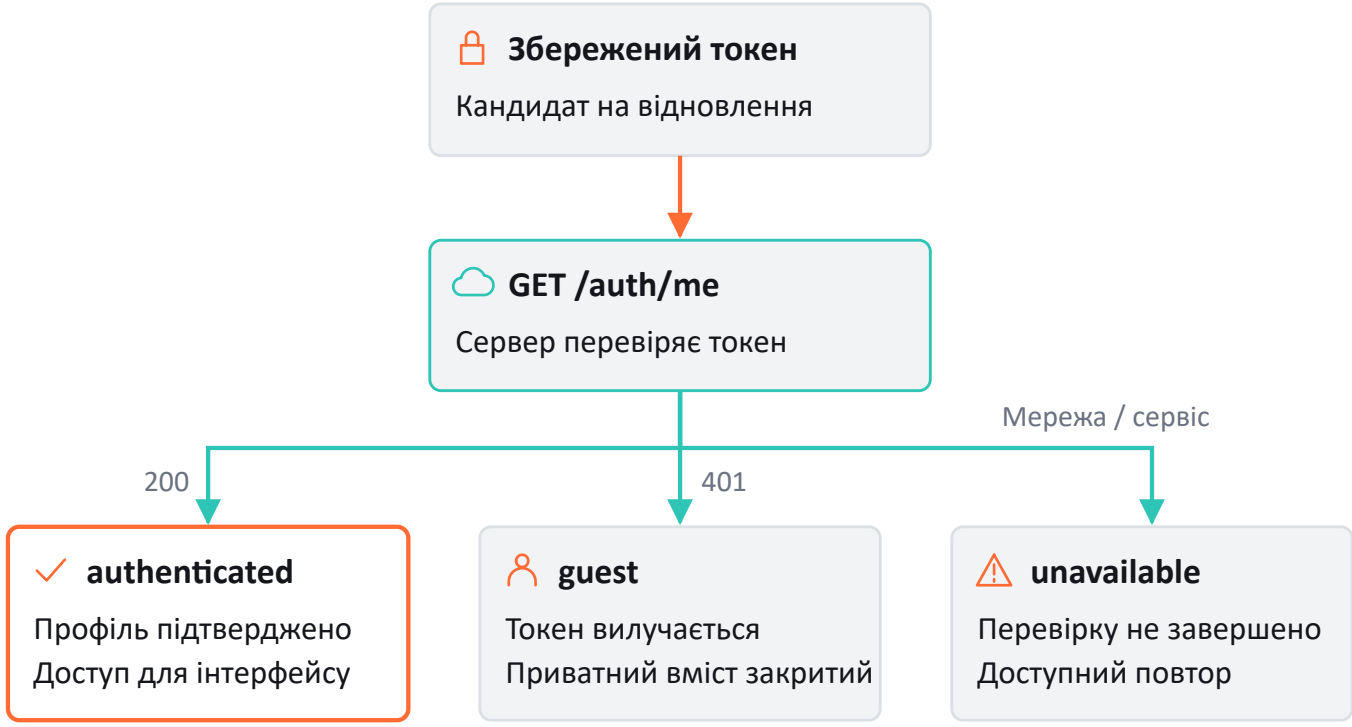
auth.js · рядки 35–46

```
    try {
      const result = await requestJson('/auth/login', {
        method: 'POST',
        body: { username: username.trim(), password, expiresInMins: 30 },
        signal: controller.signal,
      })
      if (current !== generation) return false
      accessToken.value = result.accessToken
      user.value = profileFrom(result)
      writeToken(accessToken.value)
      status.value = 'authenticated'
      return true
```

Пам’ять, sessionStorage і вихід

Поняття	Зміст
Pinia	Стан поточного запуску; перезавантаження створює новий екземпляр.
sessionStorage	Кандидат-токен для відновлення в межах вкладки; не доказ входу.
Перезавантаження	Профіль знову перевіряється через /auth/me.
Logout	Очищення локального сеансу й змін товарів; JWT на сервері не відкликається.
Межа прикладу	JS має доступ до токена; production-політика зберігання залежить від сервера.

Як відновлюється сеанс



Якщо збереженого токена немає, установлюється guest без запиту.

