ANEXO · TAREAS DEL ALUMNADO

OJOS PARA LA MÁQUINA: TESTS VISUALES Y DOCUMENTACIÓN

Nombre: _____ Fecha: _____

BUSCA LAS DIFERENCIAS (6 MIN)

Abre el archivo de Figma, **Spot the difference**. Solo, sin zoom, 60 segundos de cronómetro. Izquierda: la app aprobada el viernes por la mañana. Derecha: después del commit del viernes por la tarde.

¿Cuántas diferencias ves? _____ ¿Cuál detectaría un test unitario? _____

TUS TESTS VISUALES (20 MIN)

Individual, en **tu propio proyecto**:

OK	HAZ	ESTÁ HECHO CUANDO…
☐	Instala Playwright (`npm install -D @playwright/test`) y crea `playwright.config.js` como el del profesor (es idéntico). Añade `test-results` y `playwright-report` a `.gitignore` y el script `"test:visual": "playwright test"`.	`npx playwright test` arranca Storybook solo.
☐	Carpeta `e2e/` con un spec por componente: cada historia de Button y NoteRow tiene un test de captura. El id de la historia: ábrela en una pestaña nueva y lee `?id=...`.	Hay un test por historia.
☐	Ejecuta dos veces y **mira las capturas base** (carpetas `-snapshots`). Haz commit de ellas.	La segunda ejecución sale toda en verde y has mirado cada PNG.
☐	Rompe algo a propósito (un padding, un token de color), ejecuta, abre el informe (`npx playwright show-report`) y **guarda la imagen diff**. Repara.	Tienes la imagen diff guardada (es parte de la entrega).

Un test visual en rojo es una **pregunta**: si no querías cambiar el aspecto, es un bug y arreglas el código. Si sí querías, miras cada diff y solo entonces ejecutas `npx playwright test --update-snapshots`.

Si algo falla:

- **La captura base muestra la página «story not found» de Storybook.** El id de la historia está mal. Por eso hay que mirar el PNG.
- **`browserType.launch: Executable doesn't exist.`** Falta ejecutar `npx playwright install chromium` en ese ordenador.
- **Vitest intenta ejecutar `e2e/*.spec.js`** (o Playwright los tests unitarios). Falta el `include: ['src/**/*.test.js']` del viernes en `vite.config.js`.
- **Un test de UndoBar falla una de cada tres veces.** La historia tiene `play` y la captura se hace antes de que acabe. Espera primero a lo que quieres ver: `await expect(page.getByRole('status')).toHaveText(/deleted/)`.
- **Hiciste commit de `test-results/`.** Va en `.gitignore`. Las carpetas `-snapshots` **sí** se suben.

Antes del descanso: al menos una ejecución en verde y un diff guardado.

DOCUMENTA TU LIBRERÍA (24 MIN)

Individual, en **tu propio proyecto**, con la checklist del archivo de Figma abierta:

OK	HAZ	ESTÁ HECHO CUANDO…
☐	`tags: ['autodocs']` en cada archivo de historias.	Aparece una entrada **Docs** encima de las historias de cada componente.
☐	Lee cada página Docs como lo haría un desconocido. Reescribe cada descripción de prop que solo repite el tipo.	Cada descripción dice lo que el tipo no dice: cuándo, por qué, qué pasa con 0.
☐	Una línea (comentario HTML) encima de la primera historia de cada componente: cuándo usarlo.	La línea sale en la página Docs.
☐	`src/lib/Introduction.mdx` para **tu** librería: qué es, cómo se importa y tus reglas de la casa (al menos dos, de tu sistema de diseño de la UT1).	La página aparece en la barra lateral.
☐	`npm run build-storybook`.	Termina sin errores.

Si algo falla:

- **La tabla de props está vacía o sin descripciones.** El `@typedef` de JSDoc no está unido a `$props()` con `/** @type {Props} */`.
- **Tus descripciones describen el tipo** («string», «el título»). Pregúntate: ¿cuándo **no** pasaría esta prop? La respuesta es la descripción.
- **La página MDX no aparece.** El archivo está fuera de `src/`, o falta `<Meta title="..." />`.
- **Subiste `storybook-static/` a git.** Es un resultado de compilación; ya está en `.gitignore`.

Intercambio (últimos 6 minutos). Cambia de sitio con tu pareja. Usa un componente de su librería en una página de pruebas **leyendo solo la página Docs**: sin código fuente y sin preguntas. Cada pregunta que te apetezca hacer es una línea de documentación que falta: apúntala aquí para el autor.

ENTREGA

Haz commit y push (con las capturas base). En Classroom: el enlace del repositorio, **una imagen diff** de **Tus tests visuales** con una línea que diga qué rompiste, y la nota que te escribió tu pareja en el intercambio.

PARA CASA

- Tests visuales para cada historia de tu propio componente, con las capturas base subidas.
- Arregla cada nota que te dejó tu pareja en el intercambio.
- Ejercicio de transferencia (un párrafo). Tu equipo cambia el color de marca y 43 tests visuales se ponen en rojo a la vez. Describe paso a paso qué haces antes de ejecutar `--update-snapshots`, y qué harías distinto si solo 2 de los 43 se vieran mal en el informe.