401 завершує сеанс; мережева помилка залишає можливість повторної перевірки.

restore: очікування перевірки профілю

auth.js · рядки 71–83

```
      const profile = await requestJson('/auth/me', {
        token: accessToken.value, signal: controller.signal,
      })
      if (current !== generation) return false
      user.value = profileFrom(profile)
      status.value = 'authenticated'
      return true
    } catch (cause) {
      if (current !== generation) return false
      if (cause.status === 401) logout()
      else {
        user.value = null
        status.value = 'unavailable'
```

Guard очікує завершення restore

guards.js · рядки 4–15

```
export function installAuthGuard(router, pinia) {
  router.beforeEach(async to => {
    const auth = useAuthStore(pinia)
    await auth.restore()
    if (to.meta.requiresAuth && !auth.isAuthenticated) {
      return { name: 'login', query: { redirect: to.fullPath } }
    }
    if (to.name === 'login' && auth.isAuthenticated) {
      return resolveReturnTarget(router, to.query.redirect)
    }
  })
}
```

Повернення лише до відомого маршруту

session.js · рядки 18–26

```
export function resolveReturnTarget(router, value) {
  const fallback = { name: 'manage-products' }
  if (typeof value !== 'string' || !value.startsWith('/') || value.startsWith('//')) return fallback
  const resolved = router.resolve(value)
  return ['manage-products', 'manage-edit'].includes(resolved.name)
    ? { path: resolved.path, query: resolved.query, hash: resolved.hash }
    : fallback
}
```

Приватні запити та відповідь 401

auth.js · рядки 94–101

```
async function request(resource, options = {}) {
  if (!isAuthenticated.value) throw new HttpError(401, 'Потрібно виконати вхід')
  const token = accessToken.value
  try {
    return await requestJson(`/auth${resource}`, { ...options, token })
  } catch (cause) {
    if (cause.status === 401 && token === accessToken.value) logout()
    throw cause
```

Окрема чернетка і вихідний стан

ProductEditView.vue · рядки 23–38

```
const draft = reactive({ title: '', price: '', stock: '', description: '' })
const baseline = ref('')
const attempted = ref(false)
const message = ref('')
const form = ref(null)
const errors = computed(() => attempted.value ? normalizeProductDraft(draft).errors : {})
const dirty = computed(() => Boolean(product.value) && JSON.stringify(draft) !== baseline.value)

function resetDraft() {
  if (!product.value) return
  Object.assign(draft, productDraft(product.value))
  baseline.value = JSON.stringify(draft)
  attempted.value = false
  message.value = ''
}
watch(product, resetDraft, { immediate: true })
```

Нормалізація та валідація полів

productDraft.js · рядки 8–18

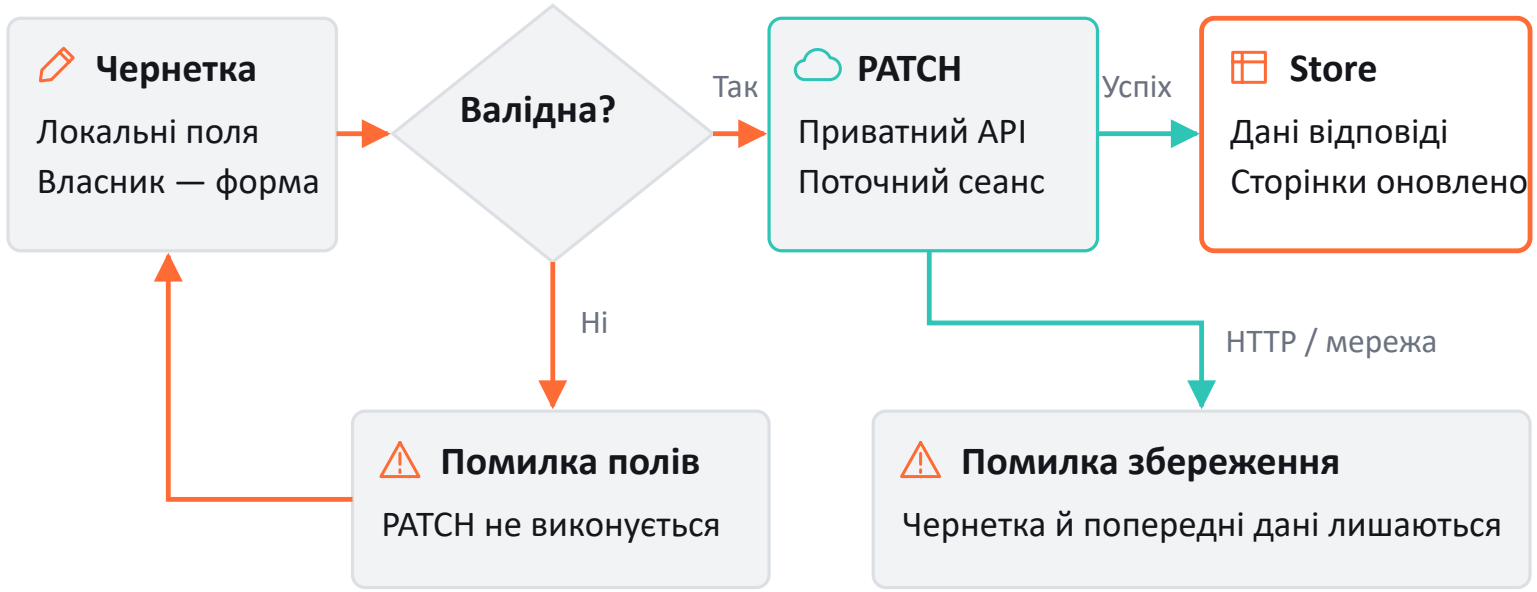
```
  const errors = {}
  if (!values.title || values.title.length > 120)
    errors.title = 'Введіть назву від 1 до 120 символів.'
  if (draft.price === '' || !Number.isFinite(values.price) || values.price <= 0)
    errors.price = 'Ціна має бути додатним числом.'
  if (draft.stock === '' || !Number.isInteger(values.stock) || values.stock < 0)
    errors.stock = 'Залишок має бути цілим числом від 0.'
  if (!values.description || values.description.length > 2000)
    errors.description = 'Введіть опис від 1 до 2000 символів.'
  return { values, errors }
}
```

Збереження: відповідь передує зміні store

products.js · рядки 38–45

```
    error.value = ''
    try {
      const saved = await auth.request(`/products/${id}`, {
        method: 'PATCH', body: values, signal: controller.signal,
      })
      if (current !== generation || token !== auth.accessToken || !auth.isAuthenticated) return null
      overrides.value = { ...overrides.value, [id]: productDraft(saved) }
      return saved
```

Результат збереження та помилки



Перед оновленням store перевіряються актуальність операції та сеансу.

401 окремо завершує сеанс і закриває приватний інтерфейс.

> Помилка не змінює підтверджені дані; 401 окремо закриває приватний вміст.

Скасування запиту і покоління операції

products.js · рядки 17–28

```
function cancelSave() {
  generation += 1
  controller?.abort()
  saving.value = false
  error.value = ''
}

function reset() {
  cancelSave()
  overrides.value = {}
}
```

Захист незбережених правок

ProductEditView.vue · рядки 56–68

```
function canLeave() {
  if (!auth.isAuthenticated || !dirty.value) return true
  return window.confirm('Залишити сторінку й відкинути незбережені зміни?')
}
onBeforeRouteLeave(canLeave)
onBeforeRouteUpdate(to => to.params.id === route.params.id || canLeave())
function preventUnload(event) {
  if (dirty.value && auth.isAuthenticated) { event.preventDefault(); event.returnValue = '' }
}
window.addEventListener('beforeunload', preventUnload)
onBeforeUnmount(() => {
  window.removeEventListener('beforeunload', preventUnload)
  edits.cancelSave()
```

Що залишається після помилки

Поняття	Зміст
400 під час login	Повідомлення про облікові дані; повторна спроба входу.
401 приватного API	Сеанс очищено, приватний вміст закрито, потрібен вхід.
Мережа під час restore	Доступ закрито; токен-кандидат лишається для повторної перевірки.
Помилка PATCH	Чернетка й підтверджені дані не втрачаються; дозволено повтор.
Запізнілий результат	Не змінює поточний сеанс або вже скасовану операцію.

Імітація збереження в DummyJSON

Поняття	Зміст
PATCH	Повертає змінений об’єкт, але не зберігає його на сервері.
Локальні зміни	Store накладає підтверджені поля на отримані товари за id.
Узгодженість	Редактор, список, сторінка товару й крихти бачать ті самі поля.
Reload або logout	Локальні зміни очищуються; новий GET повертає початкові дані.
Серверний пошук	Шукає у початкових даних DummyJSON, без локальних змін назви.

Перевірка перед лабораторною

1. **Доступ** Приватний URL гостем → вхід → повернення; reload перевіряє /auth/me.
2. **Форма** Помилки полів не надсилають PATCH; скасування відновлює значення.
3. **Спільний стан** Один PATCH → оновлення списку, товару й крихт; reload скидає зміни.
4. **Сеанс і мережа** Перевірити 401, помилку збереження та вихід під час запиту.
5. **Інтерфейс** Клавіатура, чотири розміри екрана та виробнича збірка.

Ваші запитання

Власник стану → перевірений сеанс → підтверджені зміни